

# MariaDB Server Documentation

[MariaDB Knowledge Base](#)

For any errors please see [Bug Reporting](#)

Document generated on: 2022-03-11

# Chapter Contents

## 1 SQL Statements and Structure

### 1.1 SQL Statements

## 2 MariaDB Administration

### 2.1 Getting, Installing and Upgrading MariaDB

## 3 High Availability & Performance Tuning

## 4 Columns, Storage Engines, and Plugins

### 4.1 Data Types

### 4.4 Plugins

## 5 MariaDB Community Bug Reporting

# Table of Contents

## 1 SQL Statements and Structure

### 1.1 SQL Statements

#### 1.1.1 Account Management SQL Commands

1.1.1.1 CREATE USER

1.1.1.2 ALTER USER

1.1.1.3 DROP USER

1.1.1.4 GRANT

1.1.1.5 RENAME USER

1.1.1.6 REVOKE

1.1.1.7 SET PASSWORD

1.1.1.8 CREATE ROLE

1.1.1.9 DROP ROLE

1.1.1.10 SET ROLE

1.1.1.11 SET DEFAULT ROLE

1.1.1.12 SHOW GRANTS

1.1.1.13 SHOW CREATE USER

#### 1.1.2 Administrative SQL Statements

##### 1.1.2.1 Table Statements

1.1.2.1.1 ALTER

1.1.2.1.1.1 ALTER TABLE

1.1.2.1.1.2 ALTER DATABASE

1.1.2.1.1.3 ALTER EVENT

1.1.2.1.1.4 ALTER FUNCTION

1.1.2.1.1.5 ALTER LOGFILE GROUP

1.1.2.1.1.6 ALTER PROCEDURE

1.1.2.1.1.7 ALTER SEQUENCE

1.1.2.1.1.8 ALTER SERVER

1.1.2.1.1.9 ALTER TABLESPACE

1.1.2.1.1.10 ALTER USER

1.1.2.1.2 ALTER VIEW

1.1.2.1.3 ANALYZE TABLE

1.1.2.1.4 CHECK TABLE

1.1.2.1.5 CHECK VIEW

1.1.2.1.6 CHECKSUM TABLE

1.1.2.1.7 CREATE TABLE

1.1.2.1.8 DELETE

1.1.2.1.9 DROP TABLE

1.1.2.1.10 Installing System Tables (`mysql_install_db`)

1.1.2.1.11 mysqlcheck

1.1.2.1.12 OPTIMIZE TABLE

1.1.2.1.13 RENAME TABLE

1.1.2.1.14 REPAIR TABLE

1.1.2.1.15 REPAIR VIEW

1.1.2.1.16 REPLACE

1.1.2.1.17 SHOW COLUMNS

1.1.2.1.18 SHOW CREATE TABLE

1.1.2.1.19 SHOW INDEX

1.1.2.1.20 TRUNCATE TABLE

1.1.2.1.21 UPDATE

1.1.2.1.22 IGNORE

1.1.2.1.23 System-Versioned Tables

## 1.1.2.2 ANALYZE and EXPLAIN Statements

1.1.2.2.1 ANALYZE FORMAT=JSON

1.1.2.2.2 ANALYZE FORMAT=JSON Examples

1.1.2.2.3 ANALYZE Statement

1.1.2.2.4 EXPLAIN

1.1.2.2.5 EXPLAIN ANALYZE

1.1.2.2.6 EXPLAIN FORMAT=JSON

1.1.2.2.7 SHOW EXPLAIN

1.1.2.2.8 Using Buffer UPDATE Algorithm

## 1.1.2.3 BACKUP Commands

1.1.2.3.1 BACKUP STAGE

1.1.2.3.2 BACKUP LOCK

1.1.2.3.3 Mariabackup and BACKUP STAGE Commands

1.1.2.3.4 Storage Snapshots and BACKUP STAGE Commands

## 1.1.2.4 FLUSH Commands

1.1.2.4.1 FLUSH

1.1.2.4.2 FLUSH QUERY CACHE

1.1.2.4.3 FLUSH TABLES FOR EXPORT

## 1.1.2.5 Replication Commands

1.1.2.5.1 CHANGE MASTER TO

1.1.2.5.2 START SLAVE

1.1.2.5.3 STOP SLAVE

1.1.2.5.4 RESET SLAVE

1.1.2.5.5 SET GLOBAL SQL\_SLAVE\_SKIP\_COUNTER

1.1.2.5.6 SHOW RELAYLOG EVENTS

1.1.2.5.7 SHOW SLAVE STATUS

1.1.2.5.8 SHOW MASTER STATUS

1.1.2.5.9 SHOW SLAVE HOSTS

1.1.2.5.10 RESET MASTER

1.1.2.6 Plugin SQL Statements

1.1.2.6.1 SHOW PLUGINS

1.1.2.6.2 SHOW PLUGINS SONAME

1.1.2.6.3 INSTALL PLUGIN

1.1.2.6.4 UNINSTALL PLUGIN

1.1.2.6.5 INSTALL SONAME

1.1.2.6.6 UNINSTALL SONAME

1.1.2.6.7 mysql\_plugin

1.1.2.7 SET Commands

1.1.2.7.1 SET

1.1.2.7.2 SET CHARACTER SET

1.1.2.7.3 SET GLOBAL SQL\_SLAVE\_SKIP\_COUNTER

1.1.2.7.4 SET NAMES

1.1.2.7.5 SET PASSWORD

1.1.2.7.6 SET ROLE

1.1.2.7.7 SET SQL\_LOG\_BIN

1.1.2.7.8 SET STATEMENT

1.1.2.7.9 SET TRANSACTION

1.1.2.7.10 SET Variable

1.1.2.8 SHOW

1.1.2.8.1 About SHOW

1.1.2.8.2 Extended Show

1.1.2.8.3 SHOW AUTHORS

1.1.2.8.4 SHOW BINARY LOGS

1.1.2.8.5 SHOW BINLOG EVENTS

1.1.2.8.6 SHOW CHARACTER SET

1.1.2.8.7 SHOW CLIENT\_STATISTICS

1.1.2.8.8 SHOW COLUMNS

1.1.2.8.9 SHOW CONTRIBUTORS

1.1.2.8.10 SHOW CREATE DATABASE

1.1.2.8.11 SHOW CREATE EVENT

1.1.2.8.12 SHOW CREATE FUNCTION

1.1.2.8.13 SHOW CREATE PACKAGE

1.1.2.8.14 SHOW CREATE PACKAGE BODY

1.1.2.8.15 SHOW CREATE PROCEDURE

1.1.2.8.16 SHOW CREATE SEQUENCE

1.1.2.8.17 SHOW CREATE TABLE

1.1.2.8.18 SHOW CREATE TRIGGER

1.1.2.8.19 SHOW CREATE USER

1.1.2.8.20 SHOW CREATE VIEW

1.1.2.8.21 SHOW DATABASES

1.1.2.8.22 SHOW ENGINE

1.1.2.8.23 SHOW ENGINE INNODB STATUS  
1.1.2.8.24 SHOW ENGINES  
1.1.2.8.25 SHOW ERRORS  
1.1.2.8.26 SHOW EVENTS  
1.1.2.8.27 SHOW FUNCTION STATUS  
1.1.2.8.28 SHOW GRANTS  
1.1.2.8.29 SHOW INDEX  
1.1.2.8.30 SHOW INDEX\_STATISTICS  
1.1.2.8.32 SHOW LOCALES  
1.1.2.8.33 SHOW BINLOG STATUS  
1.1.2.8.34 SHOW OPEN TABLES  
1.1.2.8.35 SHOW PACKAGE BODY STATUS  
1.1.2.8.36 SHOW PACKAGE STATUS  
1.1.2.8.37 SHOW PLUGINS  
1.1.2.8.38 SHOW PLUGINS SONAME  
1.1.2.8.39 SHOW PRIVILEGES  
1.1.2.8.40 SHOW PROCEDURE CODE  
1.1.2.8.41 SHOW PROCEDURE STATUS  
1.1.2.8.42 SHOW PROCESSLIST  
1.1.2.8.43 SHOW PROFILES  
1.1.2.8.44 SHOW QUERY\_RESPONSE\_TIME  
1.1.2.8.45 SHOW RELAYLOG EVENTS  
1.1.2.8.46 SHOW REPLICA HOSTS  
1.1.2.8.47 SHOW REPLICA STATUS  
1.1.2.8.48 SHOW STATUS  
1.1.2.8.49 SHOW TABLE STATUS  
1.1.2.8.50 SHOW TABLES  
1.1.2.8.51 SHOW TABLE\_STATISTICS  
1.1.2.8.52 SHOW TRIGGERS  
1.1.2.8.53 SHOW USER\_STATISTICS  
1.1.2.8.54 SHOW VARIABLES  
1.1.2.8.55 SHOW WARNINGS  
1.1.2.8.56 SHOW WSREP\_MEMBERSHIP  
1.1.2.8.57 SHOW WSREP\_STATUS

## 1.1.2.9 System Tables

### 1.1.2.9.1 Information Schema

#### 1.1.2.9.1.1 Information Schema Tables

1.1.2.9.1.1.1 Information Schema INNODB\_BUFFER\_PAGE Table  
1.1.2.9.1.1.1.2 Information Schema INNODB\_BUFFER\_PAGE\_LRU Table  
1.1.2.9.1.1.1.3 Information Schema INNODB\_BUFFER\_POOL\_PAGES Table  
1.1.2.9.1.1.1.4 Information Schema INNODB\_BUFFER\_POOL\_PAGES\_BLOB Table  
1.1.2.9.1.1.1.5 Information Schema INNODB\_BUFFER\_POOL\_PAGES\_INDEX Table  
1.1.2.9.1.1.1.6 Information Schema INNODB\_BUFFER\_POOL\_STATS Table

- 1.1.2.9.1.1.1.7 Information Schema INNODB\_CHANGED\_PAGES Table
- 1.1.2.9.1.1.1.8 Information Schema INNODB\_CMP and INNODB\_CMP\_RESET Tables
- 1.1.2.9.1.1.1.9 Information Schema INNODB\_CMPCMEM and INNODB\_CMPCMEM\_RESET Tables
- 1.1.2.9.1.1.1.10 Information Schema INNODB\_CMP\_PER\_INDEX and INNODB\_CMP\_PER\_INDEX\_RESET Tables
- 1.1.2.9.1.1.1.11 Information Schema INNODB\_FT\_BEING\_DELETED Table
- 1.1.2.9.1.1.1.12 Information Schema INNODB\_FT\_CONFIG Table
- 1.1.2.9.1.1.1.13 Information Schema INNODB\_FT\_DEFAULT\_STOPWORD Table
- 1.1.2.9.1.1.1.14 Information Schema INNODB\_FT\_DELETED Table
- 1.1.2.9.1.1.1.15 Information Schema INNODB\_FT\_INDEX\_CACHE Table
- 1.1.2.9.1.1.1.16 Information Schema INNODB\_FT\_INDEX\_TABLE Table
- 1.1.2.9.1.1.1.17 Information Schema INNODB\_LOCK\_WAITS Table
- 1.1.2.9.1.1.1.18 Information Schema INNODB\_LOCKS Table
- 1.1.2.9.1.1.1.19 Information Schema INNODB\_METRICS Table
- 1.1.2.9.1.1.1.20 Information Schema INNODB\_MUTEXES Table
- 1.1.2.9.1.1.1.21 Information Schema INNODB\_SYS\_COLUMNS Table
- 1.1.2.9.1.1.1.22 Information Schema INNODB\_SYS\_DATAFILES Table
- 1.1.2.9.1.1.1.23 Information Schema INNODB\_SYS\_FIELDS Table
- 1.1.2.9.1.1.1.24 Information Schema INNODB\_SYS\_FOREIGN Table
- 1.1.2.9.1.1.1.25 Information Schema INNODB\_SYS\_FOREIGN\_COLS Table
- 1.1.2.9.1.1.1.26 Information Schema INNODB\_SYS\_INDEXES Table
- 1.1.2.9.1.1.1.27 Information Schema INNODB\_SYS\_SEMAPHORE\_WAITS Table
- 1.1.2.9.1.1.1.28 Information Schema INNODB\_SYS\_TABLES Table
- 1.1.2.9.1.1.1.29 Information Schema INNODB\_SYS\_TABLESPACES Table
- 1.1.2.9.1.1.1.30 Information Schema INNODB\_SYS\_TABLESTATS Table
- 1.1.2.9.1.1.1.31 Information Schema INNODB\_SYS\_VIRTUAL Table
- 1.1.2.9.1.1.1.32 Information Schema INNODB\_TABLESPACES\_ENCRYPTION Table
- 1.1.2.9.1.1.1.33 Information Schema INNODB\_TABLESPACES\_SCRUBBING Table
- 1.1.2.9.1.1.1.34 Information Schema INNODB\_TRX Table
- 1.1.2.9.1.1.1.35 Information Schema TEMP\_TABLES\_INFO Table
- 1.1.2.9.1.1.2 Information Schema MyRocks Tables
  - 1.1.2.9.1.1.2.1 Information Schema ROCKSDB\_CFSTATS Table
  - 1.1.2.9.1.1.2.2 Information Schema ROCKSDB\_CF\_OPTIONS Table
  - 1.1.2.9.1.1.2.3 Information Schema ROCKSDB\_COMPACTION\_STATS Table
  - 1.1.2.9.1.1.2.4 Information Schema ROCKSDB\_DBSTATS Table
  - 1.1.2.9.1.1.2.5 Information Schema ROCKSDB\_DDL Table
  - 1.1.2.9.1.1.2.6 Information Schema ROCKSDB\_DEADLOCK Table
  - 1.1.2.9.1.1.2.7 Information Schema ROCKSDB\_GLOBAL\_INFO Table
  - 1.1.2.9.1.1.2.8 Information Schema ROCKSDB\_INDEX\_FILE\_MAP Table
  - 1.1.2.9.1.1.2.9 Information Schema ROCKSDB\_LOCKS Table
  - 1.1.2.9.1.1.2.10 Information Schema ROCKSDB\_PERF\_CONTEXT Table
  - 1.1.2.9.1.1.2.11 Information Schema ROCKSDB\_PERF\_CONTEXT\_GLOBAL Table
  - 1.1.2.9.1.1.2.12 Information Schema ROCKSDB\_SST\_PROPS Table
  - 1.1.2.9.1.1.2.13 Information Schema ROCKSDB\_TRX Table
- 1.1.2.9.1.1.3 ColumnStore Information Schema Tables
- 1.1.2.9.1.1.4 Information Schema ALL\_PLUGINS Table
- 1.1.2.9.1.1.5 Information Schema APPLICABLE\_ROLES Table
- 1.1.2.9.1.1.6 Information Schema CHARACTER\_SETS Table
- 1.1.2.9.1.1.7 Information Schema CHECK\_CONSTRAINTS Table
- 1.1.2.9.1.1.8 Information Schema CLIENT\_STATISTICS Table
- 1.1.2.9.1.1.9 Information Schema COLLATION\_CHARACTER\_SET\_APPLICABILITY Table
- 1.1.2.9.1.1.10 Information Schema COLLATIONS Table

1.1.2.9.1.1.11 Information Schema COLUMN\_PRIVILEGES Table  
1.1.2.9.1.1.12 Information Schema COLUMNS Table  
1.1.2.9.1.1.13 Information Schema DISKS Table  
1.1.2.9.1.1.14 Information Schema ENABLED\_ROLES Table  
1.1.2.9.1.1.15 Information Schema ENGINES Table  
1.1.2.9.1.1.16 Information Schema EVENTS Table  
1.1.2.9.1.1.17 Information Schema FEEDBACK Table  
1.1.2.9.1.1.18 Information Schema FILES Table  
1.1.2.9.1.1.19 Information Schema GEOMETRY\_COLUMNS Table  
1.1.2.9.1.1.20 Information Schema GLOBAL\_STATUS and SESSION\_STATUS Tables  
1.1.2.9.1.1.21 Information Schema GLOBAL\_VARIABLES and SESSION\_VARIABLES Tables  
1.1.2.9.1.1.22 Information Schema INDEX\_STATISTICS Table  
1.1.2.9.1.1.23 Information Schema KEY\_CACHES Table  
1.1.2.9.1.1.24 Information Schema KEY\_COLUMN\_USAGE Table  
1.1.2.9.1.1.25 Information Schema KEYWORDS Table  
1.1.2.9.1.1.26 Information Schema LOCALES Table  
1.1.2.9.1.1.27 Information Schema METADATA\_LOCK\_INFO Table  
1.1.2.9.1.1.28 Information Schema MROONGA\_STATS Table  
1.1.2.9.1.1.29 Information Schema OPTIMIZER\_TRACE Table  
1.1.2.9.1.1.30 Information Schema PARAMETERS Table  
1.1.2.9.1.1.31 Information Schema PARTITIONS Table  
1.1.2.9.1.1.32 Information Schema PLUGINS Table  
1.1.2.9.1.1.33 Information Schema PROCESSLIST Table  
1.1.2.9.1.1.34 Information Schema PROFILING Table  
1.1.2.9.1.1.35 Information Schema QUERY\_CACHE\_INFO Table  
1.1.2.9.1.1.36 Information Schema QUERY\_RESPONSE\_TIME Table  
1.1.2.9.1.1.37 Information Schema REFERENTIAL\_CONSTRAINTS Table  
1.1.2.9.1.1.38 Information Schema ROUTINES Table  
1.1.2.9.1.1.39 Information Schema SCHEMA\_PRIVILEGES Table  
1.1.2.9.1.1.40 Information Schema SCHEMATA Table  
1.1.2.9.1.1.41 Information Schema SPATIAL\_REF\_SYS Table  
1.1.2.9.1.1.42 Information Schema SPIDER\_ALLOC\_MEM Table  
1.1.2.9.1.1.43 Information Schema SPIDER\_WRAPPER\_PROTOCOLS Table  
1.1.2.9.1.1.44 Information Schema SQL\_FUNCTIONS Table  
1.1.2.9.1.1.45 Information Schema STATISTICS Table  
1.1.2.9.1.1.46 Information Schema SYSTEM\_VARIABLES Table  
1.1.2.9.1.1.47 Information Schema TABLE\_CONSTRAINTS Table  
1.1.2.9.1.1.48 Information Schema TABLE\_PRIVILEGES Table  
1.1.2.9.1.1.49 Information Schema TABLE\_STATISTICS Table  
1.1.2.9.1.1.50 Information Schema TABLES Table  
1.1.2.9.1.1.51 Information Schema TABLESPACES Table  
1.1.2.9.1.1.52 Information Schema THREAD\_POOL\_GROUPS Table  
1.1.2.9.1.1.53 Information Schema THREAD\_POOL\_STATS Table  
1.1.2.9.1.1.54 Information Schema THREAD\_POOL\_WAITS Table  
1.1.2.9.1.1.55 Information Schema TRIGGERS Table  
1.1.2.9.1.1.56 Information Schema USER\_PRIVILEGES Table  
1.1.2.9.1.1.57 Information Schema USER\_STATISTICS Table  
1.1.2.9.1.1.58 Information Schema USER\_VARIABLES Table  
1.1.2.9.1.1.59 Information Schema VIEWS Table

- 1.1.2.9.1.1.60 Information Schema WSREP\_MEMBERSHIP Table
- 1.1.2.9.1.1.61 Information Schema WSREP\_STATUS Table
- 1.1.2.9.1.2 Extended SHOW
- 1.1.2.9.1.3 TIME\_MS column in INFORMATION\_SCHEMA.PROCESSLIST
- 1.1.2.9.2 Performance Schema
  - 1.1.2.9.2.1 Performance Schema Tables
    - 1.1.2.9.2.1.1 List of Performance Schema Tables
    - 1.1.2.9.2.1.2 Performance Schema accounts Table
    - 1.1.2.9.2.1.3 Performance Schema cond\_instances Table
    - 1.1.2.9.2.1.4 Performance Schema events\_stages\_current Table
    - 1.1.2.9.2.1.5 Performance Schema events\_stages\_history Table
    - 1.1.2.9.2.1.6 Performance Schema events\_stages\_history\_long Table
    - 1.1.2.9.2.1.7 Performance Schema events\_stages\_summary\_by\_account\_by\_event\_name Table
    - 1.1.2.9.2.1.8 Performance Schema events\_stages\_summary\_by\_host\_by\_event\_name Table
    - 1.1.2.9.2.1.9 Performance Schema events\_stages\_summary\_by\_thread\_by\_event\_name Table
    - 1.1.2.9.2.1.10 Performance Schema events\_stages\_summary\_by\_user\_by\_event\_name Table
    - 1.1.2.9.2.1.11 Performance Schema events\_stages\_summary\_global\_by\_event\_name Table
    - 1.1.2.9.2.1.12 Performance Schema events\_statements\_history Table
    - 1.1.2.9.2.1.13 Performance Schema events\_statements\_history\_long Table
    - 1.1.2.9.2.1.14 Performance Schema events\_statements\_summary\_by\_account\_by\_event\_name Table
    - 1.1.2.9.2.1.15 Performance Schema events\_statements\_summary\_by\_digest Table
    - 1.1.2.9.2.1.16 Performance Schema events\_statements\_summary\_by\_host\_by\_event\_name Table
    - 1.1.2.9.2.1.17 Performance Schema events\_statements\_summary\_by\_program Table
    - 1.1.2.9.2.1.18 Performance Schema events\_statements\_summary\_by\_thread\_by\_event\_name Table
    - 1.1.2.9.2.1.19 Performance Schema events\_statements\_summary\_by\_user\_by\_event\_name Table
    - 1.1.2.9.2.1.20 Performance Schema events\_statements\_summary\_global\_by\_event\_name Table
    - 1.1.2.9.2.1.21 Performance Schema events\_transactions\_current Table
    - 1.1.2.9.2.1.22 Performance Schema events\_transactions\_history Table
    - 1.1.2.9.2.1.23 Performance Schema events\_transactions\_history\_long Table
    - 1.1.2.9.2.1.24 Performance Schema events\_transactions\_summary\_by\_account\_by\_event\_name Table
    - 1.1.2.9.2.1.25 Performance Schema events\_transactions\_summary\_by\_host\_by\_event\_name Table
    - 1.1.2.9.2.1.26 Performance Schema events\_transactions\_summary\_by\_thread\_by\_event\_name Table
    - 1.1.2.9.2.1.27 Performance Schema events\_transactions\_summary\_by\_user\_by\_event\_name Table
    - 1.1.2.9.2.1.28 Performance Schema events\_transactions\_summary\_global\_by\_event\_name Table
    - 1.1.2.9.2.1.29 Performance Schema events\_waits\_current Table
    - 1.1.2.9.2.1.30 Performance Schema events\_waits\_history Table
    - 1.1.2.9.2.1.31 Performance Schema events\_waits\_summary\_by\_account\_by\_event\_name Table

- 1.1.2.9.2.1.32 Performance Schema events\_waits\_summary\_by\_host\_by\_event\_name Table
  - 1.1.2.9.2.1.33 Performance Schema events\_waits\_summary\_by\_instance Table
  - 1.1.2.9.2.1.34 Performance Schema events\_waits\_summary\_by\_thread\_by\_event\_name Table
  - 1.1.2.9.2.1.35 Performance Schema events\_waits\_summary\_by\_user\_by\_event\_name Table
  - 1.1.2.9.2.1.36 Performance Schema events\_waits\_summary\_global\_by\_event\_name Table
  - 1.1.2.9.2.1.37 Performance Schema file\_instances Table
  - 1.1.2.9.2.1.38 Performance Schema file\_summary\_by\_event\_name Table
  - 1.1.2.9.2.1.39 Performance Schema file\_summary\_by\_instance Table
  - 1.1.2.9.2.1.40 Performance Schema global\_status Table
  - 1.1.2.9.2.1.41 Performance Schema hosts Table
  - 1.1.2.9.2.1.42 Performance Schema memory\_summary\_by\_account\_by\_event\_name Table
  - 1.1.2.9.2.1.43 Performance Schema memory\_summary\_by\_host\_by\_event\_name Table
  - 1.1.2.9.2.1.44 Performance Schema memory\_summary\_by\_thread\_by\_event\_name Table
  - 1.1.2.9.2.1.45 Performance Schema memory\_summary\_by\_user\_by\_event\_name Table
  - 1.1.2.9.2.1.46 Performance Schema memory\_summary\_global\_by\_event\_name Table
  - 1.1.2.9.2.1.47 Performance Schema metadata\_locks Table
  - 1.1.2.9.2.1.48 Performance Schema mutex\_instances Table
  - 1.1.2.9.2.1.49 Performance Schema objects\_summary\_global\_by\_type Table
  - 1.1.2.9.2.1.50 Performance Schema performance\_timers Table
  - 1.1.2.9.2.1.51 Performance Schema prepared\_statements\_instances Table
  - 1.1.2.9.2.1.52 Performance Schema replication\_applier\_configuration Table
  - 1.1.2.9.2.1.53 Performance Schema replication\_applier\_status Table
  - 1.1.2.9.2.1.54 Performance Schema replication\_applier\_status\_by\_coordinator Table
  - 1.1.2.9.2.1.55 Performance Schema replication\_applier\_status\_by\_worker Table
  - 1.1.2.9.2.1.56 Performance Schema replication\_connection\_configuration Table
  - 1.1.2.9.2.1.57 Performance Schema rwlock\_instances Table
  - 1.1.2.9.2.1.58 Performance Schema session\_account\_connect\_attrs Table
  - 1.1.2.9.2.1.59 Performance Schema session\_connect\_attrs Table
  - 1.1.2.9.2.1.60 Performance Schema session\_status Table
  - 1.1.2.9.2.1.61 Performance Schema setup\_actors Table
  - 1.1.2.9.2.1.62 Performance Schema setup\_consumers Table
  - 1.1.2.9.2.1.63 Performance Schema setup\_instruments Table
  - 1.1.2.9.2.1.64 Performance Schema setup\_objects Table
  - 1.1.2.9.2.1.65 Performance Schema setup\_timers Table
  - 1.1.2.9.2.1.66 Performance Schema socket\_instances Table
  - 1.1.2.9.2.1.67 Performance Schema socket\_summary\_by\_event\_name Table
  - 1.1.2.9.2.1.68 Performance Schema socket\_summary\_by\_instance Table
  - 1.1.2.9.2.1.69 Performance Schema status\_by\_thread Table
  - 1.1.2.9.2.1.70 Performance Schema table\_io\_waits\_summary\_by\_index\_usage Table
  - 1.1.2.9.2.1.71 Performance Schema table\_io\_waits\_summary\_by\_table Table
  - 1.1.2.9.2.1.72 Performance Schema table\_lock\_waits\_summary\_by\_table Table
  - 1.1.2.9.2.1.73 Performance Schema threads Table
  - 1.1.2.9.2.1.74 Performance Schema users Table
- ## 1.1.2.9.2.2 Performance Schema Overview
- ## 1.1.2.9.2.3 Performance Schema Status Variables
- ## 1.1.2.9.2.4 Performance Schema System Variables
- ## 1.1.2.9.2.5 Performance Schema Digests
- ## 1.1.2.9.2.6 PERFORMANCE\_SCHEMA Storage Engine

### 1.1.2.9.3 The mysql Database Tables

- 1.1.2.9.3.1 mysql.column\_stats Table
- 1.1.2.9.3.2 mysql.columns\_priv Table
- 1.1.2.9.3.3 mysql.db Table
- 1.1.2.9.3.4 mysql.event Table
- 1.1.2.9.3.5 mysql.func Table
- 1.1.2.9.3.6 mysql.general\_log Table
- 1.1.2.9.3.7 mysql.global\_priv Table
- 1.1.2.9.3.8 mysql.gtid\_slave\_pos Table
- 1.1.2.9.3.9 mysql.help\_category Table
- 1.1.2.9.3.10 mysql.help\_keyword Table
- 1.1.2.9.3.11 mysql.help\_relation Table
- 1.1.2.9.3.12 mysql.help\_topic Table
- 1.1.2.9.3.13 mysql.index\_stats Table
- 1.1.2.9.3.14 mysql.innodb\_index\_stats
- 1.1.2.9.3.15 mysql.innodb\_table\_stats
- 1.1.2.9.3.16 mysql.password\_reuse\_check\_history Table
- 1.1.2.9.3.17 mysql.plugin Table
- 1.1.2.9.3.18 mysql.proc Table
- 1.1.2.9.3.19 mysql.procs\_priv Table
- 1.1.2.9.3.20 mysql.roles\_mapping Table
- 1.1.2.9.3.21 mysql.servers Table
- 1.1.2.9.3.22 mysql.slow\_log Table
- 1.1.2.9.3.23 mysql.tables\_priv Table
- 1.1.2.9.3.24 mysql.table\_stats Table
- 1.1.2.9.3.25 mysql.time\_zone Table
- 1.1.2.9.3.26 mysql.time\_zone\_leap\_second Table
- 1.1.2.9.3.27 mysql.time\_zone\_name Table
- 1.1.2.9.3.28 mysql.time\_zone\_transition Table
- 1.1.2.9.3.29 mysql.time\_zone\_transition\_type Table
- 1.1.2.9.3.30 mysql.transaction\_registry Table
- 1.1.2.9.3.31 mysql.user Table
- 1.1.2.9.3.32 Spider mysql Database Tables
  - 1.1.2.9.3.32.1 mysql.spider\_link\_failed\_log Table
  - 1.1.2.9.3.32.2 mysql.spider\_link\_mon\_servers Table
  - 1.1.2.9.3.32.3 mysql.spider\_tables Table
  - 1.1.2.9.3.32.4 mysql.spider\_table\_crd Table
  - 1.1.2.9.3.32.5 mysql.spider\_table\_position\_for\_recovery Table
  - 1.1.2.9.3.32.6 mysql.spider\_table\_sts Table
  - 1.1.2.9.3.32.7 mysql.spider\_xa Table
  - 1.1.2.9.3.32.8 mysql.spider\_xa\_failed\_log Table
  - 1.1.2.9.3.32.9 mysql.spider\_xa\_member Table

### 1.1.2.9.4 Sys Schema

- 1.1.2.9.4.1 Sys Schema sys\_config Table

### 1.1.2.9.5 mariadb\_schema

### 1.1.2.9.6 Writing Logs Into Tables

- 1.1.2.10 BINLOG
  - 1.1.2.11 PURGE BINARY LOGS
  - 1.1.2.12 CACHE INDEX
  - 1.1.2.13 DESCRIBE
  - 1.1.2.14 EXECUTE Statement
  - 1.1.2.15 HELP Command
  - 1.1.2.16 KILL [CONNECTION | QUERY]
  - 1.1.2.17 LOAD INDEX
  - 1.1.2.18 RESET
  - 1.1.2.19 SHUTDOWN
  - 1.1.2.20 USE
- 1.1.3 Data Definition
- 1.1.3.1 CREATE
    - 1.1.3.1.1 CREATE DATABASE
    - 1.1.3.1.2 CREATE EVENT
    - 1.1.3.1.3 CREATE FUNCTION
    - 1.1.3.1.4 CREATE FUNCTION UDF
    - 1.1.3.1.5 CREATE INDEX
    - 1.1.3.1.6 CREATE LOGFILE GROUP
    - 1.1.3.1.7 CREATE PACKAGE
    - 1.1.3.1.8 CREATE PACKAGE BODY
    - 1.1.3.1.9 CREATE PROCEDURE
    - 1.1.3.1.10 CREATE ROLE
    - 1.1.3.1.11 CREATE SEQUENCE
    - 1.1.3.1.12 CREATE SERVER
    - 1.1.3.1.13 CREATE TABLE
    - 1.1.3.1.14 CREATE TABLESPACE
    - 1.1.3.1.15 CREATE TRIGGER
    - 1.1.3.1.16 CREATE USER
    - 1.1.3.1.17 CREATE VIEW
    - 1.1.3.1.18 Silent Column Changes
    - 1.1.3.1.19 Generated (Virtual and Persistent/Stored) Columns
    - 1.1.3.1.20 Invisible Columns
  - 1.1.3.2 ALTER
    - 1.1.3.2.1 ALTER TABLE
    - 1.1.3.2.2 ALTER DATABASE
    - 1.1.3.2.3 ALTER EVENT
    - 1.1.3.2.4 ALTER FUNCTION
    - 1.1.3.2.5 ALTER LOGFILE GROUP
    - 1.1.3.2.6 ALTER PROCEDURE
    - 1.1.3.2.7 ALTER SEQUENCE
    - 1.1.3.2.8 ALTER SERVER

- 1.1.3.2.9 ALTER TABLESPACE
- 1.1.3.2.10 ALTER USER
- 1.1.3.2.11 ALTER VIEW
- 1.1.3.3 DROP
  - 1.1.3.3.1 DROP DATABASE
  - 1.1.3.3.2 DROP EVENT
  - 1.1.3.3.3 DROP FUNCTION
  - 1.1.3.3.4 DROP FUNCTION UDF
  - 1.1.3.3.5 DROP INDEX
  - 1.1.3.3.6 DROP LOGFILE GROUP
  - 1.1.3.3.7 DROP PACKAGE
  - 1.1.3.3.8 DROP PACKAGE BODY
  - 1.1.3.3.9 DROP PROCEDURE
  - 1.1.3.3.10 DROP ROLE
  - 1.1.3.3.11 DROP SEQUENCE
  - 1.1.3.3.12 DROP SERVER
  - 1.1.3.3.13 DROP TABLE
  - 1.1.3.3.14 DROP TABLESPACE
  - 1.1.3.3.15 DROP TRIGGER
  - 1.1.3.3.16 DROP USER
  - 1.1.3.3.17 DROP VIEW
- 1.1.3.4
- 1.1.3.5 CONSTRAINT
- 1.1.3.6 MERGE
- 1.1.3.7 RENAME TABLE
- 1.1.3.8 TRUNCATE TABLE
- 1.1.4 Data Manipulation
  - 1.1.4.1 Selecting Data
    - 1.1.4.1.1 SELECT
    - 1.1.4.1.2 Joins and Subqueries
      - 1.1.4.1.2.1 Joins
        - 1.1.4.1.2.1.1 Joining Tables with JOIN Clauses
        - 1.1.4.1.2.1.2 More Advanced Joins
        - 1.1.4.1.2.1.3 JOIN Syntax
        - 1.1.4.1.2.1.4 Comma vs JOIN
      - 1.1.4.1.2.2 Subqueries
        - 1.1.4.1.2.2.1 Subqueries
        - 1.1.4.1.2.2.2 Scalar Subqueries
        - 1.1.4.1.2.2.3 Row Subqueries
        - 1.1.4.1.2.2.4 Subqueries and ALL
        - 1.1.4.1.2.2.5 Subqueries and ANY
        - 1.1.4.1.2.2.6 Subqueries and EXISTS
        - 1.1.4.1.2.2.7 Subqueries in a FROM Clause
        - 1.1.4.1.2.2.8 Subquery Optimizations

- 1.1.4.1.2.2.8.1 Subquery Optimizations Map
  - 1.1.4.1.2.2.8.2 Semi-join Subquery Optimizations
  - 1.1.4.1.2.2.8.3 Table Pullout Optimization
  - 1.1.4.1.2.2.8.4 Non-semi-join Subquery Optimizations
  - 1.1.4.1.2.2.8.5 Subquery Cache
  - 1.1.4.1.2.2.8.6 Condition Pushdown Into IN subqueries
  - 1.1.4.1.2.2.8.7 Conversion of Big IN Predicates Into Subqueries
  - 1.1.4.1.2.2.8.8 EXISTS-to-IN Optimization
  - 1.1.4.1.2.2.8.9 Optimizing GROUP BY and DISTINCT Clauses in Subqueries
  - 1.1.4.1.2.2.9 Subqueries and JOINS
  - 1.1.4.1.2.2.10 Subquery Limitations
  - 1.1.4.1.2.3 UNION
  - 1.1.4.1.2.4 EXCEPT
  - 1.1.4.1.2.5 INTERSECT
  - 1.1.4.1.2.6 Precedence Control in Table Operations
  - 1.1.4.1.2.7 MINUS
  - 1.1.4.1.3 LIMIT
  - 1.1.4.1.4 ORDER BY
  - 1.1.4.1.5 GROUP BY
  - 1.1.4.1.6 Common Table Expressions
    - 1.1.4.1.6.1 WITH
    - 1.1.4.1.6.2 Non-Recursive Common Table Expressions Overview
    - 1.1.4.1.6.3 Recursive Common Table Expressions Overview
  - 1.1.4.1.7 SELECT WITH ROLLUP
  - 1.1.4.1.8 SELECT INTO OUTFILE
  - 1.1.4.1.9 SELECT INTO DUMPFILE
  - 1.1.4.1.10 FOR UPDATE
  - 1.1.4.1.11 LOCK IN SHARE MODE
  - 1.1.4.1.12 Optimizer Hints
  - 1.1.4.1.13 PROCEDURE
  - 1.1.4.1.14 HANDLER
    - 1.1.4.1.14.1 HANDLER Commands
    - 1.1.4.1.14.2 HANDLER for MEMORY Tables
  - 1.1.4.1.15 DUAL
  - 1.1.4.1.16 SELECT ... OFFSET ... FETCH
- 1.1.4.2 Inserting and Loading Data
- 1.1.4.2.1 INSERT
  - 1.1.4.2.2 INSERT DELAYED
  - 1.1.4.2.3 INSERT SELECT
  - 1.1.4.2.4 LOAD Data into Tables or Index
    - 1.1.4.2.4.1 LOAD DATA INFILE
    - 1.1.4.2.4.2 LOAD INDEX
    - 1.1.4.2.4.3 LOAD XML
    - 1.1.4.2.4.4 LOAD\_FILE
  - 1.1.4.2.5 Concurrent Inserts
  - 1.1.4.2.6 HIGH\_PRIORITY and LOW\_PRIORITY

- 1.1.4.2.7 IGNORE
- 1.1.4.2.8 INSERT - Default & Duplicate Values
- 1.1.4.2.9 INSERT IGNORE
- 1.1.4.2.10 INSERT ON DUPLICATE KEY UPDATE
- 1.1.4.2.11 INSERT...RETURNING
- 1.1.4.3 Changing and Deleting Data
  - 1.1.4.3.1 DELETE
  - 1.1.4.3.2 HIGH\_PRIORITY and LOW\_PRIORITY
  - 1.1.4.3.3 IGNORE
  - 1.1.4.3.4 REPLACE
  - 1.1.4.3.5 REPLACE...RETURNING
  - 1.1.4.3.6 TRUNCATE TABLE
  - 1.1.4.3.7 UPDATE
- 1.1.5 Prepared Statements
  - 1.1.5.1 PREPARE Statement
  - 1.1.5.2 Out Parameters in PREPARE
  - 1.1.5.3 EXECUTE STATEMENT
  - 1.1.5.4 DEALLOCATE / DROP PREPARE
  - 1.1.5.5 EXECUTE IMMEDIATE
- 1.1.6 Programmatic and Compound Statements
  - 1.1.6.1 Using Compound Statements Outside of Stored Programs
  - 1.1.6.2 BEGIN END
  - 1.1.6.3 CASE Statement
  - 1.1.6.4 DECLARE CONDITION
  - 1.1.6.5 DECLARE HANDLER
  - 1.1.6.6 DECLARE Variable
  - 1.1.6.7 FOR
  - 1.1.6.8 GOTO
  - 1.1.6.9 IF
  - 1.1.6.10 ITERATE
  - 1.1.6.11 Labels
  - 1.1.6.12 LEAVE
  - 1.1.6.13 LOOP
  - 1.1.6.14 REPEAT LOOP
  - 1.1.6.15 RESIGNAL
  - 1.1.6.16 RETURN
  - 1.1.6.17 SELECT INTO
  - 1.1.6.18 SET Variable
  - 1.1.6.19 SIGNAL
  - 1.1.6.20 WHILE
- 1.1.7 Stored Routine Statements

- 1.1.7.1 CALL
- 1.1.7.2 DO
- 1.1.8 Table Statements
- 1.1.9 Transactions
  - 1.1.9.1 START TRANSACTION
  - 1.1.9.2 COMMIT
  - 1.1.9.3 ROLLBACK
  - 1.1.9.4 SET TRANSACTION
  - 1.1.9.5 LOCK TABLES
  - 1.1.9.6 SAVEPOINT
  - 1.1.9.7 Metadata Locking
  - 1.1.9.8 SQL statements That Cause an Implicit Commit
  - 1.1.9.9 Transaction Timeouts
  - 1.1.9.10 UNLOCK TABLES
  - 1.1.9.11 WAIT and NOWAIT
  - 1.1.9.12 XA Transactions
- 1.1.10 HELP Command
- 1.1.11 Comment Syntax
- 1.1.12 Built-in Functions
  - 1.1.12.1 Function and Operator Reference
    - 1.1.12.2.1.1 Regular Expressions Overview
    - 1.1.12.2.1.2 Perl Compatible Regular Expressions (PCRE) Documentation
    - 1.1.12.2.1.3 NOT REGEXP
    - 1.1.12.2.1.4 REGEXP
    - 1.1.12.2.1.5 REGEXP\_INSTR
    - 1.1.12.2.1.6 REGEXP\_REPLACE
    - 1.1.12.2.1.7 REGEXP\_SUBSTR
    - 1.1.12.2.1.8 RLIKE
    - 1.1.12.2.2.1 COLUMN\_ADD
    - 1.1.12.2.2.2 COLUMN\_CHECK
    - 1.1.12.2.2.3 COLUMN\_CREATE
    - 1.1.12.2.2.4 COLUMN\_DELETE
    - 1.1.12.2.2.5 COLUMN\_EXISTS
    - 1.1.12.2.2.6 COLUMN\_GET
    - 1.1.12.2.2.7 COLUMN\_JSON
  - 1.1.12.2.3 ASCII
  - 1.1.12.2.4 BIN
  - 1.1.12.2.5 BINARY Operator
  - 1.1.12.2.6 BIT\_LENGTH
  - 1.1.12.2.7 CAST
  - 1.1.12.2.8 CHAR Function
  - 1.1.12.2.9 CHAR\_LENGTH

1.1.12.2.10 CHARACTER\_LENGTH  
1.1.12.2.11 CHR  
1.1.12.2.12 CONCAT  
1.1.12.2.13 CONCAT\_WS  
1.1.12.2.14 CONVERT  
1.1.12.2.15 ELT  
1.1.12.2.16 EXPORT\_SET  
1.1.12.2.17 EXTRACTVALUE  
1.1.12.2.18 FIELD  
1.1.12.2.19 FIND\_IN\_SET  
1.1.12.2.20 FORMAT  
1.1.12.2.21 FROM\_BASE64  
1.1.12.2.22 HEX  
1.1.12.2.23 INSERT Function  
1.1.12.2.24 LENGTH  
1.1.12.2.25 LENGTHB  
1.1.12.2.26 LIKE  
1.1.12.2.27 LOAD\_FILE  
1.1.12.2.28 LOCATE  
1.1.12.2.29 LOWER  
1.1.12.2.30 LPAD  
1.1.12.2.31 LTRIM  
1.1.12.2.32 MAKE\_SET  
1.1.12.2.33 MATCH AGAINST  
1.1.12.2.34 Full-Text Index Stopwords  
1.1.12.2.35 MID  
1.1.12.2.36 NOT LIKE  
1.1.12.2.37 NOT REGEXP  
1.1.12.2.38 OCTET\_LENGTH  
1.1.12.2.39 ORD  
1.1.12.2.40 POSITION  
1.1.12.2.41 QUOTE  
1.1.12.2.42 QUOTE\_IDENTIFIER  
1.1.12.2.43 REPEAT Function  
1.1.12.2.44 REPLACE Function  
1.1.12.2.45 REVERSE  
1.1.12.2.46 RIGHT  
1.1.12.2.47 RPAD  
1.1.12.2.48 RTRIM  
1.1.12.2.49 SFORMAT  
1.1.12.2.50 SOUNDEX  
1.1.12.2.51 SOUNDS LIKE  
1.1.12.2.52 SPACE

- 1.1.12.2.53 STRCMP
  - 1.1.12.2.54 SUBSTR
  - 1.1.12.2.55 SUBSTRING
  - 1.1.12.2.56 SUBSTRING\_INDEX
  - 1.1.12.2.57 TO\_BASE64
  - 1.1.12.2.58 TO\_CHAR
  - 1.1.12.2.59 TRIM
  - 1.1.12.2.60 TRIM\_ORACLE
  - 1.1.12.2.61 UCASE
  - 1.1.12.2.62 UNCOMPRESS
  - 1.1.12.2.63 UNCOMPRESSED\_LENGTH
  - 1.1.12.2.64 UNHEX
  - 1.1.12.2.65 UPDATEXML
  - 1.1.12.2.66 UPPER
  - 1.1.12.2.67 WEIGHT\_STRING
  - 1.1.12.2.68 Type Conversion
- 1.1.12.3 Date and Time Functions
- 1.1.12.3.1 Microseconds in MariaDB
  - 1.1.12.3.2 Date and Time Units
  - 1.1.12.3.3 ADD\_MONTHS
  - 1.1.12.3.4 ADDDATE
  - 1.1.12.3.5 ADDTIME
  - 1.1.12.3.6 CONVERT\_TZ
  - 1.1.12.3.7 CURDATE
  - 1.1.12.3.8 CURRENT\_DATE
  - 1.1.12.3.9 CURRENT\_TIME
  - 1.1.12.3.10 CURRENT\_TIMESTAMP
  - 1.1.12.3.11 CURTIME
  - 1.1.12.3.12 DATE FUNCTION
  - 1.1.12.3.13 DATEDIFF
  - 1.1.12.3.14 DATE\_ADD
  - 1.1.12.3.15 DATE\_FORMAT
  - 1.1.12.3.16 DATE\_SUB
  - 1.1.12.3.17 DAY
  - 1.1.12.3.18 DAYNAME
  - 1.1.12.3.19 DAYOFMONTH
  - 1.1.12.3.20 DAYOFWEEK
  - 1.1.12.3.21 DAYOFYEAR
  - 1.1.12.3.22 EXTRACT
  - 1.1.12.3.23 FROM\_DAYS
  - 1.1.12.3.24 FROM\_UNIXTIME
  - 1.1.12.3.25 GET\_FORMAT
  - 1.1.12.3.26 HOUR

- 1.1.12.3.27 LAST\_DAY
  - 1.1.12.3.28 LOCALTIME
  - 1.1.12.3.29 LOCALTIMESTAMP
  - 1.1.12.3.30 MAKEDATE
  - 1.1.12.3.31 MAKETIME
  - 1.1.12.3.32 MICROSECOND
  - 1.1.12.3.33 MINUTE
  - 1.1.12.3.34 MONTH
  - 1.1.12.3.35 MONTHNAME
  - 1.1.12.3.36 NOW
  - 1.1.12.3.37 PERIOD\_ADD
  - 1.1.12.3.38 PERIOD\_DIFF
  - 1.1.12.3.39 QUARTER
  - 1.1.12.3.40 SECOND
  - 1.1.12.3.41 SEC\_TO\_TIME
  - 1.1.12.3.42 STR\_TO\_DATE
  - 1.1.12.3.43 SUBDATE
  - 1.1.12.3.44 SUBTIME
  - 1.1.12.3.45 SYSDATE
  - 1.1.12.3.46 TIME Function
  - 1.1.12.3.47 TIMEDIFF
  - 1.1.12.3.48 TIMESTAMP FUNCTION
  - 1.1.12.3.49 TIMESTAMPADD
  - 1.1.12.3.50 TIMESTAMPDIFF
  - 1.1.12.3.51 TIME\_FORMAT
  - 1.1.12.3.52 TIME\_TO\_SEC
  - 1.1.12.3.53 TO\_DAYS
  - 1.1.12.3.54 TO\_SECONDS
  - 1.1.12.3.55 UNIX\_TIMESTAMP
  - 1.1.12.3.56 UTC\_DATE
  - 1.1.12.3.57 UTC\_TIME
  - 1.1.12.3.58 UTC\_TIMESTAMP
  - 1.1.12.3.59 WEEK
  - 1.1.12.3.60 WEEKDAY
  - 1.1.12.3.61 WEEKOFYEAR
  - 1.1.12.3.62 YEAR
  - 1.1.12.3.63 YEARWEEK
- 1.1.12.4 Aggregate Functions
- 1.1.12.4.1 Stored Aggregate Functions
  - 1.1.12.4.2 AVG
  - 1.1.12.4.3 BIT\_AND
  - 1.1.12.4.4 BIT\_OR
  - 1.1.12.4.5 BIT\_XOR

- 1.1.12.4.6 COUNT
  - 1.1.12.4.7 COUNT DISTINCT
  - 1.1.12.4.8 GROUP\_CONCAT
  - 1.1.12.4.9 JSON\_ARRAYAGG
  - 1.1.12.4.10 JSON\_OBJECTAGG
  - 1.1.12.4.11 MAX
  - 1.1.12.4.12 MIN
  - 1.1.12.4.13 STD
  - 1.1.12.4.14 STDDEV
  - 1.1.12.4.15 STDDEV\_POP
  - 1.1.12.4.16 STDDEV\_SAMP
  - 1.1.12.4.17 SUM
  - 1.1.12.4.18 VARIANCE
  - 1.1.12.4.19 VAR\_POP
  - 1.1.12.4.20 VAR\_SAMP
- 1.1.12.5 Numeric Functions
- 1.1.12.5.1 Addition Operator (+)
  - 1.1.12.5.2 Subtraction Operator (-)
  - 1.1.12.5.3 Division Operator (/)
  - 1.1.12.5.4 Multiplication Operator (\*)
  - 1.1.12.5.5 Modulo Operator (%)
  - 1.1.12.5.6 DIV
  - 1.1.12.5.7 ABS
  - 1.1.12.5.8 ACOS
  - 1.1.12.5.9 ASIN
  - 1.1.12.5.10 ATAN
  - 1.1.12.5.11 ATAN2
  - 1.1.12.5.12 CEIL
  - 1.1.12.5.13 CEILING
  - 1.1.12.5.14 CONV
  - 1.1.12.5.15 COS
  - 1.1.12.5.16 COT
  - 1.1.12.5.17 CRC32
  - 1.1.12.5.18 CRC32C
  - 1.1.12.5.19 DEGREES
  - 1.1.12.5.20 EXP
  - 1.1.12.5.21 FLOOR
  - 1.1.12.5.22 GREATEST
  - 1.1.12.5.23 LEAST
  - 1.1.12.5.24 LN
  - 1.1.12.5.25 LOG
  - 1.1.12.5.26 LOG10
  - 1.1.12.5.27 LOG2

- 1.1.12.5.28 MOD
- 1.1.12.5.29 OCT
- 1.1.12.5.30 PI
- 1.1.12.5.31 POW
- 1.1.12.5.32 POWER
- 1.1.12.5.33 RADIANS
- 1.1.12.5.34 RAND
- 1.1.12.5.35 ROUND
- 1.1.12.5.36 SIGN
- 1.1.12.5.37 SIN
- 1.1.12.5.38 SQRT
- 1.1.12.5.39 TAN
- 1.1.12.5.40 TRUNCATE
- 1.1.12.6 Control Flow Functions
  - 1.1.12.6.1 CASE OPERATOR
  - 1.1.12.6.2 DECODE
  - 1.1.12.6.3 DECODE\_ORACLE
  - 1.1.12.6.4 IF Function
  - 1.1.12.6.5 IFNULL
  - 1.1.12.6.6 NULLIF
  - 1.1.12.6.7 NVL
  - 1.1.12.6.8 NVL2
- 1.1.12.7 Pseudo Columns
  - 1.1.12.7.1 \_rowid
- 1.1.12.8 Secondary Functions
  - 1.1.12.8.1 Bit Functions and Operators
    - 1.1.12.8.1.1 Operator Precedence
    - 1.1.12.8.1.2 &
    - 1.1.12.8.1.3 <<
    - 1.1.12.8.1.4 >>
    - 1.1.12.8.1.5 BIT\_COUNT
    - 1.1.12.8.1.6 ^
    - 1.1.12.8.1.7 |
    - 1.1.12.8.1.8 ~
    - 1.1.12.8.1.9 Parentheses
    - 1.1.12.8.1.10 TRUE FALSE
  - 1.1.12.8.2 Encryption, Hashing and Compression Functions
    - 1.1.12.8.2.1 AES\_DECRYPT
    - 1.1.12.8.2.2 AES\_ENCRYPT
    - 1.1.12.8.2.3 COMPRESS
    - 1.1.12.8.2.4 DECODE
    - 1.1.12.8.2.5 DES\_DECRYPT
    - 1.1.12.8.2.6 DES\_ENCRYPT
    - 1.1.12.8.2.7 ENCODE

- 1.1.12.8.2.8 ENCRYPT
- 1.1.12.8.2.9 MD5
- 1.1.12.8.2.10 PASSWORD
- 1.1.12.8.2.11 PASSWORD
- 1.1.12.8.2.12 SHA1
- 1.1.12.8.2.13 SHA2
- 1.1.12.8.2.14 UNCOMPRESS
- 1.1.12.8.2.15 UNCOMPRESSED\_LENGTH
- 1.1.12.8.3 Information Functions
  - 1.1.12.8.3.1 BENCHMARK
  - 1.1.12.8.3.2 BINLOG\_GTID\_POS
  - 1.1.12.8.3.3 CHARSET
  - 1.1.12.8.3.4 COERCIBILITY
  - 1.1.12.8.3.5 COLLATION
  - 1.1.12.8.3.6 CONNECTION\_ID
  - 1.1.12.8.3.7 CURRENT\_ROLE
  - 1.1.12.8.3.8 CURRENT\_USER
  - 1.1.12.8.3.9 DATABASE
  - 1.1.12.8.3.10 DECODE\_HISTOGRAM
  - 1.1.12.8.3.11 DEFAULT
  - 1.1.12.8.3.12 FOUND\_ROWS
  - 1.1.12.8.3.13 LAST\_INSERT\_ID
  - 1.1.12.8.3.14 LAST\_VALUE
  - 1.1.12.8.3.15 PROCEDURE ANALYSE
  - 1.1.12.8.3.16 ROWNUM
  - 1.1.12.8.3.17 ROW\_COUNT
  - 1.1.12.8.3.18 SCHEMA
  - 1.1.12.8.3.19 SESSION\_USER
  - 1.1.12.8.3.20 SYSTEM\_USER
  - 1.1.12.8.3.21 USER
  - 1.1.12.8.3.22 VERSION
- 1.1.12.8.4 Miscellaneous Functions
  - 1.1.12.8.4.1 GET\_LOCK
  - 1.1.12.8.4.2 INET6\_ATON
  - 1.1.12.8.4.3 INET6\_NTOA
  - 1.1.12.8.4.4 INET\_ATON
  - 1.1.12.8.4.5 INET\_NTOA
  - 1.1.12.8.4.6 IS\_FREE\_LOCK
  - 1.1.12.8.4.7 IS\_IPV4
  - 1.1.12.8.4.8 IS\_IPV4\_COMPAT
  - 1.1.12.8.4.9 IS\_IPV4\_MAPPED
  - 1.1.12.8.4.10 IS\_IPV6
  - 1.1.12.8.4.11 IS\_USED\_LOCK
  - 1.1.12.8.4.12 MASTER\_GTID\_WAIT
  - 1.1.12.8.4.13 MASTER\_POS\_WAIT
  - 1.1.12.8.4.14 NAME\_CONST

- 1.1.12.8.4.15
- 1.1.12.8.4.16 RELEASE\_LOCK
- 1.1.12.8.4.17 SLEEP
- 1.1.12.8.4.18 SYS\_GUID
- 1.1.12.8.4.19 UUID
- 1.1.12.8.4.20 UUID\_SHORT
- 1.1.12.8.4.21 VALUES / VALUE

## 1.1.12.9 Special Functions

- 1.1.12.9.1 Dynamic Columns Functions
- 1.1.12.9.2 Galera Functions
  - 1.1.12.9.2.1 WSREP\_LAST\_SEEN\_GTID
  - 1.1.12.9.2.2 WSREP\_LAST\_WRITTEN\_GTID
  - 1.1.12.9.2.3 WSREP\_SYNC\_WAIT\_UPTO\_GTID
- 1.1.12.9.3 Geographic Functions
  - 1.1.12.9.3.1 Geometry Constructors
    - 1.1.12.9.3.1.1 BUFFER
    - 1.1.12.9.3.1.2 CONVEXHULL
    - 1.1.12.9.3.1.3 GEOMETRYCOLLECTION
    - 1.1.12.9.3.1.4 LINESTRING
    - 1.1.12.9.3.1.5 MULTILINESTRING
    - 1.1.12.9.3.1.6 MULTIPOINT
    - 1.1.12.9.3.1.7 MULTIPOLYGON
    - 1.1.12.9.3.1.8 POINT
    - 1.1.12.9.3.1.9 PointOnSurface
    - 1.1.12.9.3.1.10 POLYGON
    - 1.1.12.9.3.1.11 ST\_BUFFER
    - 1.1.12.9.3.1.12 ST\_CONVEXHULL
    - 1.1.12.9.3.1.13 ST\_INTERSECTION
    - 1.1.12.9.3.1.14 ST\_POINTONSURFACE
    - 1.1.12.9.3.1.15 ST\_SYMDIFFERENCE
    - 1.1.12.9.3.1.16 ST\_UNION
  - 1.1.12.9.3.2 Geometry Properties
    - 1.1.12.9.3.2.1 BOUNDARY
    - 1.1.12.9.3.2.2 DIMENSION
    - 1.1.12.9.3.2.3 ENVELOPE
    - 1.1.12.9.3.2.4 GeometryN
    - 1.1.12.9.3.2.5 GeometryType
    - 1.1.12.9.3.2.6 IsClosed
    - 1.1.12.9.3.2.7 IsEmpty
    - 1.1.12.9.3.2.8 IsRing
    - 1.1.12.9.3.2.9 IsSimple
    - 1.1.12.9.3.2.10 NumGeometries
    - 1.1.12.9.3.2.11 SRID
    - 1.1.12.9.3.2.12 ST\_BOUNDARY
    - 1.1.12.9.3.2.13 ST\_DIMENSION
    - 1.1.12.9.3.2.14 ST\_ENVELOPE
    - 1.1.12.9.3.2.15 ST\_GEOMETRYN
    - 1.1.12.9.3.2.16 ST\_GEOMETRYTYPE

- 1.1.12.9.3.2.17 ST\_ISCLOSED
- 1.1.12.9.3.2.18 ST\_ISEMPTY
- 1.1.12.9.3.2.19 ST\_IsRing
- 1.1.12.9.3.2.20 ST\_IsSimple
- 1.1.12.9.3.2.21 ST\_NUMGEOMETRIES
- 1.1.12.9.3.2.22 ST\_RELATE
- 1.1.12.9.3.2.23 ST\_SRID
- 1.1.12.9.3.3 Geometry Relations
  - 1.1.12.9.3.3.1 CONTAINS
  - 1.1.12.9.3.3.2 CROSSES
  - 1.1.12.9.3.3.3 DISJOINT
  - 1.1.12.9.3.3.4 EQUALS
  - 1.1.12.9.3.3.5 INTERSECTS
  - 1.1.12.9.3.3.6 OVERLAPS
  - 1.1.12.9.3.3.7 ST\_CONTAINS
  - 1.1.12.9.3.3.8 ST\_CROSSES
  - 1.1.12.9.3.3.9 ST\_DIFFERENCE
  - 1.1.12.9.3.3.10 ST\_DISJOINT
  - 1.1.12.9.3.3.11 ST\_DISTANCE
  - 1.1.12.9.3.3.12 ST\_DISTANCE\_SPHERE
  - 1.1.12.9.3.3.13 ST\_EQUALS
  - 1.1.12.9.3.3.14 ST\_INTERSECTS
  - 1.1.12.9.3.3.15 ST\_LENGTH
  - 1.1.12.9.3.3.16 ST\_OVERLAPS
  - 1.1.12.9.3.3.17 ST\_TOUCHES
  - 1.1.12.9.3.3.18 ST\_WITHIN
  - 1.1.12.9.3.3.19 TOUCHES
  - 1.1.12.9.3.3.20 WITHIN
- 1.1.12.9.3.4 LineString Properties
  - 1.1.12.9.3.4.1 ENDPOINT
  - 1.1.12.9.3.4.2 GLENGTH
  - 1.1.12.9.3.4.3 NumPoints
  - 1.1.12.9.3.4.4 PointN
  - 1.1.12.9.3.4.5 STARTPOINT
  - 1.1.12.9.3.4.6 ST\_ENDPOINT
  - 1.1.12.9.3.4.7 ST\_NUMPOINTS
  - 1.1.12.9.3.4.8 ST\_POINTN
  - 1.1.12.9.3.4.9 ST\_STARTPOINT
- 1.1.12.9.3.5 MBR (Minimum Bounding Rectangle)
  - 1.1.12.9.3.5.1 MBR Definition
  - 1.1.12.9.3.5.2 MBRCContains
  - 1.1.12.9.3.5.3 MBRDisjoint
  - 1.1.12.9.3.5.4 MBREqual
  - 1.1.12.9.3.5.5 MBRIIntersects
  - 1.1.12.9.3.5.6 MBROverlaps
  - 1.1.12.9.3.5.7 MBRTouches
  - 1.1.12.9.3.5.8 MBRWithin
- 1.1.12.9.3.6 Point Properties
  - 1.1.12.9.3.6.1 ST\_X

1.1.12.9.3.6.2 ST\_Y

1.1.12.9.3.6.3 X

1.1.12.9.3.6.4 Y

### 1.1.12.9.3.7 Polygon Properties

1.1.12.9.3.7.1 AREA

1.1.12.9.3.7.2 CENTROID

1.1.12.9.3.7.3 ExteriorRing

1.1.12.9.3.7.4 InteriorRingN

1.1.12.9.3.7.5 NumInteriorRings

1.1.12.9.3.7.6 ST\_AREA

1.1.12.9.3.7.7 ST\_CENTROID

1.1.12.9.3.7.8 ST\_ExteriorRing

1.1.12.9.3.7.9 ST\_InteriorRingN

1.1.12.9.3.7.10 ST\_NumInteriorRings

### 1.1.12.9.3.8 WKB

1.1.12.9.3.8.1 Well-Known Binary (WKB) Format

1.1.12.9.3.8.2 AsBinary

1.1.12.9.3.8.3 AsWKB

1.1.12.9.3.8.4 MLineFromWKB

1.1.12.9.3.8.5 MPointFromWKB

1.1.12.9.3.8.6 MPolyFromWKB

1.1.12.9.3.8.7 GeomCollFromWKB

1.1.12.9.3.8.8 GeometryCollectionFromWKB

1.1.12.9.3.8.9 GeometryFromWKB

1.1.12.9.3.8.10 GeomFromWKB

1.1.12.9.3.8.11 LineFromWKB

1.1.12.9.3.8.12 LineStringFromWKB

1.1.12.9.3.8.13 MultiLineStringFromWKB

1.1.12.9.3.8.14 MultiPointFromWKB

1.1.12.9.3.8.15 MultiPolygonFromWKB

1.1.12.9.3.8.16 PointFromWKB

1.1.12.9.3.8.17 PolyFromWKB

1.1.12.9.3.8.18 PolygonFromWKB

1.1.12.9.3.8.19 ST\_AsBinary

1.1.12.9.3.8.20 ST\_AsWKB

1.1.12.9.3.8.21 ST\_GeomCollFromWKB

1.1.12.9.3.8.22 ST\_GeometryCollectionFromWKB

1.1.12.9.3.8.23 ST\_GeometryFromWKB

1.1.12.9.3.8.24 ST\_GeomFromWKB

1.1.12.9.3.8.25 ST\_LineFromWKB

1.1.12.9.3.8.26 ST\_LineStringFromWKB

1.1.12.9.3.8.27 ST\_PointFromWKB

1.1.12.9.3.8.28 ST\_PolyFromWKB

1.1.12.9.3.8.29 ST\_PolyFromWKB

### 1.1.12.9.3.9 WKT

1.1.12.9.3.9.1 WKT Definition

1.1.12.9.3.9.2 AsText

1.1.12.9.3.9.3 AsWKT

1.1.12.9.3.9.4 GeomCollFromText

- 1.1.12.9.3.9.5 GeometryCollectionFromText
- 1.1.12.9.3.9.6 GeometryFromText
- 1.1.12.9.3.9.7 GeomFromText
- 1.1.12.9.3.9.8 LineFromText
- 1.1.12.9.3.9.9 LineStringFromText
- 1.1.12.9.3.9.10 MLineFromText
- 1.1.12.9.3.9.11 MPointFromText
- 1.1.12.9.3.9.12 MPolyFromText
- 1.1.12.9.3.9.13 MultiLineStringFromText
- 1.1.12.9.3.9.14 MultiPointFromText
- 1.1.12.9.3.9.15 MultiPolygonFromText
- 1.1.12.9.3.9.16 PointFromText
- 1.1.12.9.3.9.17 PolyFromText
- 1.1.12.9.3.9.18 PolygonFromText
- 1.1.12.9.3.9.19 ST\_AsText
- 1.1.12.9.3.9.20 ST\_ASWKT
- 1.1.12.9.3.9.21 ST\_GeomCollFromText
- 1.1.12.9.3.9.22 ST\_GeometryCollectionFromText
- 1.1.12.9.3.9.23 ST\_GeometryFromText
- 1.1.12.9.3.9.24 ST\_GeomFromText
- 1.1.12.9.3.9.25 ST\_LineFromText
- 1.1.12.9.3.9.26 ST\_LineStringFromText
- 1.1.12.9.3.9.27 ST\_PointFromText
- 1.1.12.9.3.9.28 ST\_PolyFromText
- 1.1.12.9.3.9.29 ST\_PolygonFromText

#### 1.1.12.9.4 JSON Functions

- 1.1.12.9.4.1 Differences between JSON\_QUERY and JSON\_VALUE
- 1.1.12.9.4.2 JSONPath Expressions
- 1.1.12.9.4.3 JSON\_ARRAY
- 1.1.12.9.4.4 JSON\_ARRAYAGG
- 1.1.12.9.4.5 JSON\_ARRAY\_APPEND
- 1.1.12.9.4.6 JSON\_ARRAY\_INSERT
- 1.1.12.9.4.7 JSON\_COMPACT
- 1.1.12.9.4.8 JSON\_CONTAINS
- 1.1.12.9.4.9 JSON\_CONTAINS\_PATH
- 1.1.12.9.4.10 JSON\_DEPTH
- 1.1.12.9.4.11 JSON\_DETAILED
- 1.1.12.9.4.12 JSON\_EQUALS
- 1.1.12.9.4.13 JSON\_EXISTS
- 1.1.12.9.4.14 JSON\_EXTRACT
- 1.1.12.9.4.15 JSON\_INSERT
- 1.1.12.9.4.16 JSON\_KEYS
- 1.1.12.9.4.17 JSON\_LENGTH
- 1.1.12.9.4.18 JSON\_LOOSE
- 1.1.12.9.4.19 JSON\_MERGE
- 1.1.12.9.4.20 JSON\_MERGE\_PATCH
- 1.1.12.9.4.21 JSON\_MERGE PRESERVE
- 1.1.12.9.4.22 JSON\_NORMALIZE

- 1.1.12.9.4.23 JSON\_OBJECT
  - 1.1.12.9.4.24 JSON\_OBJECTAGG
  - 1.1.12.9.4.25 JSON\_OVERLAPS
  - 1.1.12.9.4.26 JSON\_QUERY
  - 1.1.12.9.4.27 JSON\_QUOTE
  - 1.1.12.9.4.28 JSON\_REMOVE
  - 1.1.12.9.4.29 JSON\_REPLACE
  - 1.1.12.9.4.30 JSON\_SEARCH
  - 1.1.12.9.4.31 JSON\_SET
  - 1.1.12.9.4.32 JSON\_TABLE
  - 1.1.12.9.4.33 JSON\_TYPE
  - 1.1.12.9.4.34 JSON\_UNQUOTE
  - 1.1.12.9.4.35 JSON\_VALID
  - 1.1.12.9.4.36 JSON\_VALUE
- 1.1.12.9.5 Spider Functions
    - 1.1.12.9.5.1 SPIDER\_BG\_DIRECT\_SQL
    - 1.1.12.9.5.2 SPIDER\_COPY\_TABLES
    - 1.1.12.9.5.3 SPIDER\_DIRECT\_SQL
    - 1.1.12.9.5.4 SPIDER\_FLUSH\_TABLE\_MON\_CACHE
  - 1.1.12.9.6 Window Functions
    - 1.1.12.9.6.1 Window Functions Overview
    - 1.1.12.9.6.2 AVG
    - 1.1.12.9.6.3 BIT\_AND
    - 1.1.12.9.6.4 BIT\_OR
    - 1.1.12.9.6.5 BIT\_XOR
    - 1.1.12.9.6.6 COUNT
    - 1.1.12.9.6.7 COUNT DISTINCT
    - 1.1.12.9.6.8 GROUP\_CONCAT
    - 1.1.12.9.6.9 JSON\_ARRAYAGG
    - 1.1.12.9.6.10 JSON\_OBJECTAGG
    - 1.1.12.9.6.11 MAX
    - 1.1.12.9.6.12 MIN
    - 1.1.12.9.6.13 STD
    - 1.1.12.9.6.14 STDDEV
    - 1.1.12.9.6.15 STDDEV\_POP
    - 1.1.12.9.6.16 STDDEV\_SAMP
    - 1.1.12.9.6.17 SUM
    - 1.1.12.9.6.18 VARIANCE
    - 1.1.12.9.6.19 VAR\_POP
    - 1.1.12.9.6.20 VAR\_SAMP

## 2 MariaDB Administration

- 2.1 Getting, Installing and Upgrading MariaDB
  - 2.1.1 Where to Download MariaDB
  - 2.1.2 MariaDB Binary Packages

## 2.1.2.1 Installing MariaDB RPM Files

- 2.1.2.1.1 About the MariaDB RPM Files
- 2.1.2.1.2 Installing MariaDB with yum/dnf
- 2.1.2.1.3 Installing MariaDB with zypper
- 2.1.2.1.4 Installing MariaDB With the rpm Tool
- 2.1.2.1.5 Checking MariaDB RPM Package Signatures
- 2.1.2.1.6 Troubleshooting MariaDB Installs on Red Hat/CentOS
- 2.1.2.1.7 MariaDB for DirectAdmin Using RPMs
- 2.1.2.1.8 MariaDB Installation (Version 10.1.21) via RPMs on CentOS 7
- 2.1.2.1.9 Why Source RPMs (SRPMs) Aren't Packaged For Some Platforms
- 2.1.2.1.10 Building MariaDB from a Source RPM

## 2.1.2.2 Installing MariaDB .deb Files

- 2.1.2.3 Installing MariaDB MSI Packages on Windows
- 2.1.2.4 Installing MariaDB Server PKG packages on macOS
- 2.1.2.5 Installing MariaDB Binary Tarballs
- 2.1.2.6 Installing MariaDB Server on macOS Using Homebrew
- 2.1.2.7 Installing MariaDB Windows ZIP Packages

## 2.1.2.8

- 2.1.2.8.1 Get, Build and Test Latest MariaDB the Lazy Way
- 2.1.2.8.2 MariaDB Source Code
- 2.1.2.8.3 Build Environment Setup for Linux
- 2.1.2.8.4 Generic Build Instructions
- 2.1.2.8.5 Compiling MariaDB with Extra Modules/Options
  - 2.1.2.8.5.1 Compiling MariaDB with TCMalloc
  - 2.1.2.8.5.2 Compiling MariaDB with Vanilla XtraDB
  - 2.1.2.8.5.3 Specifying Which Plugins to Build
- 2.1.2.8.6 Creating the MariaDB Source Tarball
- 2.1.2.8.7 Creating the MariaDB Binary Tarball
- 2.1.2.8.8 Build Environment Setup for Mac
- 2.1.2.8.9 Building MariaDB From a Source RPM
- 2.1.2.8.10 Building MariaDB on CentOS
- 2.1.2.8.11 Building MariaDB on Fedora
- 2.1.2.8.12 Building MariaDB on Debian
- 2.1.2.8.13 Building MariaDB on FreeBSD
- 2.1.2.8.14 Building MariaDB on Gentoo
- 2.1.2.8.15 Building MariaDB on Solaris and OpenSolaris
- 2.1.2.8.16 Building MariaDB on Ubuntu
- 2.1.2.8.17 Building MariaDB on Windows
- 2.1.2.8.18 Installing MariaDB Server on macOS Using Homebrew
- 2.1.2.8.19 Creating a Debian Repository
- 2.1.2.8.20 Building MariaDB From Source Using musl-based GNU/Linux
- 2.1.2.8.21 Compiling MariaDB for Debugging
- 2.1.2.8.22 Cross-compiling MariaDB

- 2.1.2.8.23 MariaDB Source Configuration Options
- 2.1.2.8.24 Building RPM Packages From Source
- 2.1.2.8.25 Compile and Using MariaDB with AddressSanitizer (ASAN)
- 2.1.2.9 Distributions Which Include MariaDB
- 2.1.2.10 Running Multiple MariaDB Server Processes
- 2.1.2.11 Installing MariaDB Alongside MySQL
- 2.1.2.12 GPG
- 2.1.2.13 MariaDB Deprecation Policy
- 2.1.2.14 Automated MariaDB Deployment and Administration
  - 2.1.2.14.1 Why to Automate MariaDB Deployments and Management
  - 2.1.2.14.2 A Comparison Between Automation Systems
  - 2.1.2.14.3 Ansible and MariaDB
    - 2.1.2.14.3.1 Ansible Overview for MariaDB Users
    - 2.1.2.14.3.2 Deploying to Remote Servers with Ansible
    - 2.1.2.14.3.3 Deploying Docker Containers with Ansible
    - 2.1.2.14.3.4 Installing MariaDB .deb Files with Ansible
    - 2.1.2.14.3.5 Running mariadb-tzinfo-to-sql with Ansible
    - 2.1.2.14.3.6 Managing Secrets in Ansible
  - 2.1.2.14.4 Puppet and MariaDB
    - 2.1.2.14.4.1 Puppet Overview for MariaDB Users
    - 2.1.2.14.4.2 Bolt Examples
    - 2.1.2.14.4.3 Puppet hiera Configuration System
    - 2.1.2.14.4.4 Deploying Docker Containers with Puppet
    - 2.1.2.14.4.5 Existing Puppet Modules for MariaDB
  - 2.1.2.14.5 Vagrant and MariaDB
    - 2.1.2.14.5.1 Vagrant Overview for MariaDB Users
    - 2.1.2.14.5.2 Creating a Vagrantfile
    - 2.1.2.14.5.3 Vagrant Security Concerns
    - 2.1.2.14.5.4 Running MariaDB ColumnStore Docker containers on Linux, Windows and MacOS
  - 2.1.2.14.6 Docker and MariaDB
    - 2.1.2.14.6.1 Benefits of Managing Docker Containers with Orchestration Software
    - 2.1.2.14.6.2 Installing and Using MariaDB via Docker
    - 2.1.2.14.6.3 Running MariaDB ColumnStore Docker containers on Linux, Windows and MacOS
    - 2.1.2.14.6.4 Creating a Custom Docker Image
    - 2.1.2.14.6.5 Setting Up a LAMP Stack with Docker Compose
    - 2.1.2.14.6.6 Docker Security Concerns
    - 2.1.2.14.6.7 MariaDB Container Cheat Sheet
  - 2.1.2.14.7 Kubernetes and MariaDB
  - 2.1.2.14.8 Kubernetes Overview for MariaDB Users
  - 2.1.2.14.9 Kubernetes Operators for MariaDB
  - 2.1.2.14.10 Automating Upgrades with MariaDB.Org Downloads REST API
  - 2.1.2.14.11 HashiCorp Vault and MariaDB

- 2.1.2.14.12 Orchestrator Overview
- 2.1.2.14.13 Rotating Logs on Unix and Linux
- 2.1.2.14.14 Automating MariaDB Tasks with Events
- 2.1.2.15 MariaDB Package Repository Setup and Usage

## 2.1.3 Upgrading MariaDB

- 2.1.3.1 Upgrading Between Major MariaDB Versions
- 2.1.3.2 Upgrading Between Minor Versions on Linux
- 2.1.3.3 Upgrading from MariaDB 10.6 to MariaDB 10.7
- 2.1.3.4 Upgrading from MariaDB 10.5 to MariaDB 10.6
- 2.1.3.5 Upgrading from MariaDB 10.4 to MariaDB 10.5
- 2.1.3.6 Upgrading from MariaDB 10.3 to MariaDB 10.4
- 2.1.3.7 Upgrading from MariaDB 10.2 to MariaDB 10.3
- 2.1.3.8 Upgrading from MariaDB 10.1 to MariaDB 10.2
- 2.1.3.9 Upgrading MariaDB on Windows
- 2.1.3.10 Upgrading Galera Cluster
  - 2.1.3.10.1 Upgrading Between Minor Versions with Galera Cluster
  - 2.1.3.10.2 Upgrading from MariaDB 10.3 to MariaDB 10.4 with Galera Cluster
  - 2.1.3.10.3 Upgrading from MariaDB 10.2 to MariaDB 10.3 with Galera Cluster
  - 2.1.3.10.4 Upgrading from MariaDB 10.1 to MariaDB 10.2 with Galera Cluster
- 2.1.3.11 Upgrading from MySQL to MariaDB
  - 2.1.3.11.1 Upgrading from MySQL to MariaDB
  - 2.1.3.11.2 Moving from MySQL to MariaDB in Debian 9
  - 2.1.3.11.3 Screencast for Upgrading MySQL to MariaDB

## 2.1.4 Downgrading between Major Versions of MariaDB

## 2.1.5 Compiling MariaDB From Source

## 2.1.6

- 2.1.6.1 Starting and Stopping MariaDB
- 2.1.6.2 Configuring MariaDB with Option Files
- 2.1.6.3 mysqld Configuration Files and Groups
- 2.1.6.4 mysqld Options
- 2.1.6.5 What to Do if MariaDB Doesn't Start
- 2.1.6.6 Running MariaDB from the Build Directory
- 2.1.6.7 mysql.server
- 2.1.6.8 mysqld\_safe
- 2.1.6.9 mysqladmin
- 2.1.6.10 Switching Between Different Installed MariaDB Versions
- 2.1.6.11 Specifying Permissions for Schema (Data) Directories and Tables
- 2.1.6.12 mysqld\_multi
- 2.1.6.13 launchd
- 2.1.6.14 systemd
- 2.1.6.15 sysVinit

2.1.6.16 mariadb-admin

2.1.6.17 mariadb

2.1.6.18 mariadb-multi

2.1.6.19 mariadb-safe

## 2.1.7 MariaDB Performance & Advanced Configurations

2.1.7.1 Fusion-io

    2.1.7.1.1 Fusion-io Introduction

    2.1.7.1.2 Atomic Write Support

    2.1.7.1.3 InnoDB Page Flushing

2.1.7.3 Configuring Linux for MariaDB

2.1.7.4 Configuring MariaDB for Optimal Performance

2.1.7.5 Configuring Swappiness

## 2.1.8 Troubleshooting Installation Issues

2.1.8.1 Troubleshooting Connection Issues

2.1.8.2 Installation issues on Windows

2.1.8.3 Troubleshooting MariaDB Installs on Red Hat/CentOS

2.1.8.4 Installation issues on Debian and Ubuntu

    2.1.8.4.1 Differences in MariaDB in Debian (and Ubuntu)

    2.1.8.4.2 Moving from MySQL to MariaDB in Debian 9

    2.1.8.4.3 Creating a Debian Repository

    2.1.8.4.4 apt-upgrade Fails, But the Database is Running

2.1.8.5 What to Do if MariaDB Doesn't Start

2.1.8.6 Installing on an Old Linux Version

2.1.8.7 Error: symbol mysql\_get\_server\_name, version libmysqlclient\_16 not defined

## 2.1.9 Installing System Tables (mysql\_install\_db)

2.1.10 mysql\_install\_db.exe

2.1.11 Configuring MariaDB with Option Files

2.1.12 MariaDB Environment Variables

2.1.13 MariaDB on Amazon AWS

2.1.14 Migrating to MariaDB

    2.1.14.1 Migrating to MariaDB from MySQL

        2.1.14.1.1 MySQL vs MariaDB: Performance

        2.1.14.1.2 MariaDB versus MySQL: Compatibility

        2.1.14.1.3 Upgrading from MySQL to MariaDB

        2.1.14.1.4

# 3 High Availability & Performance Tuning

# 4 Columns, Storage Engines, and Plugins

## 4.1 Data Types

    4.1.1 Numeric Data Types

#### 4.1.1.1 Numeric Data Type Overview

4.1.1.2 TINYINT

4.1.1.3 BOOLEAN

4.1.1.4 SMALLINT

4.1.1.5 MEDIUMINT

4.1.1.6 INT

4.1.1.7 INTEGER

4.1.1.8 BIGINT

4.1.1.9 DECIMAL

4.1.1.10 DEC, NUMERIC, FIXED

4.1.1.11 NUMBER

4.1.1.12 FLOAT

4.1.1.13 DOUBLE

4.1.1.14 DOUBLE PRECISION

4.1.1.15 BIT

4.1.1.16 Floating-point Accuracy

4.1.1.17 INT1

4.1.1.18 INT2

4.1.1.19 INT3

4.1.1.20 INT4

4.1.1.21 INT8

#### 4.1.2 String Data Types

4.1.2.1 String Literals

4.1.2.2 BINARY

4.1.2.3 BLOB

4.1.2.4 BLOB and TEXT Data Types

4.1.2.5 CHAR

4.1.2.6 CHAR BYTE

4.1.2.7 ENUM

4.1.2.9 JSON Data Type

4.1.2.10 MEDIUMBLOB

4.1.2.11 MEDIUMTEXT

4.1.2.12 LONGBLOB

4.1.2.13 LONG and LONG VARCHAR

4.1.2.14 LONGTEXT

4.1.2.15 ROW

4.1.2.16 TEXT

4.1.2.17 TINYBLOB

4.1.2.18 TINYTEXT

4.1.2.19 VARBINARY

- 4.1.2.20 VARCHAR
  - 4.1.2.21 SET Data Type
  - 4.1.2.22 UUID Data Type
  - 4.1.2.23 Data Type Storage Requirements
  - 4.1.2.24 Supported Character Sets and Collations
  - 4.1.2.25 Character Sets and Collations
- 4.1.3 Data and Time Data Types
    - 4.1.3.1 DATE
    - 4.1.3.2 TIME
    - 4.1.3.3 DATETIME
    - 4.1.3.4 TIMESTAMP
    - 4.1.3.5 YEAR Data Type
  - 4.1.4 Geometry Types
  - 4.1.5 AUTO\_INCREMENT
  - 4.1.6 Data Type Storage Requirements
  - 4.1.7 AUTO\_INCREMENT FAQ
  - 4.1.8 NULL Values
- 4.2.1 Character Set and Collation Overview
  - 4.2.2 Supported Character Sets and Collations
  - 4.2.3 Setting Character Sets and Collations
  - 4.2.4 Unicode
  - 4.2.5 SHOW CHARACTER SET
  - 4.2.6 SHOW COLLATION
  - 4.2.7 Information Schema CHARACTER\_SETS Table
  - 4.2.8 Information Schema COLLATIONS Table
  - 4.2.9 Internationalization and Localization
  - 4.2.10 SET CHARACTER SET
  - 4.2.11 SET NAMES
- 4.3.1 Choosing the Right Storage Engine
    - 4.3.2.1 InnoDB Versions
    - 4.3.2.2 InnoDB Limitations
    - 4.3.2.3 InnoDB Troubleshooting
      - 4.3.2.3.1 InnoDB Troubleshooting Overview
      - 4.3.2.3.2 InnoDB Data Dictionary Troubleshooting
      - 4.3.2.3.3 InnoDB Recovery Modes
      - 4.3.2.3.4 Troubleshooting Row Size Too Large Errors with InnoDB
    - 4.3.2.4 InnoDB System Variables
    - 4.3.2.5 InnoDB Server Status Variables
    - 4.3.2.6 AUTO\_INCREMENT Handling in InnoDB

- 4.3.2.7 InnoDB Buffer Pool
  - 4.3.2.8 InnoDB Change Buffering
  - 4.3.2.9 InnoDB Doublewrite Buffer
  - 4.3.2.10 InnoDB Tablespaces
    - 4.3.2.10.1 InnoDB System Tablespaces
    - 4.3.2.10.2 InnoDB File-Per-Table Tablespaces
    - 4.3.2.10.3 InnoDB Temporary Tablespaces
  - 4.3.2.11 InnoDB File Format
  - 4.3.2.12
    - 4.3.2.12.1 InnoDB Row Formats Overview
    - 4.3.2.12.2 InnoDB REDUNDANT Row Format
    - 4.3.2.12.3 InnoDB COMPACT Row Format
    - 4.3.2.12.4 InnoDB DYNAMIC Row Format
    - 4.3.2.12.5 InnoDB COMPRESSED Row Format
    - 4.3.2.12.6 Troubleshooting Row Size Too Large Errors with InnoDB
  - 4.3.2.13 InnoDB Strict Mode
  - 4.3.2.14 InnoDB Redo Log
  - 4.3.2.15 InnoDB Undo Log
  - 4.3.2.16 InnoDB Page Flushing
  - 4.3.2.17 InnoDB Purge
  - 4.3.2.18 Information Schema InnoDB Tables
  - 4.3.2.19 InnoDB Online DDL
    - 4.3.2.19.1 InnoDB Online DDL Overview
    - 4.3.2.19.2 InnoDB Online DDL Operations with the INPLACE Alter Algorithm
    - 4.3.2.19.3 InnoDB Online DDL Operations with the NOCOPY Alter Algorithm
    - 4.3.2.19.4 InnoDB Online DDL Operations with the INSTANT Alter Algorithm
    - 4.3.2.19.5 Instant ADD COLUMN for InnoDB
  - 4.3.2.20 Binary Log Group Commit and InnoDB Flushing Performance
- 4.3.3 MariaDB ColumnStore
- 4.3.4 Aria
- 4.3.4.1 Aria Storage Engine
  - 4.3.4.2 Aria Clients and Utilities
    - 4.3.4.2.1 aria\_chk
    - 4.3.4.2.2 aria\_pack
    - 4.3.4.2.3 aria\_read\_log
    - 4.3.4.2.4 aria\_s3\_copy
  - 4.3.4.3 Aria FAQ
  - 4.3.4.4 Aria Storage Formats
  - 4.3.4.5 Aria Status Variables
  - 4.3.4.6 Aria System Variables
  - 4.3.4.7 Aria Group Commit

- 4.3.4.8 Benchmarking Aria
- 4.3.4.9 Aria Two-step Deadlock Detection
- 4.3.4.10 Aria Encryption Overview
- 4.3.4.11 The Aria Name
- 4.3.5 Archive
- 4.3.6 BLACKHOLE
- 4.3.7 CONNECT
  - 4.3.7.1 Introduction to the CONNECT Engine
  - 4.3.7.2 Installing the CONNECT Storage Engine
  - 4.3.7.3 Creating and Dropping CONNECT Tables
  - 4.3.7.4 CONNECT Data Types
  - 4.3.7.5 Current Status of the CONNECT Handler
  - 4.3.7.6 CONNECT Table Types
    - 4.3.7.6.1 CONNECT Table Types Overview
    - 4.3.7.6.2 Inward and Outward Tables
    - 4.3.7.6.3 CONNECT Table Types - Data Files
    - 4.3.7.6.4 CONNECT Zipped File Tables
    - 4.3.7.6.5 CONNECT DOS and FIX Table Types
    - 4.3.7.6.6 CONNECT DBF Table Type
    - 4.3.7.6.7 CONNECT BIN Table Type
    - 4.3.7.6.8 CONNECT VEC Table Type
    - 4.3.7.6.9 CONNECT CSV and FMT Table Types
    - 4.3.7.6.10 CONNECT - NoSQL Table Types
    - 4.3.7.6.11 CONNECT - Files Retrieved Using Rest Queries
    - 4.3.7.6.12 CONNECT JSON Table Type
    - 4.3.7.6.13 CONNECT XML Table Type
    - 4.3.7.6.14 CONNECTINI Table Type
    - 4.3.7.6.15 CONNECT - External Table Types
    - 4.3.7.6.16 CONNECT ODBC Table Type: Accessing Tables From Another DBMS
    - 4.3.7.6.17 CONNECT JDBC Table Type: Accessing Tables from Another DBMS
    - 4.3.7.6.18 CONNECT MONGO Table Type: Accessing Collections from MongoDB
    - 4.3.7.6.19 CONNECT MYSQL Table Type: Accessing MySQL/MariaDB Tables
    - 4.3.7.6.20 CONNECT PROXY Table Type
    - 4.3.7.6.21 CONNECT XCOL Table Type
    - 4.3.7.6.22 CONNECT OCCUR Table Type
    - 4.3.7.6.23 CONNECT PIVOT Table Type
    - 4.3.7.6.24 CONNECT TBL Table Type: Table List
    - 4.3.7.6.25 CONNECT - Using the TBL and MYSQL Table Types Together
    - 4.3.7.6.26 CONNECT Table Types - Special "Virtual" Tables
    - 4.3.7.6.27 CONNECT Table Types - VIR
    - 4.3.7.6.28 CONNECT Table Types - OEM: Implemented in an External LIB
    - 4.3.7.6.29 CONNECT Table Types - Catalog Tables

- 4.3.7.7 CONNECT - Security
  - 4.3.7.8 CONNECT - OEM Table Example
  - 4.3.7.9 Using CONNECT
    - 4.3.7.9.1 Using CONNECT - General Information
    - 4.3.7.9.2 Using CONNECT - Virtual and Special Columns
    - 4.3.7.9.3 Using CONNECT - Importing File Data Into MariaDB Tables
    - 4.3.7.9.4 Using CONNECT - Exporting Data From MariaDB
    - 4.3.7.9.5 Using CONNECT - Indexing
    - 4.3.7.9.6 Using CONNECT - Condition Pushdown
    - 4.3.7.9.7 USING CONNECT - Offline Documentation
    - 4.3.7.9.8 Using CONNECT - Partitioning and Sharding
  - 4.3.7.10 CONNECT - Making the GetRest Library
  - 4.3.7.11 CONNECT - Adding the REST Feature as a Library Called by an OEM Table
  - 4.3.7.12 CONNECT - Compiling JSON UDFs in a Separate Library
  - 4.3.7.13 CONNECT System Variables
  - 4.3.7.14 JSON Sample Files
- 4.3.8 CSV
    - 4.3.8.1 CSV Overview
    - 4.3.8.2 Checking and Repairing CSV Tables
  - 4.3.9 FederatedX
    - 4.3.9.1 About FederatedX
    - 4.3.9.2 Differences Between FederatedX and Federated
  - 4.3.10 MEMORY Storage Engine
  - 4.3.11 MERGE
  - 4.3.12 Mroonga
    - 4.3.12.1 About Mroonga
    - 4.3.12.2 Mroonga Overview
    - 4.3.12.3 Mroonga Status Variables
    - 4.3.12.4 Mroonga System Variables
    - 4.3.12.5
      - 4.3.12.5.1 Creating Mroonga User-Defined Functions
      - 4.3.12.5.2 last\_insert\_grn\_id
      - 4.3.12.5.3 mroonga\_command
      - 4.3.12.5.4 mroonga\_escape
      - 4.3.12.5.5 mroonga\_highlight\_html
      - 4.3.12.5.6 mroonga\_normalize
      - 4.3.12.5.7 mroonga\_snippet
      - 4.3.12.5.8 mroonga\_snippet\_html
    - 4.3.12.6 Information Schema MROONGA\_STATS Table
  - 4.3.13 MyISAM
    - 4.3.13.1 MyISAM Overview

- 4.3.13.2 MyISAM System Variables
- 4.3.13.3 MyISAM Storage Formats
  - 4.3.13.4.1 myisamchk
  - 4.3.13.4.2 Memory and Disk Use With myisamchk
  - 4.3.13.4.3 myisamchk Table Information
  - 4.3.13.4.4 myisamlog
  - 4.3.13.4.5 myisampack
  - 4.3.13.4.6 myisam\_ftdump
- 4.3.13.5 MyISAM Index Storage Space
- 4.3.13.6 MyISAM Log
- 4.3.13.7 Concurrent Inserts
- 4.3.13.8 Segmented Key Cache
- 4.3.14 MyRocks
  - 4.3.14.1 About MyRocks for MariaDB
  - 4.3.14.2 Getting Started with MyRocks
  - 4.3.14.3 Building MyRocks in MariaDB
  - 4.3.14.4 Loading Data Into MyRocks
  - 4.3.14.5 MyRocks Status Variables
  - 4.3.14.6 MyRocks System Variables
  - 4.3.14.7 MyRocks Transactional Isolation
  - 4.3.14.8 MyRocks and Replication
  - 4.3.14.9 MyRocks and Group Commit with Binary log
  - 4.3.14.10 Optimizer Statistics in MyRocks
  - 4.3.14.11 Differences Between MyRocks Variants
  - 4.3.14.12 MyRocks and Bloom Filters
  - 4.3.14.13 MyRocks and CHECK TABLE
  - 4.3.14.14 MyRocks and Data Compression
  - 4.3.14.15 MyRocks and Index-Only Scans
  - 4.3.14.16 MyRocks and START TRANSACTION WITH CONSISTENT SNAPSHOT
  - 4.3.14.17 MyRocks Column Families
  - 4.3.14.18 MyRocks in MariaDB 10.2 vs MariaDB 10.3
  - 4.3.14.19 MyRocks Performance Troubleshooting
- 4.3.15 OQGRAPH
  - 4.3.15.1 Installing OQGRAPH
  - 4.3.15.2 OQGRAPH Overview
  - 4.3.15.3 OQGRAPH Examples
  - 4.3.15.4 Compiling OQGRAPH
  - 4.3.15.5 Building OQGRAPH Under Windows
  - 4.3.15.6 OQGRAPH System and Status Variables
- 4.3.16 S3 Storage Engine

- 4.3.16.1 Using the S3 Storage Engine
- 4.3.16.2 Testing the Connections to S3
- 4.3.16.3 S3 Storage Engine Internals
- 4.3.16.4 aria\_s3\_copy
- 4.3.16.5 S3 Storage Engine Status Variables
- 4.3.16.6 S3 Storage Engine System Variables
- 4.3.17 Sequence Storage Engine
- 4.3.18 SphinxSE
  - 4.3.18.1 About SphinxSE
  - 4.3.18.2 Installing Sphinx
  - 4.3.18.3 Configuring Sphinx
  - 4.3.18.4 Installing and Testing SphinxSE with MariaDB
  - 4.3.18.5 Sphinx Status Variables
- 4.3.19 Spider
  - 4.3.19.10 Spider Functions
    - 4.3.19.10.1 SPIDER\_BG\_DIRECT\_SQL
    - 4.3.19.10.2 SPIDER\_COPY\_TABLES
    - 4.3.19.10.3 SPIDER\_DIRECT\_SQL
    - 4.3.19.10.4 SPIDER\_FLUSH\_TABLE\_MON\_CACHE
  - 4.3.19.11 Spider mysql Database Tables
    - 4.3.19.11.1 mysqlspider\_link\_failed\_log Table
    - 4.3.19.11.2 mysqlspider\_link\_mon\_servers Table
    - 4.3.19.11.3 mysqlspider\_tables Table
    - 4.3.19.11.4 mysqlspider\_table\_crd Table
    - 4.3.19.11.5 mysqlspider\_table\_position\_for\_recovery Table
    - 4.3.19.11.6 mysqlspider\_table\_sts Table
    - 4.3.19.11.7 mysqlspider\_xa Table
    - 4.3.19.11.8 mysqlspider\_xa\_failed\_log Table
    - 4.3.19.11.9 mysqlspider\_xa\_member Table
  - 4.3.19.12 Information Schema SPIDER\_ALLOC\_MEM Table
  - 4.3.19.13 Information Schema SPIDER\_WRAPPER\_PROTOCOLS Table
  - 4.3.19.14 Spider Differences Between SpiderForMySQL and MariaDB
  - 4.3.19.15 Spider Case Studies
  - 4.3.19.16 Spider Benchmarks
  - 4.3.19.17 Spider FAQ
- 4.3.20 Information Schema ENGINES Table
- 4.3.21 PERFORMANCE\_SCHEMA Storage Engine
- 4.3.22 Storage Engine Development
  - 4.3.22.1 Storage Engine FAQ
  - 4.3.22.2 Engine-defined New Table/Field/Index Attributes
  - 4.3.22.3 Table Discovery

#### 4.3.23 Converting Tables from MyISAM to InnoDB

#### 4.3.24 Machine Learning with MindsDB

### 4.4 Plugins

#### 4.4.1 Information on Plugins

##### 4.4.1.1 List of Plugins

##### 4.4.1.2 Information Schema PLUGINS Table

##### 4.4.1.3 Information Schema ALL\_PLUGINS Table

#### 4.4.2 Plugin SQL Statements

#### 4.4.3 Installing, Using, and Creating Plugins

##### 4.4.3.1 Specifying Which Plugins to Build

##### 4.4.3.2 Writing Plugins for MariaDB

#### 4.4.4 MariaDB Audit Plugin

##### 4.4.4.1 MariaDB Audit Plugin - Installation

##### 4.4.4.2 MariaDB Audit Plugin - Configuration

##### 4.4.4.3 MariaDB Audit Plugin - Log Settings

##### 4.4.4.4 MariaDB Audit Plugin - Location and Rotation of Logs

##### 4.4.4.5 MariaDB Audit Plugin - Log Format

##### 4.4.4.6 MariaDB Audit Plugin - Versions

##### 4.4.4.7 MariaDB Audit Plugin Options and System Variables

##### 4.4.4.8 MariaDB Audit Plugin - Status Variables

#### 4.4.5 Authentication Plugins

##### 4.4.5.1 Pluggable Authentication Overview

##### 4.4.5.2 Authentication Plugin - mysql\_native\_password

##### 4.4.5.3 Authentication Plugin - mysql\_old\_password

##### 4.4.5.4 Authentication Plugin - ed25519

##### 4.4.5.5 Authentication Plugin - GSSAPI

##### 4.4.5.6 Authentication with Pluggable Authentication Modules (PAM)

##### 4.4.5.7 Authentication Plugin - Unix Socket

##### 4.4.5.8 Authentication Plugin - Named Pipe

##### 4.4.5.9 Authentication Plugin - SHA-256

#### 4.4.6 Password Validation Plugins

##### 4.4.6.1 Simple Password Check Plugin

##### 4.4.6.2 Cracklib Password Check Plugin

##### 4.4.6.3 Password Reuse Check Plugin

##### 4.4.6.4 Password Validation Plugin API

##### 4.4.6.5 password\_reuse\_check\_interval

#### 4.4.7 Key Management and Encryption Plugin

##### 4.4.7.1 Encryption Key Management

##### 4.4.7.2 File Key Management Encryption Plugin

##### 4.4.7.3 AWS Key Management Encryption Plugin

4.4.7.4 Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Setup Guide

4.4.7.5 Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Advanced Usage

4.4.7.6 Eperi Key Management Encryption Plugin

4.4.7.7 Encryption Plugin API

4.4.8

4.4.8.1 Semisynchronous Replication

4.4.8.2 WSREP\_INFO Plugin

4.4.9 Storage Engines

4.4.10

4.4.10.1 Feedback Plugin

4.4.10.2 Locales Plugin

4.4.10.3 METADATA\_LOCK\_INFO Plugin

4.4.10.4 Query Cache Information Plugin

4.4.10.5 Query Response Time Plugin

4.4.10.6 SQL Error Log Plugin

4.4.10.7 User Statistics

4.4.10.8 User Variables Plugin

4.4.10.9 Disks Plugin

4.4.10.10 Compression Plugins

## 5 MariaDB Community Bug Reporting

# 1 SQL Statements and Structure

## 1.1 SQL Statements

### 1.1.1 Account Management SQL Commands

#### 1.1.1.1 CREATE USER

##### Syntax

```
CREATE [OR REPLACE] USER [IF NOT EXISTS]
  user_specification [,user_specification ...]
  [REQUIRE {NONE | tls_option [[AND] tls_option ...]} ]
  [WITH resource_option [resource_option ...] ]
  [Lock_option] [password_option]

user_specification:
  username [authentication_option]

authentication_option:
  IDENTIFIED BY 'password'
  | IDENTIFIED BY PASSWORD 'password_hash'
  | IDENTIFIED {VIA|WITH} authentication_rule [OR authentication_rule ...]

authentication_rule:
  authentication_plugin
  | authentication_plugin {USING|AS} 'authentication_string'
  | authentication_plugin {USING|AS} PASSWORD('password')

tls_option:
  SSL
  | X509
  | CIPHER 'cipher'
  | ISSUER 'issuer'
  | SUBJECT 'subject'

resource_option:
  MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
  | MAX_STATEMENT_TIME time

password_option:
  PASSWORD EXPIRE
  | PASSWORD EXPIRE DEFAULT
  | PASSWORD EXPIRE NEVER
  | PASSWORD EXPIRE INTERVAL N DAY

lock_option:
  ACCOUNT LOCK
  | ACCOUNT UNLOCK
}
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [OR REPLACE](#)
4. [IF NOT EXISTS](#)
5. [Authentication Options](#)
  1. [IDENTIFIED BY 'password'](#)
  2. [IDENTIFIED BY PASSWORD 'password\\_hash'](#)
  3. [IDENTIFIED {VIA|WITH} authentication\\_plugin](#)
6. [TLS Options](#)
7. [Resource Limit Options](#)
8. [Account Names](#)
  1. [Host Name Component](#)
  2. [User Name Component](#)
  3. [Anonymous Accounts](#)
    1. [Fixing a Legacy Default Anonymous Account](#)
9. [Password Expiry](#)
10. [Account Locking](#)
11. [See Also](#)

## Description

The

```
CREATE USER
```

statement creates new MariaDB accounts. To use it, you must have the global [CREATE USER](#) privilege or the [INSERT](#) privilege for the [mysql](#) database. For each account,

```
CREATE USER
```

creates a new row in [mysql.user](#) (until MariaDB 10.3 this is a table, from MariaDB 10.4 it's a view) or [mysql.global\\_priv\\_table](#) (from MariaDB 10.4 ) that has no privileges.

If any of the specified accounts, or any permissions for the specified accounts, already exist, then the server returns

```
ERROR 1396 (HY000)
```

. If an error occurs,

```
CREATE USER
```

will still create the accounts that do not result in an error. Only one error is produced for all users which have not been created:

```
ERROR 1396 (HY000):
```

```
Operation CREATE USER failed for 'u1'@'%','u2'@'%'
```

```
CREATE USER
```

, [DROP USER](#) , [CREATE ROLE](#) , and [DROP ROLE](#) all produce the same error code when they fail.

See [Account Names](#) below for details on how account names are specified.

## OR REPLACE

If the optional

```
OR REPLACE
```

clause is used, it is basically a shortcut for:

```
DROP USER IF EXISTS name;  
CREATE USER name ...;
```

For example:

```
CREATE USER foo2@test IDENTIFIED BY 'password';  
ERROR 1396 (HY000): Operation CREATE USER failed for 'foo2'@'test'
```

```
CREATE OR REPLACE USER foo2@test IDENTIFIED BY 'password';  
Query OK, 0 rows affected (0.00 sec)
```

## IF NOT EXISTS

When the

```
IF NOT EXISTS
```

clause is used, MariaDB will return a warning instead of an error if the specified user already exists.

For example:

```
CREATE USER foo2@test IDENTIFIED BY 'password';
ERROR 1396 (HY000): Operation CREATE USER failed for 'foo2'@'test'

CREATE USER IF NOT EXISTS foo2@test IDENTIFIED BY 'password';
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Note  | 1973 | Can't create user 'foo2'@'test'; it already exists |
+-----+-----+
```

## Authentication Options

### IDENTIFIED BY 'password'

The optional

IDENTIFIED BY

clause can be used to provide an account with a password. The password should be specified in plain text. It will be hashed by the [PASSWORD](#) function prior to being stored in the [mysql.user](#) / [mysql.global\\_priv\\_table](#) table.

For example, if our password is

mariadb

, then we can create the user with:

```
CREATE USER foo2@test IDENTIFIED BY 'mariadb';
```

If you do not specify a password with the

IDENTIFIED BY

clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

The only [authentication plugins](#) that this clause supports are [mysql\\_native\\_password](#) and [mysql\\_old\\_password](#).

### IDENTIFIED BY PASSWORD 'password\_hash'

The optional

IDENTIFIED BY PASSWORD

clause can be used to provide an account with a password that has already been hashed. The password should be specified as a hash that was provided by the [PASSWORD](#) function. It will be stored in the [mysql.user](#) / [mysql.global\\_priv\\_table](#) table as-is.

For example, if our password is

mariadb

, then we can find the hash with:

```
SELECT PASSWORD('mariadb');
+-----+
| PASSWORD('mariadb')           |
+-----+
| *54958E764CE10E50764C2EECB71D01F08549980 |
+-----+
1 row in set (0.00 sec)
```

And then we can create a user with the hash:

```
CREATE USER foo2@test IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECB71D01F08549980';
```

If you do not specify a password with the

IDENTIFIED BY

clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

The only [authentication plugins](#) that this clause supports are [mysql\\_native\\_password](#) and [mysql\\_old\\_password](#).

### IDENTIFIED {VIA|WITH} authentication\_plugin

The optional

IDENTIFIED VIA authentication\_plugin

allows you to specify that the account should be authenticated by a specific [authentication plugin](#). The plugin name must be an active authentication plugin as per [SHOW PLUGINS](#). If it doesn't show up in that output, then you will need to install it with [INSTALL PLUGIN](#) or [INSTALL SONAME](#).

For example, this could be used with the [PAM authentication plugin](#):

```
CREATE USER foo2@test IDENTIFIED VIA pam;
```

Some authentication plugins allow additional arguments to be specified after a

USING

or

AS

keyword. For example, the [PAM authentication plugin](#) accepts a [service name](#):

```
CREATE USER foo2@test IDENTIFIED VIA pam USING 'mariadb';
```

The exact meaning of the additional argument would depend on the specific authentication plugin.

MariaDB starting with [10.4.0](#)

The

USING

or

AS

keyword can also be used to provide a plain-text password to a plugin if it's provided as an argument to the [PASSWORD\(\)](#) function. This is only valid for [authentication plugins](#) that have implemented a hook for the [PASSWORD\(\)](#) function. For example, the [ed25519](#) authentication plugin supports this:

```
CREATE USER safe@'%' IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

MariaDB starting with [10.4.3](#)

One can specify many authentication plugins, they all work as alternatives ways of authenticating a user:

```
CREATE USER safe@'%' IDENTIFIED VIA ed25519 USING PASSWORD('secret') OR unix_socket;
```

By default, when you create a user without specifying an authentication plugin, MariaDB uses the [mysql\\_native\\_password](#) plugin.

## TLS Options

By default, MariaDB transmits data between the server and clients without encrypting it. This is generally acceptable when the server and client run on the same host or in networks where security is guaranteed through other means. However, in cases where the server and client exist on separate networks or they are in a high-risk network, the lack of encryption does introduce security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

To mitigate this concern, MariaDB allows you to encrypt data in transit between the server and clients using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix `ssl_`, but internally, MariaDB only supports its secure successors.

See [Secure Connections Overview](#) for more information about how to determine whether your MariaDB server has TLS support.

You can set certain TLS-related restrictions for specific user accounts. For instance, you might use this with user accounts that require access to sensitive data while sending it across networks that you do not control. These restrictions can be enabled for a user account with the [CREATE USER](#), [ALTER USER](#), or [GRANT](#) statements. The following options are available:

Option	Description
REQUIRE NONE	TLS is not required for this account, but can still be used.
REQUIRE SSL	The account must use TLS, but no valid X509 certificate is required. This option cannot be combined with other TLS options.

REQUIRE X509	The account must use TLS and must have a valid X509 certificate. This option implies REQUIRE SSL . This option cannot be combined with other TLS options.
REQUIRE ISSUER 'issuer'	The account must use TLS and must have a valid X509 certificate. Also, the Certificate Authority must be the one specified via the string issuer . This option implies REQUIRE X509 . This option can be combined with the SUBJECT , and CIPHER options in any order.
REQUIRE SUBJECT 'subject'	The account must use TLS and must have a valid X509 certificate. Also, the certificate's Subject must be the one specified via the string subject . This option implies REQUIRE X509 . This option can be combined with the ISSUER , and CIPHER options in any order.
REQUIRE CIPHER 'cipher'	The account must use TLS, but no valid X509 certificate is required. Also, the encryption used for the connection must use a specific cipher method specified in the string cipher . This option implies REQUIRE SSL . This option can be combined with the ISSUER , and SUBJECT options in any order.

The

REQUIRE  
keyword must be used only once for all specified options, and the  
AND  
keyword can be used to separate individual options, but it is not required.

For example, you can create a user account that requires these TLS options with the following:

```
CREATE USER 'alice'@'%'  
REQUIRE SUBJECT '/CN=alice/0=My Dom, Inc./C=US/ST=Oregon/L=Portland'  
AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter Parker/emailAddress=p.parker@marvel.com'  
AND CIPHER 'SHA-DES-CBC3-EDH-RSA';
```

If any of these options are set for a specific user account, then any client who tries to connect with that user account will have to be configured to connect with TLS.

See [Securing Connections for Client and Server](#) for information on how to enable TLS on the client and server.

## Resource Limit Options

MariaDB starting with 10.2.0

MariaDB 10.2.0 introduced a number of resource limit options.

It is possible to set per-account limits for certain server resources. The following table shows the values that can be set per account:

Limit Type	Description
MAX_QUERIES_PER_HOUR	Number of statements that the account can issue per hour (including updates)
MAX_UPDATES_PER_HOUR	Number of updates (not queries) that the account can issue per hour

MAX_CONNECTIONS_PER_HOUR	Number of connections that the account can start per hour
MAX_USER_CONNECTIONS	Number of simultaneous connections that can be accepted from the same account; if it is 0, max_connections will be used instead; if max_connections is 0, there is no limit for this account's simultaneous connections.
MAX_STATEMENT_TIME	Timeout, in seconds, for statements executed by the user. See also <a href="#">Aborting Statements that Exceed a Certain Time to Execute</a> .

If any of these limits are set to

0

, then there is no limit for that resource for that user.

Here is an example showing how to create a user with resource limits:

```
CREATE USER 'someone'@'localhost' WITH
  MAX_USER_CONNECTIONS 10
  MAX_QUERIES_PER_HOUR 200;
```

The resources are tracked per account, which means

'user'@'server'  
; not per user name or per connection.

The count can be reset for all users using [FLUSH USER\\_RESOURCES](#), [FLUSH PRIVILEGES](#) or [mysqladmin reload](#).

Per account resource limits are stored in the `user` table, in the `mysql` database. Columns used for resources limits are named

max\_questions  
,  
max\_updates  
,  
max\_connections  
(for  
MAX\_CONNECTIONS\_PER\_HOUR  
, and  
max\_user\_connections  
(for  
MAX\_USER\_CONNECTIONS  
)�.

## Account Names

Account names have both a user name component and a host name component, and are specified as

'user\_name'@'host\_name'

The user name and host name may be unquoted, quoted as strings using double quotes (

"

) or single quotes (

'), or quoted as identifiers using backticks (

). You must use quotes when using special characters (such as a hyphen) or wildcard characters. If you quote, you must quote the user name and host name separately (for example

'user\_name'@'host\_name'  
).

## Host Name Component

If the host name is not provided, it is assumed to be

'%'

Host names may contain the wildcard characters

%  
and  
\_

. They are matched as if by the [LIKE](#) clause. If you need to use a wildcard character literally (for example, to match a domain name with an underscore), prefix the character with a backslash. See

[LIKE](#)

for more information on escaping wildcard characters.

Host name matches are case-insensitive. Host names can match either domain names or IP addresses. Use

'localhost'

as the host name to allow only local client connections.

You can use a netmask to match a range of IP addresses using

'base\_ip/netmask'

as the host name. A user with an IP address *ip\_addr* will be allowed to connect if the following condition is true:

```
ip_addr & netmask = base_ip
```

For example, given a user:

```
CREATE USER 'maria'@'247.150.130.0/255.255.255.0';
```

the IP addresses satisfying this condition range from 247.150.130.0 to 247.150.130.255.

Using

255.255.255.255

is equivalent to not using a netmask at all. Netmasks cannot be used for IPv6 addresses.

Note that the credentials added when creating a user with the

'%'

wildcard host will not grant access in all cases. For example, some systems come with an anonymous localhost user, and when connecting from localhost this will take precedence.

Before [MariaDB 10.6](#), the host name component could be up to 60 characters in length. Starting from [MariaDB 10.6](#), it can be up to 255 characters.

## User Name Component

User names must match exactly, including case. A user name that is empty is known as an anonymous account and is allowed to match a login attempt with any user name component. These are described more in the next section.

For valid identifiers to use as user names, see [Identifier Names](#).

It is possible for more than one account to match when a user connects. MariaDB selects the first matching account after sorting according to the following criteria:

- Accounts with an exact host name are sorted before accounts using a wildcard in the host name. Host names using a netmask are considered to be exact for sorting.
- Accounts with a wildcard in the host name are sorted according to the position of the first wildcard character. Those with a wildcard character later in the host name sort before those with a wildcard character earlier in the host name.
- Accounts with a non-empty user name sort before accounts with an empty user name.
- Accounts with an empty user name are sorted last. As mentioned previously, these are known as anonymous accounts. These are described more in the next section.

The following table shows a list of example account as sorted by these criteria:

User	Host
joffrey	192.168.0.3
	192.168.0.%
joffrey	192.168.%
	192.168.%

Once connected, you only have the privileges granted to the account that matched, not all accounts that could have matched. For example, consider the following commands:

```
CREATE USER 'joffrey'@'192.168.0.3';
CREATE USER 'joffrey'@'%';
GRANT SELECT ON test.t1 TO 'joffrey'@'192.168.0.3';
GRANT SELECT ON test.t2 TO 'joffrey'@'%';
```

If you connect as joffrey from

192.168.0.3

, you will have the

SELECT

privilege on the table

```
test.t1
, but not on the table
test.t2
. If you connect as joffrey from any other IP address, you will have the
SELECT
privilege on the table
test.t2
, but not on the table
test.t1
.
```

Usernames can be up to 80 characters long before 10.6 and starting from 10.6 it can be 128 characters long.

## Anonymous Accounts

Anonymous accounts are accounts where the user name portion of the account name is empty. These accounts act as special catch-all accounts. If a user attempts to log into the system from a host, and an anonymous account exists with a host name portion that matches the user's host, then the user will log in as the anonymous account if there is no more specific account match for the user name that the user entered.

For example, here are some anonymous accounts:

```
CREATE USER ''@'localhost';
CREATE USER ''@'192.168.0.3';
```

### Fixing a Legacy Default Anonymous Account

On some systems, the `mysql.db` table has some entries for the

`'@'%'`

anonymous account by default. Unfortunately, there is no matching entry in the `mysql.user` / `mysql.global_priv_table` table, which means that this anonymous account doesn't exactly exist, but it does have privileges--usually on the default

`test`

database created by `mysql_install_db`. These account-less privileges are a legacy that is leftover from a time when MySQL's privilege system was less advanced.

This situation means that you will run into errors if you try to create a

`'@'%'`

account. For example:

```
CREATE USER ''@'%';
ERROR 1396 (HY000): Operation CREATE USER failed for ''@'%'
```

The fix is to `DELETE` the row in the `mysql.db` table and then execute `FLUSH PRIVILEGES`:

```
DELETE FROM mysql.db WHERE User=' ' AND Host='';
FLUSH PRIVILEGES;
```

And then the account can be created:

```
CREATE USER ''@'%';
Query OK, 0 rows affected (0.01 sec)
```

See [MDEV-13486](#) for more information.

## Password Expiry

MariaDB starting with 10.4.3

Besides automatic password expiry, as determined by `default_password_lifetime`, password expiry times can be set on an individual user basis, overriding the global setting, for example:

```
CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;
```

See [User Password Expiry](#) for more details.

## Account Locking

MariaDB starting with 10.4.2

Account locking permits privileged administrators to lock/unlock user accounts. No new client connections will be permitted if an account is locked

(existing connections are not affected). For example:

```
CREATE USER 'marijn'@'localhost' ACCOUNT LOCK;
```

See [Account Locking](#) for more details.

From [MariaDB 10.4.7](#) and [MariaDB 10.5.8](#), the *lock\_option* and *password\_option* clauses can occur in either order.

## See Also

- [Troubleshooting Connection Issues](#)
- [Authentication from MariaDB 10.4](#)
- [Identifier Names](#)
- [GRANT](#)
- [ALTER USER](#)
- [DROP USER](#)
- [SET PASSWORD](#)
- [SHOW CREATE USER](#)
- [mysql.user table](#)
- [mysql.global\\_priv\\_table](#)
- [Password Validation Plugins](#) - permits the setting of basic criteria for passwords
- [Authentication Plugins](#) - allow various authentication methods to be used, and new ones to be developed.

## 1.1.1.2 ALTER USER

MariaDB starting with [10.2.0](#)

The ALTER USER statement was introduced in [MariaDB 10.2.0](#).

## Syntax

```

ALTER USER [IF EXISTS]
  user_specification [,user_specification] ...
  [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
  [WITH resource_option [resource_option] ...]
  [lock_option] [password_option]

  user_specification:
    username [authentication_option]

  authentication_option:
    IDENTIFIED BY 'password'
    | IDENTIFIED BY PASSWORD 'password_hash'
    | IDENTIFIED {VIA|WITH} authentication_rule [OR authentication_rule] ...

  authentication_rule:
    authentication_plugin
    | authentication_plugin {USING|AS} 'authentication_string'
    | authentication_plugin {USING|AS} PASSWORD('password')

  tls_option
    SSL
    | X509
    | CIPHER 'cipher'
    | ISSUER 'issuer'
    | SUBJECT 'subject'

  resource_option
    MAX_QUERIES_PER_HOUR count
    | MAX_UPDATES_PER_HOUR count
    | MAX_CONNECTIONS_PER_HOUR count
    | MAX_USER_CONNECTIONS count
    | MAX_STATEMENT_TIME time

  password_option:
    PASSWORD EXPIRE
    | PASSWORD EXPIRE DEFAULT
    | PASSWORD EXPIRE NEVER
    | PASSWORD EXPIRE INTERVAL N DAY

  lock_option:
    ACCOUNT LOCK
    | ACCOUNT UNLOCK
}

```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [IF EXISTS](#)
4. [Account Names](#)
5. [Authentication Options](#)
  1. [IDENTIFIED BY 'password'](#)
  2. [IDENTIFIED BY PASSWORD 'password\\_hash'](#)
  3. [IDENTIFIED {VIA|WITH} authentication\\_plugin](#)
6. [TLS Options](#)
7. [Resource Limit Options](#)
8. [Password Expiry](#)
9. [Account Locking](#)
10. [See Also](#)

## Description

The

`ALTER USER`

statement modifies existing MariaDB accounts. To use it, you must have the global `CREATE USER` privilege or the `UPDATE` privilege for the `mysql` database. The global `SUPER` privilege is also required if the `read_only` system variable is enabled.

If any of the specified user accounts do not yet exist, an error results. If an error occurs,

`ALTER USER`

will still modify the accounts that do not result in an error. Only one error is produced for all users which have not been modified.

# IF EXISTS

When the

IF EXISTS

clause is used, MariaDB will return a warning instead of an error for each specified user that does not exist.

## Account Names

For

ALTER USER  
statements, account names are specified as the  
username  
argument in the same way as they are for [CREATE USER](#) statements. See [account names](#) from the  
[CREATE USER](#)  
page for details on how account names are specified.

[CURRENT\\_USER](#) or

[CURRENT\\_USER\(\)](#)  
can also be used to alter the account logged into the current session. For example, to change the current user's password to  
mariadb  
:

```
ALTER USER CURRENT_USER() IDENTIFIED BY 'mariadb';
```

## Authentication Options

MariaDB starting with [10.4](#)

From [MariaDB 10.4](#), it is possible to use more than one authentication plugin for each user account. For example, this can be useful to slowly migrate users to the more secure ed25519 authentication plugin over time, while allowing the old mysql\_native\_password authentication plugin as an alternative for the transitional period. See [Authentication from MariaDB 10.4](#) for more.

When running

ALTER USER  
, not specifying an authentication option in the IDENTIFIED VIA clause will remove that authentication method. (However this was not the case before [MariaDB 10.4.13](#), see [MDEV-21928](#))

For example, a user is created with the ability to authenticate via both a password and unix\_socket:

```
CREATE USER 'bob'@'localhost'  
IDENTIFIED VIA mysql_native_password USING PASSWORD('pwd')  
OR unix_socket;  
  
SHOW CREATE USER 'bob'@'localhost'  
***** 1. row *****  
CREATE USER for bob@localhost: CREATE USER `bob`@`localhost`  
IDENTIFIED VIA mysql_native_password USING '*975B2CD4FF9AE554FE8AD33168FBFC326D2021DD'  
OR unix_socket
```

If the user's password is updated, but unix\_socket authentication is not specified in the

IDENTIFIED VIA  
clause, unix\_socket authentication will no longer be permitted.

```
ALTER USER 'bob'@'localhost' IDENTIFIED VIA mysql_native_password USING PASSWORD('pwd2');  
  
SHOW CREATE USER 'bob'@'localhost'  
***** 1. row *****  
CREATE USER for bob@localhost: CREATE USER `bob`@`localhost`  
IDENTIFIED BY PASSWORD '*38366FDA01695B6A5A9DD4E428D9FB8F7EB75512'
```

## IDENTIFIED BY 'password'

The optional

IDENTIFIED BY

clause can be used to provide an account with a password. The password should be specified in plain text. It will be hashed by the [PASSWORD](#) function prior to being stored to the [mysql.user](#) table.

For example, if our password is

mariadb

, then we can set the account's password with:

```
ALTER USER foo2@test IDENTIFIED BY 'mariadb';
```

If you do not specify a password with the  
IDENTIFIED BY  
clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

The only [authentication plugins](#) that this clause supports are [mysql\\_native\\_password](#) and [mysql\\_old\\_password](#).

## IDENTIFIED BY PASSWORD 'password\_hash'

The optional  
IDENTIFIED BY PASSWORD  
clause can be used to provide an account with a password that has already been hashed. The password should be specified as a hash that was provided by the [PASSWORD](#) #function. It will be stored to the [mysql.user](#) table as-is.

For example, if our password is

mariadb

, then we can find the hash with:

```
SELECT PASSWORD('mariadb');
+-----+
| PASSWORD('mariadb') |
+-----+
| *54958E764CE10E50764C2EECB71D01F08549980 |
+-----+
```

And then we can set an account's password with the hash:

```
ALTER USER foo2@test IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECB71D01F08549980';
```

If you do not specify a password with the  
IDENTIFIED BY  
clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

The only [authentication plugins](#) that this clause supports are [mysql\\_native\\_password](#) and [mysql\\_old\\_password](#).

## IDENTIFIED {VIA|WITH} authentication\_plugin

The optional

IDENTIFIED VIA authentication\_plugin

allows you to specify that the account should be authenticated by a specific [authentication plugin](#). The plugin name must be an active authentication plugin as per [SHOW PLUGINS](#). If it doesn't show up in that output, then you will need to install it with [INSTALL PLUGIN](#) or [INSTALL SONAME](#).

For example, this could be used with the [PAM authentication plugin](#):

```
ALTER USER foo2@test IDENTIFIED VIA pam;
```

Some authentication plugins allow additional arguments to be specified after a

USING

or

AS

keyword. For example, the [PAM authentication plugin](#) accepts a [service name](#):

```
ALTER USER foo2@test IDENTIFIED VIA pam USING 'mariadb';
```

The exact meaning of the additional argument would depend on the specific authentication plugin.

In [MariaDB 10.4](#) and later, the

USING

or

AS

keyword can also be used to provide a plain-text password to a plugin if it's provided as an argument to the [PASSWORD\(\)](#) function. This is only valid for [authentication plugins](#) that have implemented a hook for the [PASSWORD\(\)](#) function. For example, the [ed25519](#) authentication plugin supports this:

```
ALTER USER safe@'%' IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

# TLS Options

By default, MariaDB transmits data between the server and clients without encrypting it. This is generally acceptable when the server and client run on the same host or in networks where security is guaranteed through other means. However, in cases where the server and client exist on separate networks or they are in a high-risk network, the lack of encryption does introduce security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

To mitigate this concern, MariaDB allows you to encrypt data in transit between the server and clients using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix `ssl_`, but internally, MariaDB only supports its secure successors.

See [Secure Connections Overview](#) for more information about how to determine whether your MariaDB server has TLS support.

You can set certain TLS-related restrictions for specific user accounts. For instance, you might use this with user accounts that require access to sensitive data while sending it across networks that you do not control. These restrictions can be enabled for a user account with the `CREATE USER`, `ALTER USER`, or `GRANT` statements. The following options are available:

Option	Description
<code>REQUIRE NONE</code>	TLS is not required for this account, but can still be used.
<code>REQUIRE SSL</code>	The account must use TLS, but no valid X509 certificate is required. This option cannot be combined with other TLS options.
<code>REQUIRE X509</code>	The account must use TLS and must have a valid X509 certificate. This option implies <code>REQUIRE SSL</code> . This option cannot be combined with other TLS options.
<code>'issuer' REQUIRE ISSUER</code>	The account must use TLS and must have a valid X509 certificate. Also, the Certificate Authority must be the one specified via the string <code>issuer</code> . This option implies <code>REQUIRE X509</code> . This option can be combined with the <code>SUBJECT</code> , and <code>CIPHER</code> options in any order.
<code>SUBJECT 'subject' REQUIRE</code>	The account must use TLS and must have a valid X509 certificate. Also, the certificate's Subject must be the one specified via the string <code>subject</code> . This option implies <code>REQUIRE X509</code> . This option can be combined with the <code>ISSUER</code> , and <code>CIPHER</code> options in any order.
<code>'cipher' REQUIRE CIPHER</code>	The account must use TLS, but no valid X509 certificate is required. Also, the encryption used for the connection must use a specific cipher method specified in the string <code>cipher</code> . This option implies <code>REQUIRE SSL</code> . This option can be combined with the <code>ISSUER</code> , and <code>SUBJECT</code> options in any order.

The

`REQUIRE`  
keyword must be used only once for all specified options, and the  
`AND`  
keyword can be used to separate individual options, but it is not required.

For example, you can alter a user account to require these TLS options with the following:

```

ALTER USER 'alice'@'%'
REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland'
AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter Parker/emailAddress=p.parker@marvel.com'
AND CIPHER 'SHA-DES-CBC3-EDH-RSA';

```

If any of these options are set for a specific user account, then any client who tries to connect with that user account will have to be configured to connect with TLS.

See [Securing Connections for Client and Server](#) for information on how to enable TLS on the client and server.

## Resource Limit Options

MariaDB starting with 10.2.0

MariaDB 10.2.0 introduced a number of resource limit options.

It is possible to set per-account limits for certain server resources. The following table shows the values that can be set per account:

Limit Type	Description
MAX_QUERIES_PER_HOUR	Number of statements that the account can issue per hour (including updates)
MAX_UPDATES_PER_HOUR	Number of updates (not queries) that the account can issue per hour
MAX_CONNECTIONS_PER_HOUR	Number of connections that the account can start per hour
MAX_USER_CONNECTIONS	Number of simultaneous connections that can be accepted from the same account; if it is 0, <code>max_connections</code> will be used instead; if <code>max_connections</code> is 0, there is no limit for this account's simultaneous connections.
MAX_STATEMENT_TIME	Timeout, in seconds, for statements executed by the user. See also <a href="#">Aborting Statements that Exceed a Certain Time to Execute</a> .

If any of these limits are set to

0

, then there is no limit for that resource for that user.

Here is an example showing how to set an account's resource limits:

```

ALTER USER 'someone'@'localhost' WITH
  MAX_USER_CONNECTIONS 10
  MAX_QUERIES_PER_HOUR 200;

```

The resources are tracked per account, which means

'user'@'server'  
; not per user name or per connection.

The count can be reset for all users using [FLUSH USER\\_RESOURCES](#), [FLUSH PRIVILEGES](#) or [mysqladmin reload](#).

Per account resource limits are stored in the `user` table, in the `mysql` database. Columns used for resources limits are named

`max_questions`  
,  
`max_updates`  
,  
`max_connections`  
(for  
`MAX_CONNECTIONS_PER_HOUR`  
, and  
`max_user_connections`  
(for  
`MAX_USER_CONNECTIONS`  
).

# Password Expiry

MariaDB starting with 10.4.3

Besides automatic password expiry, as determined by `default_password_lifetime`, password expiry times can be set on an individual user basis, overriding the global setting, for example:

```
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE NEVER;
ALTER USER 'monty'@'localhost' PASSWORD EXPIRE DEFAULT;
```

See [User Password Expiry](#) for more details.

# Account Locking

MariaDB starting with 10.4.2

Account locking permits privileged administrators to lock/unlock user accounts. No new client connections will be permitted if an account is locked (existing connections are not affected). For example:

```
ALTER USER 'marijn'@'localhost' ACCOUNT LOCK;
```

See [Account Locking](#) for more details.

From MariaDB 10.4.7 and MariaDB 10.5.8, the `lock_option` and `password_option` clauses can occur in either order.

## See Also

- [Authentication from MariaDB 10.4](#)
- [GRANT](#)
- [CREATE USER](#)
- [DROP USER](#)
- [SET PASSWORD](#)
- [SHOW CREATE USER](#)
- [mysql.user table](#)
- [Password Validation Plugins](#) - permits the setting of basic criteria for passwords
- [Authentication Plugins](#) - allow various authentication methods to be used, and new ones to be developed.

## 1.1.1.3 DROP USER

### Syntax

```
DROP USER [IF EXISTS] user_name [, user_name] ...
```

### Contents

1. [Syntax](#)
2. [Description](#)
  - 1. [IF EXISTS](#)
3. [Examples](#)
4. [See Also](#)

### Description

The

```
DROP USER
```

statement removes one or more MariaDB accounts. It removes privilege rows for the account from all grant tables. To use this statement, you must have the global `CREATE USER` privilege or the `DELETE` privilege for the `mysql` database. Each account is named using the same format as for the

```
CREATE USER
statement; for example,
'jeffrey'@'localhost'
. If you specify only the user name part of the account name, a host name part of
'%'
is used. For additional information about specifying account names, see CREATE USER.
```

Note that, if you specify an account that is currently connected, it will not be deleted until the connection is closed. The connection will not be automatically

closed.

If any of the specified user accounts do not exist,

```
ERROR 1396 (HY000)
results. If an error occurs,
DROP USER
```

will still drop the accounts that do not result in an error. Only one error is produced for all users which have not been dropped:

```
ERROR 1396 (HY000): Operation DROP USER failed for 'u1'@'%','u2'@'%'
```

Failed

```
CREATE
or
DROP
operations, for both users and roles, produce the same error code.
```

## IF EXISTS

If the

```
IF EXISTS
clause is used, MariaDB will return a note instead of an error if the user does not exist.
```

## Examples

```
DROP USER bob;
```

```
IF EXISTS
:
DROB USER bob;
ERROR 1396 (HY000): Operation DROB USER failed for 'bob'@'%'

DROB USER IF EXISTS bob;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1974 | Can't drop user 'bob'@'%'; it doesn't exist |
+-----+-----+
```

## See Also

- [CREATE USER](#)
- [ALTER USER](#)
- [GRANT](#)
- [SHOW CREATE USER](#)
- [mysql.user table](#)

## 1.1.1.4 GRANT

# Contents

- 1. [Syntax](#)
- 2. [Description](#)
- 3. [Account Names](#)
- 4. [Implicit Account Creation](#)
- 5. [Privilege Levels](#)
  - 1. [The USAGE Privilege](#)
  - 2. [The ALL PRIVILEGES Privilege](#)
  - 3. [The GRANT OPTION Privilege](#)
  - 4. [Global Privileges](#)
    - 1. [BINLOG ADMIN](#)
    - 2. [BINLOG MONITOR](#)
    - 3. [BINLOG REPLAY](#)
    - 4. [CONNECTION ADMIN](#)
    - 5. [CREATE USER](#)
    - 6. [FEDERATED ADMIN](#)
    - 7. [FILE](#)
    - 8. [GRANT OPTION](#)
    - 9. [PROCESS](#)
    - 10. [READ\\_ONLY ADMIN](#)
    - 11. [RELOAD](#)
    - 12. [REPLICATION CLIENT](#)
    - 13. [REPLICATION MASTER ADMIN](#)
    - 14. [REPLICA MONITOR](#)
    - 15. [REPLICATION REPLICA](#)
    - 16. [REPLICATION SLAVE](#)
    - 17. [REPLICATION SLAVE ADMIN](#)
    - 18. [SET USER](#)
    - 19. [SHOW DATABASES](#)
    - 20. [SHUTDOWN](#)
    - 21. [SUPER](#)
  - 5. [Database Privileges](#)
  - 6. [Table Privileges](#)
  - 7. [Column Privileges](#)
  - 8. [Function Privileges](#)
  - 9. [Procedure Privileges](#)
  - 10. [Proxy Privileges](#)
  - 6. [Authentication Options](#)
    - 1. [IDENTIFIED BY 'password'](#)
    - 2. [IDENTIFIED BY PASSWORD 'password\\_hash'](#)
    - 3. [IDENTIFIED {VIA|WITH} authentication\\_plugin](#)
  - 7. [Resource Limit Options](#)
  - 8. [TLS Options](#)
  - 9. [Roles](#)
    - 1. [Syntax](#)
  - 0. [Grant Examples](#)
    - 1. [Granting Root-like Privileges](#)
  - 1. [See Also](#)

## Syntax

```

GRANT
  priv_type [(column_list)]
  [, priv_type [(column_list)]] ...
  ON [object_type] priv_level
  TO user_specification [ user_options ...]

user_specification:
  username [authentication_option]

authentication_option:
  IDENTIFIED BY 'password'
  | IDENTIFIED BY PASSWORD 'password_hash'
  | IDENTIFIED {VIA|WITH} authentication_rule [OR authentication_rule ...]

authentication_rule:
  authentication_plugin
  | authentication_plugin {USING|AS} 'authentication_string'
  | authentication_plugin {USING|AS} PASSWORD('password')

GRANT PROXY ON username
  TO user_specification [, user_specification ...]
  [WITH GRANT OPTION]

GRANT rolename TO grantee [, grantee ...]
  [WITH ADMIN OPTION]

grantee:
  rolename
  username [authentication_option]

user_options:
  [REQUIRE {NONE | tls_option [[AND] tls_option] ...}]
  [WITH with_option [with_option] ...]

object_type:
  TABLE
  | FUNCTION
  | PROCEDURE
  | PACKAGE

priv_level:
  *
  | *.*
  | db_name.*
  | db_name.tbl_name
  | tbl_name
  | db_name.routine_name

with_option:
  GRANT OPTION
  | resource_option

resource_option:
  MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
  | MAX_STATEMENT_TIME time

tls_option:
  SSL
  | X509
  | CIPHER 'cipher'
  | ISSUER 'issuer'
  | SUBJECT 'subject'

```

## Description

The

```

GRANT
statement allows you to grant privileges or roles to accounts. To use
GRANT
, you must have the
GRANT OPTION

```

privilege, and you must have the privileges that you are granting.

Use the [REVOKE](#) statement to revoke privileges granted with the  
GRANT  
statement.

Use the [SHOW GRANTS](#) statement to determine what privileges an account has.

## Account Names

For

GRANT  
statements, account names are specified as the  
username  
argument in the same way as they are for [CREATE USER](#) statements. See [account names](#) from the  
[CREATE USER](#)  
page for details on how account names are specified.

## Implicit Account Creation

The

GRANT  
statement also allows you to implicitly create accounts in some cases.

If the account does not yet exist, then

GRANT  
can implicitly create it. To implicitly create an account with  
GRANT  
, a user is required to have the same privileges that would be required to explicitly create the account with the  
[CREATE USER](#)  
statement.

If the

NO\_AUTO\_CREATE\_USER  
[SQL\\_MODE](#) is set, then accounts can only be created if authentication information is specified, or with a [CREATE USER](#) statement. If no authentication information is provided,  
GRANT  
will produce an error when the specified account does not exist, for example:

```
show variables like '%sql_mode' ;
+-----+-----+
| Variable_name | Value           |
+-----+-----+
| sql_mode      | NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----+-----+

GRANT USAGE ON *.* TO 'user123'@'%' IDENTIFIED BY '';
ERROR 1133 (28000): Can't find any matching row in the user table

GRANT USAGE ON *.* TO 'user123'@'%' IDENTIFIED VIA PAM using 'mariadb' require ssl ;
Query OK, 0 rows affected (0.00 sec)

select host, user from mysql.user where user='user123' ;

+-----+-----+
| host | user   |
+-----+-----+
| %    | user123 |
+-----+-----+
```

## Privilege Levels

Privileges can be set globally, for an entire database, for a table or routine, or for individual columns in a table. Certain privileges can only be set at certain levels.

- Global privileges *priv\_type* are granted using  
\*.\*  
for *priv\_level*. Global privileges include privileges to administer the database and manage user accounts, as well as privileges for all tables, functions, and procedures. Global privileges are stored in the [mysql.user table](#).
- Database privileges *priv\_type* are granted using  
db\_name.\*  
for *priv\_level*, or using just

- \* to use default database. Database privileges include privileges to create tables and functions, as well as privileges for all tables, functions, and procedures in the database. Database privileges are stored in the [mysql.db table](#) .
- **Table privileges *priv\_type*** are granted using
 

```
db_name.tbl_name
for priv_level, or using just
tbl_name
to specify a table in the default database. The
TABLE
keyword is optional. Table privileges include the ability to select and change data in the table. Certain table privileges can be granted for individual columns.
```
- **Column privileges *priv\_type*** are granted by specifying a table for *priv\_level* and providing a column list after the privilege type. They allow you to control exactly which columns in a table users can select and change.
- **Function privileges *priv\_type*** are granted using
 

```
FUNCTION db_name.routine_name
for priv_level, or using just
FUNCTION routine_name
to specify a function in the default database.
```
- **Procedure privileges *priv\_type*** are granted using
 

```
PROCEDURE db_name.routine_name
for priv_level, or using just
PROCEDURE routine_name
to specify a procedure in the default database.
```

## The

### USAGE Privilege

The

```
USAGE
privilege grants no real privileges. The SHOW GRANTS statement will show a global
USAGE
privilege for a newly-created user. You can use
USAGE
with the
GRANT
statement to change options like
GRANT OPTION
and
MAX_USER_CONNECTIONS
without changing any account privileges.
```

## The

### ALL PRIVILEGES Privilege

The

```
ALL PRIVILEGES
privilege grants all available privileges. Granting all privileges only affects the given privilege level. For example, granting all privileges on a
table does not grant any privileges on the database or globally.
```

Using

```
ALL PRIVILEGES
does not grant the special
GRANT OPTION
privilege.
```

You can use

```
ALL
instead of
ALL PRIVILEGES
```

## The

### GRANT OPTION Privilege

Use the

```
WITH GRANT OPTION
```

clause to give users the ability to grant privileges to other users at the given privilege level. Users with the GRANT OPTION privilege can only grant privileges they have. They cannot grant privileges at a higher privilege level than they have the GRANT OPTION privilege.

The

GRANT OPTION  
privilege cannot be set for individual columns. If you use  
WITH GRANT OPTION  
when specifying [column privileges](#), the  
GRANT OPTION  
privilege will be granted for the entire table.

Using the

WITH GRANT OPTION  
clause is equivalent to listing  
GRANT OPTION  
as a privilege.

## Global Privileges

The following table lists the privileges that can be granted globally. You can also grant all database, table, and function privileges globally. When granted globally, these privileges apply to all databases, tables, or functions, including those created later.

To set a global privilege, use

\*.\*  
for *priv\_level*.

## BINLOG ADMIN

Enables administration of the [binary log](#), including the [PURGE BINARY LOGS](#) statement and setting the [binlog\\_annotation\\_row\\_events](#), [binlog\\_cache\\_size](#), [binlog\\_commit\\_wait\\_count](#), [binlog\\_commit\\_wait\\_usec](#), [binlog\\_direct\\_non\\_transactional\\_updates](#), [binlog\\_expire\\_logs\\_seconds](#), [binlog\\_file\\_cache\\_size](#), [binlog\\_format](#), [binlog\\_row\\_image](#), [binlog\\_row\\_metadata](#), [binlog\\_stmt\\_cache\\_size](#), [expire\\_logs\\_days](#), [log\\_bin\\_compress](#), [log\\_bin\\_compress\\_min\\_len](#), [log\\_bin\\_trust\\_function\\_creators](#), [max\\_binlog\\_cache\\_size](#), [max\\_binlog\\_size](#), [max\\_binlog\\_stmt\\_cache\\_size](#), [sql\\_log\\_bin](#) and [sync\\_binlog](#) system variables. Added in [MariaDB 10.5.2](#).

## BINLOG MONITOR

New name for [REPLICATION CLIENT](#) from [MariaDB 10.5.2](#),

REPLICATION CLIENT  
still supported as an alias for compatibility purposes). Permits running SHOW commands related to the [binary log](#), in particular the [SHOW BINLOG STATUS](#), [SHOW REPLICA STATUS](#) and [SHOW BINARY LOGS](#) statements.

## BINLOG REPLAY

Enables replaying the binary log with the [BINLOG](#) statement (generated by [mariadb-binlog](#)), executing [SET timestamp](#) when [secure\\_timestamp](#) is set to

replication  
, and setting the session values of system variables usually included in BINLOG output, in particular [gtid\\_domain\\_id](#), [gtid\\_seq\\_no](#), [pseudo\\_thread\\_id](#) and [server\\_id](#). Added in [MariaDB 10.5.2](#).

## CONNECTION ADMIN

Enables administering connection resource limit options. This includes ignoring the limits specified by [max\\_connections](#), [max\\_user\\_connections](#) and [max\\_password\\_errors](#), not executing the statements specified in [init\\_connect](#), [killing\\_connections\\_and\\_queries](#) owned by other users as well as setting the following connection-related system variables: [connect\\_timeout](#), [disconnect\\_on\\_expired\\_password](#), [extra\\_max\\_connections](#), [init\\_connect](#), [max\\_connections](#), [max\\_connect\\_errors](#), [max\\_password\\_errors](#), [proxy\\_protocol\\_networks](#), [secure\\_auth](#), [slow\\_launch\\_time](#), [thread\\_pool\\_exact\\_stats](#), [thread\\_pool\\_dedicated\\_listener](#), [thread\\_pool\\_idle\\_timeout](#), [thread\\_pool\\_max\\_threads](#), [thread\\_pool\\_min\\_threads](#), [thread\\_pool\\_mode](#), [thread\\_pool\\_oversubscribe](#), [thread\\_pool\\_prio\\_kickup\\_timer](#), [thread\\_pool\\_priority](#), [thread\\_pool\\_size](#), [thread\\_pool\\_stall\\_limit](#). Added in [MariaDB 10.5.2](#).

## CREATE USER

Create a user using the [CREATE USER](#) statement, or implicitly create a user with the  
GRANT  
statement.

## FEDERATED ADMIN

Execute [CREATE SERVER](#), [ALTER SERVER](#), and [DROP SERVER](#) statements. Added in [MariaDB 10.5.2](#).

## FILE

Read and write files on the server, using statements like [LOAD DATA INFILE](#) or functions like [LOAD\\_FILE\(\)](#). Also needed to create [CONNECT](#) outward tables. MariaDB server must have the permissions to access those files.

## GRANT OPTION

Grant global privileges. You can only grant privileges that you have.

## PROCESS

Show information about the active processes, for example via [SHOW PROCESSLIST](#) or [mysqladmin processlist](#). If you have the PROCESS privilege, you can see all threads. Otherwise, you can see only your own threads (that is, threads associated with the MariaDB account that you are using).

## READ\_ONLY ADMIN

User can set the [read\\_only](#) system variable and allows the user to perform write operations, even when the `read_only` option is active. Added in [MariaDB 10.5.2](#).

## RELOAD

Execute [FLUSH](#) statements or equivalent [mariadb-admin/mysqladmin](#) commands.

## REPLICATION CLIENT

Execute [SHOW MASTER STATUS](#), [SHOW SLAVE STATUS](#) and [SHOW BINARY LOGS](#) informative statements. Renamed to [BINLOG MONITOR](#) in [MariaDB 10.5.2](#) (but still supported as an alias for compatibility reasons).

## REPLICATION MASTER ADMIN

Permits administration of primary servers, including the [SHOW REPLICHOSTS](#) statement, and setting the [gtid\\_binlog\\_state](#), [gtid\\_domain\\_id](#), [master\\_verify\\_checksum](#) and [server\\_id](#) system variables. Added in [MariaDB 10.5.2](#).

## REPLICA MONITOR

Permit [SHOW REPLICASTATUS](#) and [SHOW RELAYLOG EVENTS](#). From [MariaDB 10.5.9](#).

When a user would upgrade from an older major release to a [MariaDB 10.5](#) minor release prior to [MariaDB 10.5.9](#), certain user accounts would lose capabilities. For example, a user account that had the REPLICATION CLIENT privilege in older major releases could run [SHOW REPLICASTATUS](#), but after upgrading to a [MariaDB 10.5](#) minor release prior to [MariaDB 10.5.9](#), they could no longer run [SHOW REPLICASTATUS](#), because that statement was changed to require the REPLICATION REPLICA ADMIN privilege.

This issue is fixed in [MariaDB 10.5.9](#) with this new privilege, which now grants the user the ability to execute `SHOW [ALL] (SLAVE | REPLICA) STATUS`.

When a database is upgraded from an older major release to MariaDB Server 10.5.9 or later, any user accounts with the REPLICATION CLIENT or REPLICATION SLAVE privileges will automatically be granted the new REPLICA MONITOR privilege. The privilege fix occurs when the server is started up, not when mariadb-upgrade is performed.

However, when a database is upgraded from an early 10.5 minor release to 10.5.9 and later, the user will have to fix any user account privileges manually.

## REPLICATION REPLICA

Synonym for [REPLICATION SLAVE](#). From [MariaDB 10.5.1](#).

## REPLICATION SLAVE

Accounts used by replica servers on the primary need this privilege. This is needed to get the updates made on the master. From [MariaDB 10.5.1](#), [REPLICATION REPLICA](#) is an alias for

`REPLICATION SLAVE`

## REPLICATION SLAVE ADMIN

Permits administering replica servers, including [START REPLICASLAVE](#), [STOP REPLICASLAVE](#), [CHANGE MASTER](#), [SHOW REPLICASLAVE STATUS](#), [SHOW RELAYLOG EVENTS](#) statements, replaying the binary log with the [BINLOG](#) statement (generated by [mariadb-binlog](#)), and setting the [gtid\\_cleanup\\_batch\\_size](#), [gtid\\_ignore\\_duplicates](#), [gtid\\_pos\\_auto\\_engines](#), [gtid\\_slave\\_pos](#), [gtid\\_strict\\_mode](#), [init\\_slave](#), [read\\_binlog\\_speed\\_limit](#), [relay\\_log\\_purge](#), [relay\\_log\\_recovery](#), [replicate\\_do\\_db](#), [replicate\\_do\\_table](#), [replicate\\_events\\_marked\\_for\\_skip](#), [replicate\\_ignore\\_db](#), [replicate\\_ignore\\_table](#), [replicate\\_wild\\_do\\_table](#), [replicate\\_wild\\_ignore\\_table](#), [slave\\_compressed\\_protocol](#), [slave\\_ddl\\_exec\\_mode](#), [slave\\_domain\\_parallel\\_threads](#), [slave\\_exec\\_mode](#), [slave\\_max\\_allowed\\_packet](#), [slave\\_net\\_timeout](#), [slave\\_parallel\\_max\\_queued](#), [slave\\_parallel\\_mode](#), [slave\\_parallel\\_threads](#), [slave\\_parallel\\_workers](#), [slave\\_run\\_triggers\\_for\\_rbr](#), [slave\\_sql\\_verify\\_checksum](#), [slave\\_transaction\\_retry\\_interval](#), [slave\\_type\\_conversions](#), [sync\\_master\\_info](#), [sync\\_relay\\_log](#) and [sync\\_relay\\_log\\_info](#) system variables. Added in

## SET USER

Enables setting the  
DEFINER  
when creating triggers , views , stored functions and stored procedures . Added in MariaDB 10.5.2 .

## SHOW DATABASES

List all databases using the SHOW DATABASES statement. Without the

```
SHOW DATABASES
privilege, you can still issue the
SHOW DATABASES
statement, but it will only list databases containing tables on which you have privileges.
```

## SHUTDOWN

Shut down the server using SHUTDOWN or the mysqladmin shutdown command.

## SUPER

Execute superuser statements: CHANGE MASTER TO , KILL (users who do not have this privilege can only  
KILL  
their own threads), PURGE LOGS , SET global system variables , or the mysqladmin debug command. Also, this permission allows the user to  
write data even if the read\_only startup option is set, enable or disable logging, enable or disable replication on replica, specify a  
DEFINER  
for statements that support that clause, connect once after reaching the  
MAX\_CONNECTIONS  
. If a statement has been specified for the init-connect  
mysqld  
option, that command will not be executed when a user with  
SUPER  
privileges connects to the server.

The SUPER privilege has been split into multiple smaller privileges from MariaDB 10.5.2 to allow for more fine-grained privileges, although it remains an alias for these smaller privileges.

## Database Privileges

The following table lists the privileges that can be granted at the database level. You can also grant all table and function privileges at the database level. Table and function privileges on a database apply to all tables or functions in that database, including those created later.

To set a privilege for a database, specify the database using

```
db_name.*  
for priv_level , or just use  
*  
to specify the default database.
```

Privilege	Description
CREATE	Create a database using the CREATE DATABASE statement, when the privilege is granted for a database. You can grant the CREATE privilege on databases that do not yet exist. This also grants the CREATE privilege on all tables in the database.
CREATE ROUTINE	Create Stored Programs using the CREATE PROCEDURE and CREATE FUNCTION statements.
CREATE TEMPORARY TABLES	Create temporary tables with the CREATE TEMPORARY TABLE statement. This privilege enables writing and dropping those temporary tables
DROP	Drop a database using the DROP DATABASE statement, when the privilege is granted for a database. This also grants the DROP privilege on all tables in the database.

EVENT	Create, drop and alter EVENT S.
GRANT OPTION	Grant database privileges. You can only grant privileges that you have.
LOCK TABLES	Acquire explicit locks using the <a href="#">LOCK TABLES</a> statement; you also need to have the SELECT privilege on a table, in order to lock it.

## Table Privileges

Privilege	Description
ALTER	Change the structure of an existing table using the <a href="#">ALTER TABLE</a> statement.
CREATE	Create a table using the <a href="#">CREATE TABLE</a> statement. You can grant the CREATE privilege on tables that do not yet exist.
CREATE VIEW	Create a view using the <a href="#">CREATE_VIEW</a> statement.
DELETE	Remove rows from a table using the <a href="#">DELETE</a> statement.
DELETE HISTORY	Remove <a href="#">historical rows</a> from a table using the <a href="#">DELETE HISTORY</a> statement. Displays as DELETE VERSIONING ROWS when running <a href="#">SHOW GRANTS</a> until MariaDB 10.3.15 and until MariaDB 10.4.5 ( MDEV-17655 ), or when running <a href="#">SHOW PRIVILEGES</a> until MariaDB 10.5.2 , MariaDB 10.4.13 and MariaDB 10.3.23 ( MDEV-20382 ). From MariaDB 10.3.4 . From MariaDB 10.3.5 , if a user has the SUPER privilege but not this privilege, running <a href="#">mysql_upgrade</a> will grant this privilege as well.
DROP	Drop a table using the <a href="#">DROP TABLE</a> statement or a view using the <a href="#">DROP VIEW</a> statement. Also required to execute the <a href="#">TRUNCATE TABLE</a> statement.
GRANT OPTION	Grant table privileges. You can only grant privileges that you have.
INDEX	Create an index on a table using the <a href="#">CREATE INDEX</a> statement. Without the INDEX privilege, you can still create indexes when creating a table using the <a href="#">CREATE TABLE</a> statement if the you have the CREATE privilege, and you can create indexes using the <a href="#">ALTER TABLE</a> statement if you have the ALTER privilege.
INSERT	Add rows to a table using the <a href="#">INSERT</a> statement. The INSERT privilege can also be set on individual columns; see <a href="#">Column Privileges</a> below for details.
REFERENCES	Unused.

SELECT	Read data from a table using the <a href="#">SELECT</a> statement. The <a href="#">SELECT</a> privilege can also be set on individual columns; see <a href="#">Column Privileges</a> below for details.
SHOW VIEW	Show the <a href="#">CREATE VIEW</a> statement to create a view using the <a href="#">SHOW CREATE VIEW</a> statement.
TRIGGER	Execute triggers associated to tables you update, execute the <a href="#">CREATE TRIGGER</a> and <a href="#">DROP TRIGGER</a> statements. You will still be able to see triggers.
UPDATE	Update existing rows in a table using the <a href="#">UPDATE</a> statement. UPDATE statements usually include a WHERE clause to update only certain rows. You must have SELECT privileges on the table or the appropriate columns for the WHERE clause. The UPDATE privilege can also be set on individual columns; see <a href="#">Column Privileges</a> below for details.

## Column Privileges

Some table privileges can be set for individual columns of a table. To use column privileges, specify the table explicitly and provide a list of column names after the privilege type. For example, the following statement would allow the user to read the names and positions of employees, but not other information from the same table, such as salaries.

```
GRANT SELECT (name, position) ON Employee TO 'jeffrey'@'localhost';
```

Privilege	Description
INSERT (column_list)	Add rows specifying values in columns using the <a href="#">INSERT</a> statement. If you only have column-level INSERT privileges, you must specify the columns you are setting in the <a href="#">INSERT</a> statement. All other columns will be set to their default values, or NULL.
REFERENCES (column_list)	Unused.
SELECT (column_list)	Read values in columns using the <a href="#">SELECT</a> statement. You cannot access or query any columns for which you do not have SELECT privileges, including in WHERE, , ON, , GROUP BY, , and ORDER BY clauses.
UPDATE (column_list)	Update values in columns of existing rows using the <a href="#">UPDATE</a> statement. UPDATE statements usually include a WHERE clause to update only certain rows. You must have SELECT privileges on the table or the appropriate columns for the WHERE clause.

## Function Privileges

Privilege	Description
ALTER ROUTINE	Change the characteristics of a stored function using the <a href="#">ALTER FUNCTION</a> statement.
EXECUTE	Use a stored function. You need SELECT privileges for any tables or columns accessed by the function.
GRANT OPTION	Grant function privileges. You can only grant privileges that you have.

## Procedure Privileges

Privilege	Description
ALTER ROUTINE	Change the characteristics of a stored procedure using the <a href="#">ALTER PROCEDURE</a> statement.
EXECUTE	Execute a <a href="#">stored procedure</a> using the <a href="#">CALL</a> statement. The privilege to call a procedure may allow you to perform actions you wouldn't otherwise be able to do, such as insert rows into a table.
GRANT OPTION	Grant procedure privileges. You can only grant privileges that you have.

## Proxy Privileges

Privilege	Description
PROXY	Permits one user to be a proxy for another.

The

PROXY

privilege allows one user to proxy as another user, which means their privileges change to that of the proxy user, and the [CURRENT\\_USER\(\)](#) function returns the user name of the proxy user.

The

PROXY

privilege only works with authentication plugins that support it. The default [mysql\\_native\\_password](#) authentication plugin does not support proxy users.

The [pam](#) authentication plugin is the only plugin included with MariaDB that currently supports proxy users. The

PROXY

privilege is commonly used with the [pam](#) authentication plugin to enable [user and group mapping with PAM](#).

For example, to grant the

PROXY

privilege to an [anonymous account](#) that authenticates with the [pam](#) authentication plugin, you could execute the following:

```
CREATE USER 'dba'@'%' IDENTIFIED BY 'strongpassword';
GRANT ALL PRIVILEGES ON *.* TO 'dba'@'%';
```

```
CREATE USER ''@'%' IDENTIFIED VIA pam USING 'mariadb';
GRANT PROXY ON 'dba'@'%' TO ''@'%';
```

A user account can only grant the

PROXY

privilege for a specific user account if the grantor also has the

PROXY

privilege for that specific user account, and if that privilege is defined

WITH GRANT OPTION

. For example, the following example fails because the grantee does not have the PROXY privilege for that specific user account at all:

```
SELECT USER(), CURRENT_USER();
+-----+-----+
| USER()          | CURRENT_USER()   |
+-----+-----+
| alice@localhost | alice@localhost |
+-----+-----+


SHOW GRANTS;
+-----+
| Grants for alice@localhost                                |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'alice'@'localhost' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' |
+-----+


GRANT PROXY ON 'dba'@'localhost' TO 'bob'@'localhost';
ERROR 1698 (28000): Access denied for user 'alice'@'localhost'
```

And the following example fails because the grantee does have the

PROXY  
privilege for that specific user account, but it is not defined  
WITH GRANT OPTION

```
SELECT USER(), CURRENT_USER();
+-----+-----+
| USER()          | CURRENT_USER()   |
+-----+-----+
| alice@localhost | alice@localhost |
+-----+-----+



SHOW GRANTS;
+-----+
| Grants for alice@localhost
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'alice'@'localhost' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19'
| GRANT PROXY ON 'dba'@'localhost' TO 'alice'@'localhost'
+-----+



GRANT PROXY ON 'dba'@'localhost' TO 'bob'@'localhost';
ERROR 1698 (28000): Access denied for user 'alice'@'localhost'
```

But the following example succeeds because the grantee does have the

PROXY  
privilege for that specific user account, and it is defined  
WITH GRANT OPTION

```
SELECT USER(), CURRENT_USER();
+-----+-----+
| USER() | CURRENT_USER() |
+-----+-----+
| alice@localhost | alice@localhost |
+-----+-----+


SHOW GRANTS;
+-----+
| Grants for alice@localhost
+
| GRANT ALL PRIVILEGES ON *.* TO 'alice'@'localhost' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' WITH GRANTEE
| GRANT PROXY ON 'dba'@'localhost' TO 'alice'@'localhost' WITH GRANT OPTION
+
| GRANT PROXY ON 'dba'@'localhost' TO 'bob'@'localhost';
```

A user account can grant the

account can grant the  
PROXY  
privilege for any other user account if the grantor has the  
PROXY  
privilege for the

```
' '%'@'
```

anonymous user account, like this:

```
GRANT PROXY ON ''@''%'' TO 'dba'@'localhost' WITH GRANT OPTION;
```

For example, the following example succeeds because the user can grant the PROXY privilege for any other user account:

```
SELECT USER(), CURRENT_USER();
+-----+-----+
| USER() | CURRENT_USER() |
+-----+-----+
| alice@localhost | alice@localhost |
+-----+-----+


SHOW GRANTS;
+-----+
| Grants for alice@localhost
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'alice'@'localhost' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E9' WITH GRANT OPTION
| GRANT PROXY ON ''@''%'' TO 'alice'@'localhost' WITH GRANT OPTION
+-----+


GRANT PROXY ON 'app1_dba'@'localhost' TO 'bob'@'localhost';
Query OK, 0 rows affected (0.004 sec)

GRANT PROXY ON 'app2_dba'@'localhost' TO 'carol'@'localhost';
Query OK, 0 rows affected (0.004 sec)
```

The default

root

user accounts created by `mysql_install_db` have this privilege. For example:

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
GRANT PROXY ON ''@''%'' TO 'root'@'localhost' WITH GRANT OPTION;
```

This allows the default

root

user accounts to grant the

PROXY

privilege for any other user account, and it also allows the default

root

user accounts to grant others the privilege to do the same.

## Authentication Options

The authentication options for the

`GRANT`

statement are the same as those for the `CREATE USER` statement.

### IDENTIFIED BY 'password'

The optional

`IDENTIFIED BY`

clause can be used to provide an account with a password. The password should be specified in plain text. It will be hashed by the `PASSWORD` function prior to being stored to the `mysql.user` table.

For example, if our password is

`mariadb`

, then we can create the user with:

```
GRANT USAGE ON *.* TO foo2@test IDENTIFIED BY 'mariadb';
```

If you do not specify a password with the

`IDENTIFIED BY`

clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

If the user account already exists and if you provide the

IDENTIFIED BY  
clause, then the user's password will be changed. You must have the privileges needed for the [SET PASSWORD](#) statement to change a user's password with  
GRANT

The only [authentication plugins](#) that this clause supports are [mysql\\_native\\_password](#) and [mysql\\_old\\_password](#).

## IDENTIFIED BY PASSWORD 'password\_hash'

The optional

IDENTIFIED BY PASSWORD

clause can be used to provide an account with a password that has already been hashed. The password should be specified as a hash that was provided by the [PASSWORD](#) function. It will be stored to the [mysql.user](#) table as-is.

For example, if our password is

mariadb

, then we can find the hash with:

```
SELECT PASSWORD('mariadb');
+-----+
| PASSWORD('mariadb') |
+-----+
| *54958E764CE10E50764C2EECBB71D01F08549980 |
+-----+
1 row in set (0.00 sec)
```

And then we can create a user with the hash:

```
GRANT USAGE ON *.* TO foo2@test IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECBB71D01F08549980';
```

If you do not specify a password with the

IDENTIFIED BY

clause, the user will be able to connect without a password. A blank password is not a wildcard to match any password. The user must connect without providing a password if no password is set.

If the user account already exists and if you provide the

IDENTIFIED BY

clause, then the user's password will be changed. You must have the privileges needed for the [SET PASSWORD](#) statement to change a user's password with

GRANT

The only [authentication plugins](#) that this clause supports are [mysql\\_native\\_password](#) and [mysql\\_old\\_password](#).

## IDENTIFIED {VIA|WITH} authentication\_plugin

The optional

IDENTIFIED VIA authentication\_plugin

allows you to specify that the account should be authenticated by a specific [authentication plugin](#). The plugin name must be an active authentication plugin as per [SHOW PLUGINS](#). If it doesn't show up in that output, then you will need to install it with [INSTALL PLUGIN](#) or [INSTALL SONAME](#).

For example, this could be used with the [PAM authentication plugin](#):

```
GRANT USAGE ON *.* TO foo2@test IDENTIFIED VIA pam;
```

Some authentication plugins allow additional arguments to be specified after a

USING

or

AS

keyword. For example, the [PAM authentication plugin](#) accepts a [service name](#):

```
GRANT USAGE ON *.* TO foo2@test IDENTIFIED VIA pam USING 'mariadb';
```

The exact meaning of the additional argument would depend on the specific authentication plugin.

MariaDB starting with 10.4.0

The

USING

or

AS

keyword can also be used to provide a plain-text password to a plugin if it's provided as an argument to the `PASSWORD()` function. This is only valid for [authentication plugins](#) that have implemented a hook for the `PASSWORD()` function. For example, the `ed25519` authentication plugin supports this:

```
CREATE USER safe@'%' IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

MariaDB starting with 10.4.3

One can specify many authentication plugins, they all work as alternatives ways of authenticating a user:

```
CREATE USER safe@'%' IDENTIFIED VIA ed25519 USING PASSWORD('secret') OR unix_socket;
```

By default, when you create a user without specifying an authentication plugin, MariaDB uses the `mysql_native_password` plugin.

## Resource Limit Options

MariaDB starting with 10.2.0

[MariaDB 10.2.0](#) introduced a number of resource limit options.

It is possible to set per-account limits for certain server resources. The following table shows the values that can be set per account:

Limit Type	Description
MAX_QUERIES_PER_HOUR	Number of statements that the account can issue per hour (including updates)
MAX_UPDATES_PER_HOUR	Number of updates (not queries) that the account can issue per hour
MAX_CONNECTIONS_PER_HOUR	Number of connections that the account can start per hour
MAX_USER_CONNECTIONS	Number of simultaneous connections that can be accepted from the same account; if it is 0, <code>max_connections</code> will be used instead; if <code>max_connections</code> is 0, there is no limit for this account's simultaneous connections.
MAX_STATEMENT_TIME	Timeout, in seconds, for statements executed by the user. See also <a href="#">Aborting Statements that Exceed a Certain Time to Execute</a> .

If any of these limits are set to

0

, then there is no limit for that resource for that user.

To set resource limits for an account, if you do not want to change that account's privileges, you can issue a

```
GRANT USAGE ON *.* TO 'someone'@'localhost' WITH  
  MAX_USER_CONNECTIONS 0  
  MAX_QUERIES_PER_HOUR 200;
```

The resources are tracked per account, which means

'user'@'server'  
; not per user name or per connection.

The count can be reset for all users using `FLUSH USER_RESOURCES`, `FLUSH PRIVILEGES` or `mysqladmin reload`.

Users with the  
CONNECTION ADMIN

```

privilege (in MariaDB 10.5.2 and later) or the
SUPER
privilege are not restricted by
max_user_connections
,
max_connections
, or
max_password_errors
.

```

Per account resource limits are stored in the `user` table, in the `mysql` database. Columns used for resources limits are named

```

max_questions
,
max_updates
,
max_connections
(for
MAX_CONNECTIONS_PER_HOUR
), and
max_user_connections
(for
MAX_USER_CONNECTIONS
).

```

## TLS Options

By default, MariaDB transmits data between the server and clients without encrypting it. This is generally acceptable when the server and client run on the same host or in networks where security is guaranteed through other means. However, in cases where the server and client exist on separate networks or they are in a high-risk network, the lack of encryption does introduce security concerns as a malicious actor could potentially eavesdrop on the traffic as it is sent over the network between them.

To mitigate this concern, MariaDB allows you to encrypt data in transit between the server and clients using the Transport Layer Security (TLS) protocol. TLS was formerly known as Secure Socket Layer (SSL), but strictly speaking the SSL protocol is a predecessor to TLS and, that version of the protocol is now considered insecure. The documentation still uses the term SSL often and for compatibility reasons TLS-related server system and status variables still use the prefix `ssl_`, but internally, MariaDB only supports its secure successors.

See [Secure Connections Overview](#) for more information about how to determine whether your MariaDB server has TLS support.

You can set certain TLS-related restrictions for specific user accounts. For instance, you might use this with user accounts that require access to sensitive data while sending it across networks that you do not control. These restrictions can be enabled for a user account with the `CREATE USER`, `ALTER USER`, or `GRANT` statements. The following options are available:

Option	Description
REQUIRE NONE	TLS is not required for this account, but can still be used.
REQUIRE SSL	The account must use TLS, but no valid X509 certificate is required. This option cannot be combined with other TLS options.
REQUIRE X509	The account must use TLS and must have a valid X509 certificate. This option implies REQUIRE SSL . This option cannot be combined with other TLS options.
'issuer' REQUIRE ISSUER	The account must use TLS and must have a valid X509 certificate. Also, the Certificate Authority must be the one specified via the string issuer . This option implies REQUIRE X509 . This option can be combined with the SUBJECT , and CIPHER options in any order.

<pre>REQUIRE SUBJECT 'subject'</pre>	<p>The account must use TLS and must have a valid X509 certificate. Also, the certificate's Subject must be the one specified via the string</p> <pre>    subject     . This option implies REQUIRE X509     . This option can be combined with the ISSUER , and CIPHER options in any order.</pre>
<pre>REQUIRE CIPHER 'cipher'</pre>	<p>The account must use TLS, but no valid X509 certificate is required. Also, the encryption used for the connection must use a specific cipher method specified in the string</p> <pre>    cipher     . This option implies REQUIRE SSL     . This option can be combined with the ISSUER , and SUBJECT options in any order.</pre>

The

```
REQUIRE
keyword must be used only once for all specified options, and the
AND
keyword can be used to separate individual options, but it is not required.
```

For example, you can create a user account that requires these TLS options with the following:

```
GRANT USAGE ON *.* TO 'alice'@'%'
REQUIRE SUBJECT '/CN=alice/O=My Dom, Inc./C=US/ST=Oregon/L=Portland'
AND ISSUER '/C=FI/ST=Somewhere/L=City/ O=Some Company/CN=Peter Parker/emailAddress=p.parker@marvel.com'
AND CIPHER 'SHA-DES-CBC3-EDH-RSA';
```

If any of these options are set for a specific user account, then any client who tries to connect with that user account will have to be configured to connect with TLS.

See [Securing Connections for Client and Server](#) for information on how to enable TLS on the client and server.

## Roles

### Syntax

```
GRANT role TO grantee [, grantee ... ]
[ WITH ADMIN OPTION ]

grantee:
  rolename
  username [authentication_option]
```

The GRANT statement is also used to grant the use a [role](#) to one or more users or other roles. In order to be able to grant a role, the grantor doing so must have permission to do so (see WITH ADMIN in the [CREATE ROLE](#) article).

Specifying the

```
WITH ADMIN OPTION
permits the grantee to in turn grant the role to another.
```

For example, the following commands show how to grant the same role to a couple different users.

```
GRANT journalist TO hulda;
GRANT journalist TO berengar WITH ADMIN OPTION;
```

If a user has been granted a role, they do not automatically obtain all permissions associated with that role. These permissions are only in use when the user activates the role with the [SET ROLE](#) statement.

## Grant Examples

### Granting Root-like Privileges

You can create a user that has privileges similar to the default

root

accounts by executing the following:

```
CREATE USER 'alexander'@'localhost';
GRANT ALL PRIVILEGES ON *.* TO 'alexander'@'localhost' WITH GRANT OPTION;
```

## See Also

- [Troubleshooting Connection Issues](#)
- [--skip-grant-tables](#) allows you to start MariaDB without  
GRANT  
. This is useful if you lost your root password.
- [CREATE USER](#)
- [ALTER USER](#)
- [DROP USER](#)
- [SET PASSWORD](#)
- [SHOW CREATE USER](#)
- [mysql.user table](#)
- [Password Validation Plugins](#) - permits the setting of basic criteria for passwords
- [Authentication Plugins](#) - allow various authentication methods to be used, and new ones to be developed.

## 1.1.1.5 RENAME USER

### Syntax

```
RENAME USER old_user TO new_user
[, old_user TO new_user] ...
```

### Description

The RENAME USER statement renames existing MariaDB accounts. To use it, you must have the global

[CREATE\\_USER](#)

privilege or the

[UPDATE](#)

privilege for the

[mysql](#)

database. Each account is named using the same format as for the

[CREATE\\_USER](#)

statement; for example,

'jeffrey'@'localhost'

. If you specify only the user name part of the account name, a host name part of

'%'

is used.

If any of the old user accounts do not exist or any of the new user accounts already exist,

[ERROR 1396 \(HY000\)](#)

results. If an error occurs,

[RENAME\\_USER](#)

will still rename the accounts that do not result in an error.

### Examples

```
CREATE USER 'donald', 'mickey';
RENAME USER 'donald' TO 'duck'@'localhost', 'mickey' TO 'mouse'@'localhost';
```

## 1.1.1.6 REVOKE

## Contents

- 1. Privileges
  - 1. Syntax
  - 2. Description
  - 3. Examples
- 2. Roles
  - 1. Syntax
  - 2. Description
  - 3. Example

# Privileges

## Syntax

```
REVOKE  
    priv_type [(column_list)]  
    [, priv_type [(column_list)]] ...  
ON [object_type] priv_level  
FROM user [, user] ...  
  
REVOKE ALL PRIVILEGES, GRANT OPTION  
    FROM user [, user] ...
```

## Description

The

```
REVOKE
```

statement enables system administrators to revoke privileges (or roles - see [section below](#)) from MariaDB accounts. Each account is named using the same format as for the

```
GRANT
```

statement; for example,

```
jeffrey'@'localhost
```

'. If you specify only the user name part of the account name, a host name part of '

```
%
```

' is used. For details on the levels at which privileges exist, the allowable

```
priv_type
```

and

```
priv_level
```

values, and the syntax for specifying users and passwords, see [GRANT](#).

To use the first

```
REVOKE
```

syntax, you must have the

```
GRANT OPTION
```

privilege, and you must have the privileges that you are revoking.

To revoke all privileges, use the second syntax, which drops all global, database, table, column, and routine privileges for the named user or users:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

To use this

```
REVOKE
```

syntax, you must have the global [CREATE USER](#) privilege or the [UPDATE](#) privilege for the mysql database. See [GRANT](#).

## Examples

```
REVOKE SUPER ON *.* FROM 'alexander'@'localhost';
```

# Roles

## Syntax

```
REVOKE role [, role ...]
  FROM grantee [, grantee2 ... ]

REVOKE ADMIN OPTION FOR role FROM grantee [, grantee2]
```

## Description

REVOKE

is also used to remove a [role](#) from a user or another role that it's previously been assigned to. If a role has previously been set as a [default role](#), REVOKE

does not remove the record of the default role from the [mysql.user](#) table. If the role is subsequently granted again, it will again be the user's default. Use [SET DEFAULT ROLE NONE](#) to explicitly remove this.

Before [MariaDB 10.1.13](#), the

```
REVOKE role
statement was not permitted in prepared statements.
```

## Example

```
REVOKE journalist FROM hulda
```

## 1.1.1.7 SET PASSWORD

### Syntax

```
SET PASSWORD [FOR user] =
{
  PASSWORD('some password')
  | OLD_PASSWORD('some password')
  | 'encrypted password'
}
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Authentication Plugin Support](#)
4. [Passwordless User Accounts](#)
5. [Example](#)
6. [See Also](#)

## Description

The

```
SET PASSWORD
```

statement assigns a password to an existing MariaDB user account.

If the password is specified using the

```
PASSWORD()
```

or

```
OLD_PASSWORD()
```

function, the literal text of the password should be given. If the password is specified without using either function, the password should be the already-encrypted password value as returned by

```
PASSWORD()
```

```
OLD_PASSWORD()
```

should only be used if your MariaDB/MySQL clients are very old (< 4.0.0).

With no

FOR

clause, this statement sets the password for the current user. Any client that has connected to the server using a non-anonymous account can change the password for that account.

With a

FOR

clause, this statement sets the password for a specific account on the current server host. Only clients that have the

UPDATE

privilege for the

mysql

database can do this. The user value should be given in

user\_name@host\_name

format, where

user\_name

and

host\_name

are exactly as they are listed in the User and Host columns of the

mysql.user

table entry.

The argument to

PASSWORD()

and the password given to MariaDB clients can be of arbitrary length.

## Authentication Plugin Support

MariaDB starting with 10.4

In MariaDB 10.4 and later,

SET PASSWORD

(with or without

PASSWORD()

) works for accounts authenticated via any [authentication plugin](#) that supports passwords stored in the

mysql.global\_priv

table.

The

ed25519

,

mysql\_native\_password

, and

mysql\_old\_password

authentication plugins store passwords in the

mysql.global\_priv

table.

If you run

SET PASSWORD

on an account that authenticates with one of these authentication plugins that stores passwords in the

mysql.global\_priv

table, then the

PASSWORD()

function is evaluated by the specific authentication plugin used by the account. The authentication plugin hashes the password with a method that is compatible with that specific authentication plugin.

The

`unix_socket`  
,

`named_pipe`  
,

`gssapi`

, and

`pam`

authentication plugins do **not** store passwords in the

`mysql.global_priv`

table. These authentication plugins rely on other methods to authenticate the user.

If you attempt to run

`SET PASSWORD`

on an account that authenticates with one of these authentication plugins that doesn't store a password in the

`mysql.global_priv`

table, then MariaDB Server will raise a warning like the following:

```
SET PASSWORD is ignored for users authenticating via unix_socket plugin
```

See [Authentication from MariaDB 10.4](#) for an overview of authentication changes in [MariaDB 10.4](#).

#### MariaDB until 10.3

In [MariaDB 10.3](#) and before,

`SET PASSWORD`

(with or without

`PASSWORD()`

) only works for accounts authenticated via

`mysql_native_password`

or

`mysql_old_password`

authentication plugins

## Passwordless User Accounts

User accounts do not always require passwords to login.

The

`unix_socket`

,

`named_pipe`

and

`gssapi`

authentication plugins do not require a password to authenticate the user.

The

`pam`

authentication plugin may or may not require a password to authenticate the user, depending on the specific configuration.

The

`mysql_native_password`

and

`mysql_old_password`

authentication plugins require passwords for authentication, but the password can be blank. In that case, no password is required.

If you provide a password while attempting to log into the server as an account that doesn't require a password, then MariaDB server will simply ignore the password.

MariaDB starting with 10.4

In MariaDB 10.4 and later, a user account can be defined to use multiple authentication plugins in a specific order of preference. This specific scenario may be more noticeable in these versions, since an account could be associated with some authentication plugins that require a password, and some that do not.

## Example

For example, if you had an entry with User and Host column values of '

bob

' and '

.loc.gov

', you would write the statement like this:

```
SET PASSWORD FOR 'bob'@'%loc.gov' = PASSWORD('newpass');
```

If you want to delete a password for a user, you would do:

```
SET PASSWORD FOR 'bob'@localhost = PASSWORD("");
```

## See Also

- [Password Validation Plugins](#) - permits the setting of basic criteria for passwords
- 

[ALTER USER](#)

## 1.1.1.8 CREATE ROLE

### Syntax

```
CREATE [OR REPLACE] ROLE [IF NOT EXISTS] role  
[WITH ADMIN  
{CURRENT_USER | CURRENT_ROLE | user | role}]
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [WITH ADMIN](#)
  2. [OR REPLACE](#)
  3. [IF NOT EXISTS](#)
3. [Examples](#)
4. [See Also](#)

### Description

The

`CREATE ROLE`

statement creates one or more MariaDB [roles](#). To use it, you must have the global [CREATE USER](#) privilege or the [INSERT](#) privilege for the

mysql database. For each account,

```
CREATE ROLE  
creates a new row in the mysql.user table that has no privileges, and with the corresponding  
is_role  
field set to  
Y  
. It also creates a record in the mysql.roles_mapping table.
```

If any of the specified roles already exist,

```
ERROR 1396 (HY000)  
results. If an error occurs,  
CREATE ROLE  
will still create the roles that do not result in an error. The maximum length for a role is 128 characters. Role names can be quoted, as explained  
in the Identifier names page. Only one error is produced for all roles which have not been created:
```

```
ERROR 1396 (HY000): Operation CREATE ROLE failed for 'a','b','c'
```

Failed

```
CREATE  
or  
DROP  
operations, for both users and roles, produce the same error code.
```

```
PUBLIC  
and  
NONE  
are reserved, and cannot be used as role names.  
NONE  
is used to unset a role and  
PUBLIC  
has a special use in other systems, such as Oracle, so is reserved for compatibility purposes.
```

Before MariaDB 10.1.13 , the

```
CREATE ROLE  
statement was not permitted in prepared statements .
```

For valid identifiers to use as role names, see Identifier Names .

## WITH ADMIN

The optional

```
WITH ADMIN  
clause determines whether the current user, the current role or another user or role has use of the newly created role. If the clause is omitted,  
WITH ADMIN CURRENT_USER  
is treated as the default, which means that the current user will be able to GRANT this role to users.
```

## OR REPLACE

If the optional

```
OR REPLACE  
clause is used, it acts as a shortcut for:
```

```
DROP ROLE IF EXISTS name;  
CREATE ROLE name ...;
```

## IF NOT EXISTS

When the

```
IF NOT EXISTS  
clause is used, MariaDB will return a warning instead of an error if the specified role already exists. Cannot be used together with the  
OR REPLACE  
clause.
```

# Examples

```
CREATE ROLE journalist;  
  
CREATE ROLE developer WITH ADMIN lorinda@localhost;
```

Granting the role to another user. Only user

lorinda@localhost  
has permission to grant the  
developer  
role:

```
SELECT USER();
+-----+
| USER() |
+-----+
| henning@localhost |
+-----+
...
GRANT developer TO ian@localhost;
Access denied for user 'henning'@'localhost'

SELECT USER();
+-----+
| USER() |
+-----+
| lorinda@localhost |
+-----+

GRANT m_role TO ian@localhost;
```

The

OR REPLACE  
and  
IF NOT EXISTS  
clauses. The  
journalist  
role already exists:

```
CREATE ROLE journalist;
ERROR 1396 (HY000): Operation CREATE ROLE failed for 'journalist'

CREATE OR REPLACE ROLE journalist;
Query OK, 0 rows affected (0.00 sec)

CREATE ROLE IF NOT EXISTS journalist;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1975 | Can't create role 'journalist'; it already exists |
+-----+-----+
```

## See Also

- [Identifier Names](#)
- [Roles Overview](#)
- [DROP ROLE](#)

### 1.1.1.9 DROP ROLE

#### Syntax

```
DROP ROLE [IF EXISTS] role_name [,role_name ...]
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [IF EXISTS](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

```
DROP ROLE
```

statement removes one or more MariaDB [roles](#). To use this statement, you must have the global [CREATE USER](#) privilege or the [DELETE](#) privilege for the `mysql` database.

```
DROP ROLE
```

does not disable roles for connections which selected them with [SET ROLE](#). If a role has previously been set as a [default role](#),

```
DROP ROLE
```

does not remove the record of the default role from the `mysql.user` table. If the role is subsequently recreated and granted, it will again be the user's default. Use [SET DEFAULT ROLE NONE](#) to explicitly remove this.

If any of the specified user accounts do not exist,

```
ERROR 1396 (HY000)
```

results. If an error occurs,

```
DROP ROLE
```

will still drop the roles that do not result in an error. Only one error is produced for all roles which have not been dropped:

```
ERROR 1396 (HY000): Operation DROP ROLE failed for 'a','b','c'
```

Failed

```
CREATE
```

or

```
DROP
```

operations, for both users and roles, produce the same error code.

Before [MariaDB 10.1.13](#), the

```
DROP ROLE
```

statement was not permitted in [prepared statements](#).

## IF EXISTS

If the

```
IF EXISTS
```

clause is used, MariaDB will return a warning instead of an error if the role does not exist.

## Examples

```
DROP ROLE journalist;
```

The same thing using the optional

```
IF EXISTS
```

clause:

```
DROP ROLE journalist;
ERROR 1396 (HY000): Operation DROP ROLE failed for 'journalist'
```

```
DROP ROLE IF EXISTS journalist;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
Note (Code 1975): Can't drop role 'journalist'; it doesn't exist
```

## See Also

- [Roles Overview](#)
- [CREATE ROLE](#)

## 1.1.1.10 SET ROLE

# Syntax

```
SET ROLE { role | NONE }
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)

## Description

The

```
SET ROLE
```

statement enables a [role](#), along with all of its associated permissions, for the current session. To unset a role, use [NONE](#).

If a role that doesn't exist, or to which the user has not been assigned, is specified, an

```
ERROR 1959 (OP000): Invalid role specification
```

error occurs.

An automatic SET ROLE is implicitly performed when a user connects if that user has been assigned a default role. See [SET DEFAULT ROLE](#).

## Example

```
SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |
+-----+
| NULL         |
+-----+

SET ROLE staff;

SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |
+-----+
| staff        |
+-----+

SET ROLE NONE;

SELECT CURRENT_ROLE();
+-----+
| CURRENT_ROLE() |
+-----+
| NULL          |
+-----+
```

## 1.1.1.11 SET DEFAULT ROLE

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

## Syntax

```
SET DEFAULT ROLE { role | NONE } [ FOR user@host ]
```

## Description

The

```
SET DEFAULT ROLE
```

statement sets a [default role](#) for a specified (or current) user. A default role is automatically enabled when a user connects (an implicit [SET ROLE](#) statement is executed immediately after a connection is established).

To be able to set a role as a default, the role must already have been granted to that user, and one needs the privileges to enable this role (if you cannot do

```
SET ROLE X
, you won't be able to do
SET DEFAULT ROLE X
). To set a default role for another user one needs to have write access to the
mysql
database.
```

To remove a user's default role, use

```
SET DEFAULT ROLE NONE [ FOR user@host ]
```

. The record of the default role is not removed if the role is [dropped](#) or [revoked](#), so if the role is subsequently re-created or granted, it will again be the user's default role.

The default role is stored in the

```
default_role
column in the mysql.user table/view, as well as in the Information Schema APPLICABLE\_ROLES table, so these can be viewed to see which role has been assigned to a user as the default.
```

## Examples

Setting a default role for the current user:

```
SET DEFAULT ROLE journalist;
```

Removing a default role from the current user:

```
SET DEFAULT ROLE NONE;
```

Setting a default role for another user. The role has to have been granted to the user before it can be set as default:

```
CREATE ROLE journalist;
CREATE USER taniel;

SET DEFAULT ROLE journalist FOR taniel;
ERROR 1959 (OP000): Invalid role specification `journalist'

GRANT journalist TO taniel;
SET DEFAULT ROLE journalist FOR taniel;
```

Viewing mysql.user:

```
select * from mysql.user where user='taniel'\G
***** 1. row *****
      Host: %
      User: taniel
...
      is_role: N
      default_role: journalist
...
```

Removing a default role for another user

```
SET DEFAULT ROLE NONE FOR taniel;
```

## 1.1.1.12 SHOW GRANTS

### Contents

- 1. [Syntax](#)
- 2. [Description](#)
  - 1. [Users](#)
  - 2. [Roles](#)
    - 1. [Example](#)
- 3. [See Also](#)

### Syntax

```
SHOW GRANTS [FOR user|role]
```

# Description

The

```
SHOW GRANTS  
statement lists privileges granted to a particular user or role.
```

## Users

The statement lists the [GRANT](#) statement or statements that must be issued to duplicate the privileges that are granted to a MariaDB user account. The account is named using the same format as for the

```
GRANT  
statement; for example, '  
jeffrey'@'localhost'  
. If you specify only the user name part of the account name, a host name part of '  
' is used. For additional information about specifying account names, see GRANT.
```

```
SHOW GRANTS FOR 'root'@'localhost';  
+-----+  
| Grants for root@localhost |  
+-----+  
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |  
+-----+
```

To list the privileges granted to the account that you are using to connect to the server, you can use any of the following statements:

```
SHOW GRANTS;  
SHOW GRANTS FOR CURRENT_USER;  
SHOW GRANTS FOR CURRENT_USER();
```

If

```
SHOW GRANTS FOR CURRENT_USER  
(or any of the equivalent syntaxes) is used in  
DEFINER  
context (such as within a stored procedure that is defined with  
SQL SECURITY DEFINER  
, the grants displayed are those of the definer and not the invoker.
```

Note that the

```
DELETE HISTORY  
privilege, introduced in MariaDB 10.3.4, was displayed as  
DELETE VERSIONING ROWS  
when running  
SHOW GRANTS  
until MariaDB 10.3.15 (MDEV-17655).
```

## Roles

```
SHOW GRANTS  
can also be used to view the privileges granted to a role.
```

### Example

```
SHOW GRANTS FOR journalist;  
+-----+  
| Grants for journalist |  
+-----+  
| GRANT USAGE ON *.* TO 'journalist' |  
| GRANT DELETE ON `test`.* TO 'journalist' |  
+-----+
```

## See Also

- [Authentication from MariaDB 10.4](#)
- [SHOW CREATE USER](#) shows how the user was created.
- [SHOW PRIVILEGES](#) shows the privileges supported by MariaDB.
- [Roles](#)

## 1.1.1.13 SHOW CREATE USER

MariaDB starting with 10.2.0

```
SHOW CREATE USER  
was introduced in MariaDB 10.2.0
```

### Syntax

```
SHOW CREATE USER user_name
```

### Description

Shows the [CREATE USER](#) statement that created the given user. The statement requires the [SELECT](#) privilege for the [mysql](#) database, except for the current user.

### Examples

```
CREATE USER foo4@test require cipher 'text'  
    issuer 'foo_issuer' subject 'foo_subject';  
  
SHOW CREATE USER foo4@testG  
*****  
***** 1. row *****  
CREATE USER 'foo4'@'test'  
  REQUIRE ISSUER 'foo_issuer'  
  SUBJECT 'foo_subject'  
  CIPHER 'text'
```

User Password Expiry :

```
CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY;  
  
SHOW CREATE USER 'monty'@'localhost';  
+-----+  
| CREATE USER for monty@localhost |  
+-----+  
| CREATE USER 'monty'@'localhost' PASSWORD EXPIRE INTERVAL 120 DAY |  
+-----+
```

### See Also

- [CREATE USER](#)
- [ALTER USER](#)
- [SHOW GRANTS](#) shows the GRANTS/PRIVILEGES for a user.
- [SHOW PRIVILEGES](#) shows the privileges supported by MariaDB.

## 1.1.2 Administrative SQL Statements

### 1.1.2.1 Table Statements

#### 1.1.2.1.1 ALTER

##### 1.1.2.1.1.1 ALTER TABLE

### Syntax

```
ALTER [ONLINE] [IGNORE] TABLE [IF EXISTS] tbl_name  
    [WAIT n | NOWAIT]  
    alter_specification [, alter_specification] ...
```

```

alter_specification:
  table_option ...
  | ADD [COLUMN] [IF NOT EXISTS] col_name column_definition
    [FIRST | AFTER col_name ]
  | ADD [COLUMN] [IF NOT EXISTS] (col_name column_definition,...)
  | ADD {INDEX|KEY} [IF NOT EXISTS] [index_name]
    [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY
    [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
    UNIQUE [INDEX|KEY] [index_name]
    [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [index_name]
    (index_col_name,...) [index_option] ...
  | ADD SPATIAL [INDEX|KEY] [index_name]
    (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
    FOREIGN KEY [IF NOT EXISTS] [index_name] (index_col_name,...)
    reference_definition
  | ADD PERIOD FOR SYSTEM_TIME (start_column_name, end_column_name)
  | ALTER [COLUMN] col_name SET DEFAULT literal | (expression)
  | ALTER [COLUMN] col_name DROP DEFAULT
  | ALTER {INDEX|KEY} index_name [NOT] INVISIBLE
  | CHANGE [COLUMN] [IF EXISTS] old_col_name new_col_name column_definition
    [FIRST|AFTER col_name]
  | MODIFY [COLUMN] [IF EXISTS] col_name column_definition
    [FIRST | AFTER col_name]
  | DROP [COLUMN] [IF EXISTS] col_name [RESTRICT|CASCADE]
  | DROP PRIMARY KEY
  | DROP {INDEX|KEY} [IF EXISTS] index_name
  | DROP FOREIGN KEY [IF EXISTS] fk_symbol
  | DROP CONSTRAINT [IF EXISTS] constraint_name
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col_name [, col_name] ...
  | RENAME COLUMN old_col_name TO new_col_name
  | RENAME {INDEX|KEY} old_index_name TO new_index_name
  | CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
  | [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | ALGORITHM [=] {DEFAULT|INPLACE|COPY|NOCOPY|INSTANT}
  | LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
  | FORCE
  | partition_options
  | ADD PARTITION [IF NOT EXISTS] (partition_definition)
  | DROP PARTITION [IF EXISTS] partition_names
  | COALESCE PARTITION number
  | REORGANIZE PARTITION [partition_names INTO (partition_definitions)]
  | ANALYZE PARTITION partition_names
  | CHECK PARTITION partition_names
  | OPTIMIZE PARTITION partition_names
  | REBUILD PARTITION partition_names
  | REPAIR PARTITION partition_names
  | EXCHANGE PARTITION partition_name WITH TABLE tbl_name
  | REMOVE PARTITIONING
  | ADD SYSTEM VERSIONING
  | DROP SYSTEM VERSIONING

index_col_name:
  col_name [(length)] [ASC | DESC]

index_type:
  USING {BTREE | HASH | RTREE}

index_option:
  [ KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | CLUSTERING={YES| NO} ]

```

```
[ IGNORED | NOT IGNORED ]
```

```
table_options:  
    table_option [ [, ] table_option] ...
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Privileges](#)
4. [Online DDL](#)
  1. [ALTER ONLINE TABLE](#)
5. [WAIT/NOWAIT](#)
6. [IF EXISTS](#)
7. [Column Definitions](#)
8. [Index Definitions](#)
9. [Character Sets and Collations](#)
10. [Alter Specifications](#)
  1. [Table Options](#)
  2. [ADD COLUMN](#)
  3. [DROP COLUMN](#)
  4. [MODIFY COLUMN](#)
  5. [CHANGE COLUMN](#)
  6. [ALTER COLUMN](#)
  7. [RENAME INDEX/KEY](#)
  8. [RENAME COLUMN](#)
  9. [ADD PRIMARY KEY](#)
  10. [DROP PRIMARY KEY](#)
  11. [ADD FOREIGN KEY](#)
  12. [DROP FOREIGN KEY](#)
  13. [ADD INDEX](#)
  14. [DROP INDEX](#)
  15. [ADD UNIQUE INDEX](#)
  16. [DROP UNIQUE INDEX](#)
  17. [ADD FULLTEXT INDEX](#)
  18. [DROP FULLTEXT INDEX](#)
  19. [ADD SPATIAL INDEX](#)
  20. [DROP SPATIAL INDEX](#)
  21. [ENABLE/ DISABLE KEYS](#)
  22. [RENAME TO](#)
  23. [ADD CONSTRAINT](#)
  24. [DROP CONSTRAINT](#)
  25. [ADD SYSTEM VERSIONING](#)
  26. [DROP SYSTEM VERSIONING](#)
  27. [ADD PERIOD FOR SYSTEM\\_TIME](#)
  28. [FORCE](#)
  29. [EXCHANGE PARTITION](#)
  30. [DISCARD TABLESPACE](#)
  31. [IMPORT TABLESPACE](#)
  32. [ALGORITHM](#)
    1. [ALGORITHM=DEFAULT](#)
    2. [ALGORITHM=COPY](#)
    3. [ALGORITHM=INPLACE](#)
    4. [ALGORITHM=NOCOPY](#)
    5. [ALGORITHM=INSTANT](#)
  33. [LOCK](#)
11. [Progress Reporting](#)
12. [Aborting ALTER TABLE Operations](#)
13. [Atomic ALTER TABLE](#)
14. [Replication](#)
15. [Examples](#)
16. [See Also](#)

## Description

### ALTER TABLE

enables you to change the structure of an existing table. For example, you can add or delete columns, create or destroy indexes, change the type of existing columns, or rename columns or the table itself. You can also change the comment for the table and the storage engine of the table.

If another connection is using the table, a [metadata lock](#) is active, and this statement will wait until the lock is released. This is also true for non-transactional tables.

When adding a  
UNIQUE  
index on a column (or a set of columns) which have duplicated values, an error will be produced and the statement will be stopped. To suppress the error and force the creation of  
UNIQUE  
indexes, discarding duplicates, the [IGNORE](#) option can be specified. This can be useful if a column (or a set of columns) should be UNIQUE but it contains duplicate values; however, this technique provides no control on which rows are preserved and which are deleted. Also, note that  
IGNORE  
is accepted but ignored in  
ALTER TABLE ... EXCHANGE PARTITION  
statements.

This statement can also be used to rename a table. For details see [RENAME TABLE](#).

When an index is created, the storage engine may use a configurable buffer in the process. Incrementing the buffer speeds up the index creation. [Aria](#) and [MyISAM](#) allocate a buffer whose size is defined by [aria\\_sort\\_buffer\\_size](#) or [myisam\\_sort\\_buffer\\_size](#), also used for [REPAIR TABLE](#). [InnoDB](#) allocates three buffers whose size is defined by [innodb\\_sort\\_buffer\\_size](#).

## Privileges

Executing the

```
ALTER TABLE
```

statement generally requires at least the [ALTER](#) privilege for the table or the database..

If you are renaming a table, then it also requires the [DROP](#), [CREATE](#) and [INSERT](#) privileges for the table or the database as well.

## Online DDL

Online DDL is supported with the [ALGORITHM](#) and [LOCK](#) clauses.

See [InnoDB Online DDL Overview](#) for more information on online DDL with [InnoDB](#).

### ALTER ONLINE TABLE

ALTER ONLINE TABLE also works for partitioned tables.

Online

```
ALTER TABLE
```

is available by executing the following:

```
ALTER ONLINE TABLE ...;
```

This statement has the following semantics:

This statement is equivalent to the following:

```
ALTER TABLE ... LOCK=NONE;
```

See the [LOCK](#) alter specification for more information. <>

This statement is equivalent to the following:

```
ALTER TABLE ... ALGORITHM=INPLACE;
```

See the [ALGORITHM](#) alter specification for more information. <>

## WAIT/NOWAIT

MariaDB starting with 10.3.0

Set the lock wait timeout. See [WAIT](#) and [NOWAIT](#).

## IF EXISTS

The

```
IF EXISTS
```

and

```
IF NOT EXISTS
```

clauses are available for the following:

```
ADD COLUMN      [IF NOT EXISTS]
ADD INDEX       [IF NOT EXISTS]
ADD FOREIGN KEY [IF NOT EXISTS]
ADD PARTITION   [IF NOT EXISTS]
CREATE INDEX    [IF NOT EXISTS]

DROP COLUMN     [IF EXISTS]
DROP INDEX      [IF EXISTS]
DROP FOREIGN KEY [IF EXISTS]
DROP PARTITION   [IF EXISTS]
CHANGE COLUMN   [IF EXISTS]
MODIFY COLUMN   [IF EXISTS]
DROP INDEX      [IF EXISTS]
```

When

IF EXISTS  
and  
IF NOT EXISTS

are used in clauses, queries will not report errors when the condition is triggered for that clause. A warning with the same message text will be issued and the ALTER will move on to the next clause in the statement (or end if finished). <>/product>>

MariaDB starting with 10.5.2

If this is directive is used after

```
ALTER ... TABLE
, one will not get an error if the table doesn't exist.
```

## Column Definitions

See [CREATE TABLE: Column Definitions](#) for information about column definitions.

## Index Definitions

See [CREATE TABLE: Index Definitions](#) for information about index definitions.

The [CREATE INDEX](#) and [DROP INDEX](#) statements can also be used to add or remove an index.

## Character Sets and Collations

```
CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
[DEFAULT] CHARACTER SET [=] charset_name
[DEFAULT] COLLATE [=] collation_name
```

See [Setting Character Sets and Collations](#) for details on setting the character sets and collations .

## Alter Specifications

### Table Options

See [CREATE TABLE: Table Options](#) for information about table options.

### ADD COLUMN

```
... ADD COLUMN [IF NOT EXISTS] (col_name column_definition,...)
```

Adds a column to the table. The syntax is the same as in [CREATE TABLE](#) . If you are using

IF NOT\_EXISTS  
the column will not be added if it was not there already. This is very useful when doing scripts to modify tables.

The

FIRST  
and  
AFTER  
clauses affect the physical order of columns in the datafile. Use  
FIRST

to add a column in the first (leftmost) position, or

AFTER

followed by a column name to add the new column in any other position. Note that, nowadays, the physical position of a column is usually irrelevant.

See also [Instant ADD COLUMN for InnoDB](#).

## DROP COLUMN

```
... DROP COLUMN [IF EXISTS] col_name [CASCADE|RESTRICT]
```

Drops the column from the table. If you are using

IF EXISTS

you will not get an error if the column didn't exist. If the column is part of any index, the column will be dropped from them, except if you add a new column with identical name at the same time. The index will be dropped if all columns from the index were dropped. If the column was used in a view or trigger, you will get an error next time the view or trigger is accessed.

MariaDB starting with 10.2.8

Dropping a column that is part of a multi-column

UNIQUE

constraint is not permitted. For example:

```
CREATE TABLE a (
    a int,
    b int,
    primary key (a,b)
);

ALTER TABLE x DROP COLUMN a;
[42000][1072] Key column 'A' doesn't exist in table
```

The reason is that dropping column

a

would result in the new constraint that all values in column

b

be unique. In order to drop the column, an explicit

DROP PRIMARY KEY

and

ADD PRIMARY KEY

would be required. Up until [MariaDB 10.2.7](#), the column was dropped and the additional constraint applied, resulting in the following structure:

```
ALTER TABLE x DROP COLUMN a;
Query OK, 0 rows affected (0.46 sec)
```

```
DESC x;
```

Field	Type	Null	Key	Default	Extra
b	int(11)	NO	PRI	NULL	

MariaDB starting with 10.4.0

[MariaDB 10.4.0](#) supports instant DROP COLUMN. DROP COLUMN of an indexed column would imply [DROP INDEX](#) (and in the case of a non-UNIQUE multi-column index, possibly ADD INDEX). These will not be allowed with [ALGORITHM=INSTANT](#), but unlike before, they can be allowed with [ALGORITHM=NOCOPY](#)

RESTRICT

and

CASCADE

are allowed to make porting from other database systems easier. In MariaDB, they do nothing.

## MODIFY COLUMN

Allows you to modify the type of a column. The column will be at the same place as the original column and all indexes on the column will be kept. Note that when modifying column, you should specify all attributes for the new column.

```
CREATE TABLE t1 (a INT UNSIGNED AUTO_INCREMENT, PRIMARY KEY((a));
ALTER TABLE t1 MODIFY a BIGINT UNSIGNED AUTO_INCREMENT;
```

## CHANGE COLUMN

Works like

MODIFY COLUMN

except that you can also change the name of the column. The column will be at the same place as the original column and all index on the column will be kept.

```
CREATE TABLE t1 (a INT UNSIGNED AUTO_INCREMENT, PRIMARY KEY(a));
ALTER TABLE t1 CHANGE a b BIGINT UNSIGNED AUTO_INCREMENT;
```

## ALTER COLUMN

This lets you change column options.

```
CREATE TABLE t1 (a INT UNSIGNED AUTO_INCREMENT, b varchar(50), PRIMARY KEY(a));
ALTER TABLE t1 ALTER b SET DEFAULT 'hello';
```

## RENAME INDEX/KEY

MariaDB starting with 10.5.2

From MariaDB 10.5.2 , it is possible to rename an index using the

RENAME INDEX

(or

RENAME KEY

) syntax, for example:

```
ALTER TABLE t1 RENAME INDEX i_old TO i_new;
```

## RENAME COLUMN

MariaDB starting with 10.5.2

From MariaDB 10.5.2 , it is possible to rename a column using the

RENAME COLUMN

syntax, for example:

```
ALTER TABLE t1 RENAME COLUMN c_old TO c_new;
```

## ADD PRIMARY KEY

Add a primary key.

For

PRIMARY KEY

indexes, you can specify a name for the index, but it is silently ignored, and the name of the index is always

PRIMARY

See [Getting Started with Indexes: Primary Key](#) for more information.

## DROP PRIMARY KEY

Drop a primary key.

For

PRIMARY KEY

indexes, you can specify a name for the index, but it is silently ignored, and the name of the index is always

PRIMARY

See [Getting Started with Indexes: Primary Key](#) for more information.

## ADD FOREIGN KEY

Add a foreign key.

For

FOREIGN KEY  
indexes, a reference definition must be provided.

For

FOREIGN KEY  
indexes, you can specify a name for the constraint, using the  
CONSTRAINT  
keyword. That name will be used in error messages.

First, you have to specify the name of the target (parent) table and a column or a column list which must be indexed and whose values must match to the foreign key's values. The

MATCH  
clause is accepted to improve the compatibility with other DBMS's, but has no meaning in MariaDB. The  
ON DELETE  
and  
ON UPDATE  
clauses specify what must be done when a  
DELETE  
(or a  
REPLACE  
) statements attempts to delete a referenced row from the parent table, and when an  
UPDATE  
statement attempts to modify the referenced foreign key columns in a parent table row, respectively. The following options are allowed:

- RESTRICT  
: The delete/update operation is not performed. The statement terminates with a 1451 error (SQLSTATE '2300').
- NO ACTION  
: Synonym for  
RESTRICT
- CASCADE  
: The delete/update operation is performed in both tables.
- SET NULL  
: The update or delete goes ahead in the parent table, and the corresponding foreign key fields in the child table are set to  
NULL  
. (They must not be defined as  
NOT NULL  
for this to succeed).
- SET DEFAULT  
: This option is implemented only for the legacy PBXT storage engine, which is disabled by default and no longer maintained. It sets the  
child table's foreign key fields to their  
DEFAULT  
values when the referenced parent table key entries are updated or deleted.

If either clause is omitted, the default behavior for the omitted clause is

RESTRICT

See [Foreign Keys](#) for more information.

## DROP FOREIGN KEY

Drop a foreign key.

See [Foreign Keys](#) for more information.

## ADD INDEX

Add a plain index.

Plain indexes are regular indexes that are not unique, and are not acting as a primary key or a foreign key. They are also not the "specialized"

FULLTEXT  
or  
SPATIAL

indexes.

See [Getting Started with Indexes: Plain Indexes](#) for more information.

## DROP INDEX

Drop a plain index.

Plain indexes are regular indexes that are not unique, and are not acting as a primary key or a foreign key. They are also not the "specialized"

FULLTEXT  
or  
SPATIAL  
indexes.

See [Getting Started with Indexes: Plain Indexes](#) for more information.

## ADD UNIQUE INDEX

Add a unique index.

The

UNIQUE  
keyword means that the index will not accept duplicated values, except for NULLs. An error will raise if you try to insert duplicate values in a  
UNIQUE index.

For

UNIQUE  
indexes, you can specify a name for the constraint, using the  
CONSTRAINT  
keyword. That name will be used in error messages.

See [Getting Started with Indexes: Unique Index](#) for more information.

## DROP UNIQUE INDEX

Drop a unique index.

The

UNIQUE  
keyword means that the index will not accept duplicated values, except for NULLs. An error will raise if you try to insert duplicate values in a  
UNIQUE index.

For

UNIQUE  
indexes, you can specify a name for the constraint, using the  
CONSTRAINT  
keyword. That name will be used in error messages.

See [Getting Started with Indexes: Unique Index](#) for more information.

## ADD FULLTEXT INDEX

Add a

FULLTEXT  
index.

See [Full-Text Indexes](#) for more information.

## DROP FULLTEXT INDEX

Drop a

FULLTEXT  
index.

See [Full-Text Indexes](#) for more information.

## ADD SPATIAL INDEX

Add a SPATIAL index.

See [SPATIAL INDEX](#) for more information.

## DROP SPATIAL INDEX

Drop a SPATIAL index.

See [SPATIAL INDEX](#) for more information.

## ENABLE/ DISABLE KEYS

DISABLE KEYS

will disable all non unique keys for the table for storage engines that support this (at least MyISAM and Aria). This can be used to speed up [inserts](#) into empty tables.

ENABLE KEYS

will enable all disabled keys.

## RENAME TO

Renames the table. See also [RENAME TABLE](#).

## ADD CONSTRAINT

Modifies the table adding a [constraint](#) on a particular column or columns.

MariaDB starting with 10.2.1

MariaDB 10.2.1 introduced new ways to define a constraint.

Note: Before MariaDB 10.2.1 , constraint expressions were accepted in syntax, but ignored.

```
ALTER TABLE table_name  
ADD CONSTRAINT [constraint_name] CHECK(expression);
```

Before a row is inserted or updated, all constraints are evaluated in the order they are defined. If any constraint fails, then the row will not be updated. One can use most deterministic functions in a constraint, including [UDF's](#).

```
CREATE TABLE account_ledger (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    transaction_name VARCHAR(100),  
    credit_account VARCHAR(100),  
    credit_amount INT,  
    debit_account VARCHAR(100),  
    debit_amount INT);  
  
ALTER TABLE account_ledger  
ADD CONSTRAINT is_balanced  
    CHECK((debit_amount + credit_amount) = 0);
```

The

constraint\_name  
is optional. If you don't provide one in the  
ALTER TABLE  
statement, MariaDB auto-generates a name for you. This is done so that you can remove it later using [DROP CONSTRAINT](#) clause.

You can disable all constraint expression checks by setting the variable [check\\_constraint\\_checks](#) to

OFF

. You may find this useful when loading a table that violates some constraints that you want to later find and fix in SQL.

To view constraints on a table, query [information\\_schema.TABLE\\_CONSTRAINTS](#) :

```
SELECT CONSTRAINT_NAME, TABLE_NAME, CONSTRAINT_TYPE  
FROM information_schema.TABLE_CONSTRAINTS  
WHERE TABLE_NAME = 'account_ledger';  
  
+-----+-----+  
| CONSTRAINT_NAME | TABLE_NAME | CONSTRAINT_TYPE |  
+-----+-----+  
| is_balanced | account_ledger | CHECK |  
+-----+-----+
```

## DROP CONSTRAINT

MariaDB starting with 10.2.22

```
DROP CONSTRAINT
for
UNIQUE
and
FOREIGN KEY
constraints was introduced in MariaDB 10.2.22 and MariaDB 10.3.13.
```

MariaDB starting with 10.2.1

```
DROP CONSTRAINT
for
CHECK
constraints was introduced in MariaDB 10.2.1
```

Modifies the table, removing the given constraint.

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

When you add a constraint to a table, whether through a `CREATE TABLE` or `ALTER TABLE...ADD CONSTRAINT` statement, you can either set a `constraint_name` yourself, or allow MariaDB to auto-generate one for you. To view constraints on a table, query `information_schema.TABLE_CONSTRAINTS`. For instance,

```
CREATE TABLE t (
    a INT,
    b INT,
    c INT,
    CONSTRAINT CHECK(a > b),
    CONSTRAINT check_equals CHECK(a = c);

SELECT CONSTRAINT_NAME, TABLE_NAME, CONSTRAINT_TYPE
FROM information_schema.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 't';

+-----+-----+
| CONSTRAINT_NAME | TABLE_NAME      | CONSTRAINT_TYPE |
+-----+-----+
| check_equals   | t              | CHECK           |
| CONSTRAINT_1   | t              | CHECK           |
+-----+-----+
```

To remove a constraint from the table, issue an

```
ALTER TABLE...DROP CONSTRAINT
statement. For example,
```

```
ALTER TABLE t DROP CONSTRAINT is_unique;
```

## ADD SYSTEM VERSIONING

MariaDB starting with 10.3.4

`System-versioned tables` was added in MariaDB 10.3.4 .

Add system versioning.

## DROP SYSTEM VERSIONING

MariaDB starting with 10.3.4

`System-versioned tables` was added in MariaDB 10.3.4 .

Drop system versioning.

## ADD PERIOD FOR SYSTEM\_TIME

MariaDB starting with 10.3.4

System-versioned tables was added in MariaDB 10.3.4 .

## FORCE

```
ALTER TABLE ... FORCE  
can force MariaDB to re-build the table.
```

In MariaDB 5.5 and before, this could only be done by setting the [ENGINE](#) table option to its old value. For example, for an InnoDB table, one could execute the following:

```
ALTER TABLE tab_name ENGINE = InnoDB;
```

The

```
FORCE  
option can be used instead. For example, :
```

```
ALTER TABLE tab_name FORCE;
```

With InnoDB, the table rebuild will only reclaim unused space (i.e. the space previously used for deleted rows) if the [innodb\\_file\\_per\\_table](#) system variable is set to

```
ON  
. If the system variable is  
OFF  
, then the space will not be reclaimed, but it will be-re-used for new data that's later added.
```

## EXCHANGE PARTITION

This is used to exchange the tablespace files between a partition and another table.

See [copying InnoDB's transportable tablespaces](#) for more information.

## DISCARD TABLESPACE

This is used to discard an InnoDB table's tablespace.

See [copying InnoDB's transportable tablespaces](#) for more information.

## IMPORT TABLESPACE

This is used to import an InnoDB table's tablespace. The tablespace should have been copied from its original server after executing [FLUSH TABLES FOR EXPORT](#).

See [copying InnoDB's transportable tablespaces](#) for more information.

```
ALTER TABLE ... IMPORT  
only applies to InnoDB tables. Most other popular storage engines, such as Aria and MyISAM, will recognize their data files as soon as they've  
been placed in the proper directory under the datadir, and no special DDL is required to import them.
```

## ALGORITHM

The

```
ALTER TABLE  
statement supports the  
ALGORITHM  
clause. This clause is one of the clauses that is used to implement online DDL.  
ALTER TABLE  
supports several different algorithms. An algorithm can be explicitly chosen for an  
ALTER TABLE  
operation by setting the  
ALGORITHM  
clause. The supported values are:  
  
•  
    ALGORITHM=DEFAULT
```

- This implies the default behavior for the specific statement, such as if no `ALGORITHM` clause is specified.
- `ALGORITHM=COPY`
- `ALGORITHM=INPLACE`
- `ALGORITHM=NOCOPY`
  - This was added in [MariaDB 10.3.7](#).
- `ALGORITHM=INSTANT`
  - This was added in [MariaDB 10.3.7](#).

See [InnoDB Online DDL Overview: ALGORITHM](#) for information on how the `ALGORITHM` clause affects InnoDB.

## `ALGORITHM=DEFAULT`

The default behavior, which occurs if

```
ALGORITHM=DEFAULT
is specified, or if
ALGORITHM
```

is not specified at all, usually only makes a copy if the operation doesn't support being done in-place at all. In this case, the most efficient available algorithm will usually be used.

However, in [MariaDB 10.3.6](#) and before, if the value of the `old_alter_table` system variable is set to

```
ON
, then the default behavior is to perform
ALTER TABLE
operations by making a copy of the table using the old algorithm.
```

In [MariaDB 10.3.7](#) and later, the `old_alter_table` system variable is deprecated. Instead, the `alter_algorithm` system variable defines the default algorithm for

```
ALTER TABLE
operations.
```

## `ALGORITHM=COPY`

```
ALGORITHM=COPY
is the name for the original ALTER TABLE algorithm from early MariaDB versions.
```

When

```
ALGORITHM=COPY
is set, MariaDB essentially does the following operations:
```

```
-- Create a temporary table with the new definition
CREATE TEMPORARY TABLE tmp_tab (
  ...
);

-- Copy the data from the original table
INSERT INTO tmp_tab
  SELECT * FROM original_tab;

-- Drop the original table
DROP TABLE original_tab;

-- Rename the temporary table, so that it replaces the original one
RENAME TABLE tmp_tab TO original_tab;
```

This algorithm is very inefficient, but it is generic, so it works for all storage engines.

If

```
ALGORITHM=COPY
is specified, then the copy algorithm will be used even if it is not necessary. This can result in a lengthy table copy. If multiple ALTER TABLE operations are required that each require the table to be rebuilt, then it is best to specify all operations in a single ALTER TABLE statement, so that the table is only rebuilt once.
```

## `ALGORITHM=INPLACE`

`ALGORITHM=COPY`

can be incredibly slow, because the whole table has to be copied and rebuilt.

`ALGORITHM=INPLACE`

was introduced as a way to avoid this by performing operations in-place and avoiding the table copy and rebuild, when possible.

When

`ALGORITHM=INPLACE`

is set, the underlying storage engine uses optimizations to perform the operation while avoiding the table copy and rebuild. However, `INPLACE`

is a bit of a misnomer, since some operations may still require the table to be rebuilt for some storage engines. Regardless, several operations can be performed without a full copy of the table for some storage engines.

A more accurate name would have been

`ALGORITHM=ENGINE`

, where

`ENGINE`

refers to an "engine-specific" algorithm.

If an [ALTER TABLE](#) operation supports

`ALGORITHM=INPLACE`

, then it can be performed using optimizations by the underlying storage engine, but it may rebuilt.

See [InnoDB Online DDL Operations with ALGORITHM=INPLACE](#) for more.

## ALGORITHM=NOCOPY

`ALGORITHM=NOCOPY`

was introduced in [MariaDB 10.3.7](#).

`ALGORITHM=INPLACE`

can sometimes be surprisingly slow in instances where it has to rebuild the clustered index, because when the clustered index has to be rebuilt, the whole table has to be rebuilt.

`ALGORITHM=NOCOPY`

was introduced as a way to avoid this.

If an

`ALTER TABLE`

operation supports

`ALGORITHM=NOCOPY`

, then it can be performed without rebuilding the clustered index.

If

`ALGORITHM=NOCOPY`

is specified for an

`ALTER TABLE`

operation that does not support

`ALGORITHM=NOCOPY`

, then an error will be raised. In this case, raising an error is preferable, if the alternative is for the operation to rebuild the clustered index, and perform unexpectedly slowly.

See [InnoDB Online DDL Operations with ALGORITHM=NOCOPY](#) for more.

## ALGORITHM=INSTANT

`ALGORITHM=INSTANT`

was introduced in [MariaDB 10.3.7](#).

`ALGORITHM=INPLACE`

can sometimes be surprisingly slow in instances where it has to modify data files.

`ALGORITHM=INSTANT`

was introduced as a way to avoid this.

If an

`ALTER TABLE`

operation supports

`ALGORITHM=INSTANT`

, then it can be performed without modifying any data files.

If

`ALGORITHM=INSTANT`

is specified for an

```
ALTER TABLE
operation that does not support
ALGORITHM=INSTANT
, then an error will be raised. In this case, raising an error is preferable, if the alternative is for the operation to modify data files, and perform
unexpectedly slowly.
```

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT](#) for more.

## LOCK

The

```
ALTER TABLE
statement supports the
LOCK
clause. This clause is one of the clauses that is used to implement online DDL.
```

```
ALTER TABLE
supports several different locking strategies. A locking strategy can be explicitly chosen for an
ALTER TABLE
operation by setting the
LOCK
clause. The supported values are:
```

- **DEFAULT**  
: Acquire the least restrictive lock on the table that is supported for the specific operation. Permit the maximum amount of concurrency that is supported for the specific operation.
- **NONE**  
: Acquire no lock on the table. Permit **all** concurrent DML. If this locking strategy is not permitted for an operation, then an error is raised.
- **SHARED**  
: Acquire a read lock on the table. Permit **read-only** concurrent DML. If this locking strategy is not permitted for an operation, then an error is raised.
- **EXCLUSIVE**  
: Acquire a write lock on the table. Do **not** permit concurrent DML.

Different storage engines support different locking strategies for different operations. If a specific locking strategy is chosen for an

```
ALTER TABLE
operation, and that table's storage engine does not support that locking strategy for that specific operation, then an error will be raised.
```

If the

```
LOCK
clause is not explicitly set, then the operation uses
LOCK=DEFAULT
```

[ALTER ONLINE TABLE](#) is equivalent to

```
LOCK=NONE
. Therefore, the ALTER ONLINE TABLE statement can be used to ensure that your
ALTER TABLE
operation allows all concurrent DML.
```

See [InnoDB Online DDL Overview: LOCK](#) for information on how the

```
LOCK
clause affects InnoDB.
```

## Progress Reporting

MariaDB provides progress reporting for

```
ALTER TABLE
statement for clients that support the new progress reporting protocol. For example, if you were using the mysql client, then the progress report
might look like this::
```

```
ALTER TABLE test ENGINE=Aria;
Stage: 1 of 2 'copy to tmp table'    46% of stage
```

The progress report is also shown in the output of the `SHOW PROCESSLIST` statement and in the contents of the `information_schema.PROCESSLIST` table.

See [Progress Reporting](#) for more information.

# Aborting ALTER TABLE Operations

If an

```
ALTER TABLE
```

operation is being performed and the connection is killed, the changes will be rolled back in a controlled manner. The rollback can be a slow operation as the time it takes is relative to how far the operation has progressed.

MariaDB starting with 10.2.13

Aborting

```
ALTER TABLE ... ALGORITHM=COPY
```

was made faster by removing excessive undo logging ( [MDEV-11415](#) ). This significantly shortens the time it takes to abort a running ALTER TABLE operation.

# Atomic ALTER TABLE

MariaDB starting with 10.6.1

From [MariaDB 10.6](#),

```
ALTER TABLE
```

is atomic for most engines, including InnoDB, MyRocks, MyISAM and Aria ( [MDEV-25180](#) ). This means that if there is a crash (server down or power outage) during an

```
ALTER TABLE
```

operation, after recovery, either the old table and associated triggers and status will be intact, or the new table will be active.

In older MariaDB versions one could get leftover #sql-alter..', '#sql-backup..' or 'table\_name.frm" files if the system crashed during the

```
ALTER TABLE
```

operation.

See [Atomic DDL](#) for more information.

# Replication

MariaDB starting with 10.8.0

Before [MariaDB 10.8.0](#), ALTER TABLE got fully executed on the primary first, and only then was it replicated and started executing on replicas. From [MariaDB 10.8.0](#), ALTER TABLE gets replicated and starts executing on replicas when it *starts* executing on the primary, not when it *finishes*. This way the replication lag caused by a heavy ALTER TABLE can be completely eliminated ( [MDEV-11675](#) ).

# Examples

Adding a new column:

```
ALTER TABLE t1 ADD x INT;
```

Dropping a column:

```
ALTER TABLE t1 DROP x;
```

Modifying the type of a column:

```
ALTER TABLE t1 MODIFY x bigint unsigned;
```

Changing the name and type of a column:

```
ALTER TABLE t1 CHANGE a b bigint unsigned auto_increment;
```

Combining multiple clauses in a single ALTER TABLE statement, separated by commas:

```
ALTER TABLE t1 DROP x, ADD x2 INT, CHANGE y y2 INT;
```

Changing the storage engine and adding a comment:

```
ALTER TABLE t1
  ENGINE = InnoDB
  COMMENT = 'First of three tables containing usage info';
```

Rebuilding the table (the previous example will also rebuild the table if it was already InnoDB):

```
ALTER TABLE t1 FORCE;
```

Dropping an index:

```
ALTER TABLE rooms DROP INDEX u;
```

Adding a unique index:

```
ALTER TABLE rooms ADD UNIQUE INDEX u(room_number);
```

From [MariaDB 10.5.3](#), adding a primary key for an [application-time period table](#) with a [WITHOUT OVERLAPS](#) constraint:

```
ALTER TABLE rooms ADD PRIMARY KEY(room_number, p WITHOUT OVERLAPS);
```

## See Also

- [CREATE TABLE](#)
- [DROP TABLE](#)
- [Character Sets and Collations](#)
- [SHOW CREATE TABLE](#)
- [Instant ADD COLUMN for InnoDB](#)

## 1.1.2.1.1.2 ALTER DATABASE

Modifies a database, changing its overall characteristics.

## Syntax

```
ALTER {DATABASE | SCHEMA} [db_name]
    alter_specification ...
ALTER {DATABASE | SCHEMA} db_name
    UPGRADE DATA DIRECTORY NAME

alter_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
    | [DEFAULT] COLLATE [=] collation_name
    | COMMENT [=] 'comment'
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

ALTER DATABASE  
enables you to change the overall characteristics of a database. These characteristics are stored in the db.opt file in the database directory. To use  
ALTER DATABASE , you need the  
ALTER privilege on the database.  
ALTER SCHEMA is a synonym for ALTER DATABASE.

The

CHARACTER SET clause changes the default database character set. The  
COLLATE clause changes the default database collation. See [Character Sets and Collations](#) for more.

You can see what character sets and collations are available using, respectively, the [SHOW CHARACTER SET](#) and [SHOW COLLATION](#) statements.

Changing the default character set/collation of a database does not change the character set/collation of any [stored procedures](#) or [stored functions](#) that were previously created, and relied on the defaults. These need to be dropped and recreated in order to apply the character set/collation changes.

The database name can be omitted from the first syntax, in which case the statement applies to the default database.

The syntax that includes the

```
UPGRADE DATA DIRECTORY NAME
```

clause was added in MySQL 5.1.23. It updates the name of the directory associated with the database to use the encoding implemented in MySQL 5.1 for mapping database names to database directory names (see [Identifier to File Name Mapping](#)). This clause is for use under these conditions:

- It is intended when upgrading MySQL to 5.1 or later from older versions.
- It is intended to update a database directory name to the current encoding format if the name contains special characters that need encoding.
- The statement is used by

```
mysqlcheck  
(as invoked by  
mysql_upgrade  
).
```

For example, if a database in MySQL 5.0 has a name of a-b-c, the name contains instance of the '-' character. In 5.0, the database directory is also named a-b-c, which is not necessarily safe for all file systems. In MySQL 5.1 and up, the same database name is encoded as a@002db@002dc to produce a file system-neutral directory name.

When a MySQL installation is upgraded to MySQL 5.1 or later from an older version, the server displays a name such as a-b-c (which is in the old format) as #mysql50#a-b-c, and you must refer to the name using the #mysql50# prefix. Use

```
UPGRADE DATA DIRECTORY NAME
```

in this case to explicitly tell the server to re-encode the database directory name to the current encoding format:

```
ALTER DATABASE `#mysql50#a-b-c` UPGRADE DATA DIRECTORY NAME;
```

After executing this statement, you can refer to the database as a-b-c without the special #mysql50# prefix.

## COMMENT

MariaDB starting with [10.5.0](#)

From [MariaDB 10.5.0](#), it is possible to add a comment of a maximum of 1024 bytes. If the comment length exceeds this length, an error/warning code 4144 is thrown. The database comment is also added to the db.opt file, as well as to the [information\\_schema.schemata table](#).

## Examples

```
ALTER DATABASE test CHARACTER SET='utf8' COLLATE='utf8_bin';
```

From [MariaDB 10.5.0](#):

```
ALTER DATABASE p COMMENT='Presentations';
```

## See Also

- [CREATE DATABASE](#)
- [DROP DATABASE](#)
- [SHOW CREATE DATABASE](#)
- [SHOW DATABASES](#)
- [Character Sets and Collations](#)
- [Information Schema SCHEMATA Table](#)

## 1.1.2.1.1.3 ALTER EVENT

Modifies one or more characteristics of an existing event.

## Syntax

```
ALTER
[DEFINER = { user | CURRENT_USER }]
EVENT event_name
[ON SCHEDULE schedule]
[ON COMPLETION [NOT] PRESERVE]
[RENAME TO new_event_name]
[ENABLE | DISABLE | DISABLE ON SLAVE]
[COMMENT 'comment']
[DO sql_statement]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

ALTER EVENT

statement is used to change one or more of the characteristics of an existing [event](#) without the need to drop and recreate it. The syntax for each of the

DEFINER

,

ON SCHEDULE

,

ON COMPLETION

,

COMMENT

,

ENABLE

/

DISABLE

, and

DO

clauses is exactly the same as when used with

[CREATE EVENT](#)

This statement requires the

[EVENT](#)

privilege. When a user executes a successful

ALTER EVENT

statement, that user becomes the definer for the affected event.

(In MySQL 5.1.11 and earlier, an event could be altered only by its definer, or by a user having the

[SUPER](#)

privilege.)

ALTER EVENT

works only with an existing event:

```
ALTER EVENT no_such_event ON SCHEDULE EVERY '2:3' DAY_HOUR;
ERROR 1539 (HY000): Unknown event 'no_such_event'
```

## Examples

```
ALTER EVENT myevent
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 2 HOUR
DO
    UPDATE myschema.mytable SET mycol = mycol + 1;
```

## See Also

- [Events Overview](#)
- [CREATE EVENT](#)
- [SHOW CREATE EVENT](#)
- [DROP EVENT](#)

### 1.1.2.1.1.4 ALTER FUNCTION

#### Syntax

```
ALTER FUNCTION func_name [characteristic ...]

characteristic:
  { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [See Also](#)

#### Description

This statement can be used to change the characteristics of a stored function. More than one change may be specified in an

ALTER FUNCTION

statement. However, you cannot change the parameters or body of a stored function using this statement; to make such changes, you must drop and re-create the function using [DROP FUNCTION](#) and [CREATE FUNCTION](#).

You must have the

ALTER ROUTINE

privilege for the function. (That privilege is granted automatically to the function creator.) If binary logging is enabled, the

ALTER FUNCTION

statement might also require the

SUPER

privilege, as described in [Binary Logging of Stored Routines](#).

#### Example

```
ALTER FUNCTION hello SQL SECURITY INVOKER;
```

#### See Also

- [CREATE FUNCTION](#)
- [SHOW CREATE FUNCTION](#)
- [DROP FUNCTION](#)
- [SHOW FUNCTION STATUS](#)
- [Information Schema ROUTINES Table](#)

### 1.1.2.1.1.5 ALTER LOGFILE GROUP

#### Syntax

```
ALTER LOGFILE GROUP logfile_group
  ADD UNDOFILE 'file_name'
  [INITIAL_SIZE [=] size]
  [WAIT]
  ENGINE [=] engine_name
```

The

ALTER LOGFILE GROUP

statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. See [MDEV-19295](#) for more information.

## 1.1.2.1.1.6 ALTER PROCEDURE

### Syntax

```
ALTER PROCEDURE proc_name [characteristic ...]

characteristic:
  { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'
```

### Description

This statement can be used to change the characteristics of a [stored procedure](#). More than one change may be specified in an

ALTER PROCEDURE

statement. However, you cannot change the parameters or body of a stored procedure using this statement. To make such changes, you must drop and re-create the procedure using either [CREATE OR REPLACE PROCEDURE](#) (since [MariaDB 10.1.3](#)) or [DROP PROCEDURE](#) and [CREATE PROCEDURE](#) ([MariaDB 10.1.2](#) and before).

You must have the

ALTER ROUTINE

privilege for the procedure. By default, that privilege is granted automatically to the procedure creator. See [Stored Routine Privileges](#).

### Example

```
ALTER PROCEDURE simpleproc SQL SECURITY INVOKER;
```

### See Also

- [Stored Procedure Overview](#)
- [CREATE PROCEDURE](#)
- [SHOW CREATE PROCEDURE](#)
- [DROP PROCEDURE](#)
- [SHOW CREATE PROCEDURE](#)
- [SHOW PROCEDURE STATUS](#)
- [Stored Routine Privileges](#)
- [Information Schema ROUTINES Table](#)

## 1.1.2.1.1.7 ALTER SEQUENCE

MariaDB starting with [10.3.1](#)

```
ALTER SEQUENCE
was introduced in MariaDB 10.3.
```

### Syntax

```

ALTER SEQUENCE [IF EXISTS] sequence_name
[ INCREMENT [ BY | = ] increment ]
[ MINVALUE [=] minvalue | NO MINVALUE | NOMINVALUE ]
[ MAXVALUE [=] maxvalue | NO MAXVALUE | NOMAXVALUE ]
[ START [ WITH | = ] start ] [ CACHE [=] cache ] [ [ NO ] CYCLE ]
[ RESTART [[WITH | =] restart]

```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [Arguments to ALTER SEQUENCE](#)
  2. [INSERT](#)
  3. [Notes](#)
3. [See Also](#)

ALTER SEQUENCE  
 allows one to change any values for a  
 SEQUENCE  
 created with [CREATE SEQUENCE](#).

The options for

ALTER SEQUENCE  
 can be given in any order.

## Description

ALTER SEQUENCE  
 changes the parameters of an existing sequence generator. Any parameters not specifically set in the  
 ALTER SEQUENCE  
 command retain their prior settings.

ALTER SEQUENCE  
 requires the [ALTER](#) privilege.

## Arguments to ALTER SEQUENCE

The following options may be used:

Option	Default value	Description
INCREMENT	1	Increment to use for values. May be negative.
MINVALUE	1 if INCREMENT > 0 and - 9223372036854775807 if INCREMENT < 0	Minimum value for the sequence.
MAXVALUE	9223372036854775806 if INCREMENT > 0 and -1 if INCREMENT < 0	Max value for sequence.

START	<pre> MINVALUE if INCREMENT &gt; 0 and MAX_VALUE if INCREMENT &lt; 0 </pre>	First value that the sequence will generate.
CACHE	1000	Number of values that should be cached. 0 if no CACHE . The underlying table will be updated first time a new sequence number is generated and each time the cache runs out.
CYCLE	0 (= NO CYCLE )	1 if the sequence should start again from MINVALUE # after it has run out of values.
RESTART	<pre> START if restart value not is given </pre>	If RESTART option is used, NEXT VALUE will return the restart value.

The optional clause

```

RESTART [ WITH restart ]
sets the next value for the sequence. This is equivalent to calling the SETVAL() function with the
is_used
argument as 0. The specified value will be returned by the next call of nextval. Using
RESTART
with no restart value is equivalent to supplying the start value that was recorded by CREATE SEQUENCE or last set by
ALTER SEQUENCE START WITH

```

```

ALTER SEQUENCE
will not allow you to change the sequence so that it's inconsistent. For example:

```

```

CREATE SEQUENCE s1;
ALTER SEQUENCE s1 MINVALUE 10;
ERROR 4061 (HY000): Sequence 'test.t1' values are conflicting

ALTER SEQUENCE s1 MINVALUE 10 RESTART 10;
ERROR 4061 (HY000): Sequence 'test.t1' values are conflicting

ALTER SEQUENCE s1 MINVALUE 10 START 10 RESTART 10;

```

## INSERT

To allow

```

SEQUENCE
objects to be backed up by old tools, like mysqldump , one can use
SELECT
to read the current state of a
SEQUENCE
object and use an
INSERT
to update the
SEQUENCE
object.
INSERT
is only allowed if all fields are specified:

```

```
CREATE SEQUENCE s1;
INSERT INTO s1 VALUES(1000,10,2000,1005,1,1000,0,0);
SELECT * FROM s1;

+-----+-----+-----+-----+-----+-----+
| next_value | min_value | max_value | start | increment | cache |
+-----+-----+-----+-----+-----+-----+
|          1000 |         10 |        2000 |    1005 |          1 |     1000 |
+-----+-----+-----+-----+-----+-----+
SHOW CREATE SEQUENCE s1;
+-----+
| Table | Create Table |
+-----+
| s1   | CREATE SEQUENCE `s1` start with 1005 minvalue 10 maxvalue 2000 increment by 1 cache 1000 nocycle ENGINE=Aria |
+-----+
```

## Notes

**ALTER SEQUENCE**  
will instantly affect all future  
**SEQUENCE**  
operations. This is in contrast to some other databases where the changes requested by  
**ALTER SEQUENCE**  
will not be seen until the sequence cache has run out.

ALTER SEQUENCE  
will take a full table lock of the sequence object during its (brief) operation. This ensures that  
ALTER SEQUENCE  
is replicated correctly. If you only want to set the next sequence value to a higher value than current, then you should use [SETVAL\(\)](#) instead, as this is not blocking.

If you want to change storage engine, sequence comment or rename the sequence, you can use `ALTER TABLE` for this.

#### See Also

- Sequence Overview
  - CREATE SEQUENCE
  - DROP SEQUENCE
  - NEXT VALUE FOR
  - PREVIOUS VALUE FOR
  - SETVAL() . Set next value for the sequence.
  - AUTOINCREMENT
  - ALTER TABLE

## 1.1.2.1.1.8 ALTER SERVER

## Syntax

```
ALTER SERVER server_name  
    OPTIONS (option [, option] ...)
```

## Contents

1. Syntax
  2. Description
  3. Examples
  4. See Also

## Description

Alters the server information for `server_name`, adjusting the specified options as per the [CREATE SERVER](#) command. The corresponding fields in the [mysql.servers](#) table are updated accordingly. This statement requires the [SUPER](#) privilege or, from MariaDB 10.5.2, the [FEDERATED ADMIN](#) privilege.

ALTER SERVER is not written to the [binary log](#), irrespective of the [binary log format](#) being used. From MariaDB 10.1.13, Galera replicates the [CREATE SERVER](#), [ALTER SERVER](#) and [DROP SERVER](#) statements.

## Examples

```
ALTER SERVER s OPTIONS (USER 'sally');
```

## See Also

- [CREATE SERVER](#)
- [DROP SERVER](#)
- [Spider Storage Engine](#)

## 1.1.2.1.1.9 ALTER TABLESPACE

The

ALTER TABLESPACE

statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. In MySQL 5.7 and later, the statement is also supported for InnoDB. However, MariaDB has chosen not to include that specific feature. See [MDEV-19294](#) for more information.

## 1.1.2.1.1.10 ALTER USER

## 1.1.2.1.2 ALTER VIEW

### Syntax

```
ALTER
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
[DEFINER = { user | CURRENT_USER }]
[SQL SECURITY { DEFINER | INVOKER }]
VIEW view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [See Also](#)

### Description

This statement changes the definition of a [view](#), which must exist. The syntax is similar to that for [CREATE VIEW](#) and the effect is the same as for

CREATE OR REPLACE VIEW  
if the view exists. This statement requires the  
CREATE VIEW  
and  
DROP  
[privileges](#) for the view, and some privilege for each column referred to in the  
SELECT  
statement.  
ALTER VIEW  
is allowed only to the definer or users with the [SUPER](#) privilege.

### Example

```
ALTER VIEW v AS SELECT a, a*3 AS a2 FROM t;
```

## See Also

- [CREATE VIEW](#)
- [DROP VIEW](#)
- [SHOW CREATE VIEW](#)
- [INFORMATION SCHEMA VIEWS Table](#)

## 1.1.2.1.3 ANALYZE TABLE

### Syntax

```
ANALYZE [NO_WRITE_TO_BINLOG | LOCAL] TABLE tbl_name [,tbl_name ...]
[PERSISTENT FOR [ALL|COLUMNS ([col_name [,col_name ...]])]
[INDEXES ([index_name [,index_name ...]])]]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Engine-Independent Statistics](#)
4. [See Also](#)

### Description

#### ANALYZE TABLE

analyzes and stores the key distribution for a table ( [index statistics](#) ). This statement works with [MyISAM](#) , [Aria](#) and [InnoDB](#) tables. During the analysis, InnoDB will allow reads/writes, and MyISAM/Aria reads/inserts. For MyISAM tables, this statement is equivalent to using `myisamchk --analyze` .

For more information on how the analysis works within InnoDB, see [InnoDB Limitations](#) .

MariaDB uses the stored key distribution to decide the order in which tables should be joined when you perform a join on something other than a constant. In addition, key distributions can be used when deciding which indexes to use for a specific table within a query.

This statement requires [SELECT](#) and [INSERT](#) privileges for the table.

By default, ANALYZE TABLE statements are written to the [binary log](#) and will be [replicated](#) . The

```
NO_WRITE_TO_BINLOG
keyword (
LOCAL
is an alias) will ensure the statement is not written to the binary log.
```

From [MariaDB 10.3.19](#) ,

```
ANALYZE TABLE
statements are not logged to the binary log if read\_only is set. See also Read-Only Replicas .
```

#### ANALYZE TABLE

is also supported for partitioned tables. You can use

#### ALTER TABLE

```
... ANALYZE PARTITION
to analyze one or more partitions.
```

The [Aria](#) storage engine supports [progress reporting](#) for the

```
ANALYZE TABLE
statement.
```

## Engine-Independent Statistics

#### ANALYZE TABLE

supports [engine-independent statistics](#) . See [Engine-Independent Table Statistics: Collecting Statistics with the ANALYZE TABLE Statement](#) for more information.

### See Also

- [Index Statistics](#)
- [InnoDB Persistent Statistics](#)
- [Progress Reporting](#)
- [Engine-independent Statistics](#)
- [Histogram-based Statistics](#)
- [ANALYZE Statement](#)

## 1.1.2.1.4 CHECK TABLE

# Syntax

```
CHECK TABLE tbl_name [, tbl_name] ... [option] ...
option = {FOR UPGRADE | QUICK | FAST | MEDIUM | EXTENDED | CHANGED}
```

## Description

**CHECK TABLE**  
checks a table or tables for errors.  
**CHECK TABLE**  
works for [Archive](#) , [Aria](#) , [CSV](#) , [InnoDB](#) , and [MyISAM](#) tables. For Aria and MyISAM tables, the key statistics are updated as well. For CSV, see also [Checking and Repairing CSV Tables](#) .

As an alternative, [myisamchk](#) is a commandline tool for checking MyISAM tables when the tables are not being accessed.

For checking [dynamic columns](#) integrity,

[COLUMN\\_CHECK\(\)](#)

can be used.

**CHECK TABLE**  
can also check views for problems, such as tables that are referenced in the view definition that no longer exist.

**CHECK TABLE**  
is also supported for partitioned tables. You can use

[ALTER TABLE](#)

... [CHECK PARTITION](#)  
to check one or more partitions.

The meaning of the different options are as follows - note that this can vary a bit between storage engines:

FOR UPGRADE	Do a very quick check if the storage format for the table has changed so that one needs to do a REPAIR. This is only needed when one upgrades between major versions of MariaDB or MySQL. This is usually done by <a href="#">running mysql_upgrade</a> .
FAST	Only check tables that has not been closed properly or are marked as corrupt. Only supported by the MyISAM and Aria engines. For other engines the table is checked normally
CHANGED	Check only tables that has changed since last REPAIR / CHECK. Only supported by the MyISAM and Aria engines. For other engines the table is checked normally.
QUICK	Do a fast check. For MyISAM and Aria engine this means we skip checking the delete link chain which may take some time.
MEDIUM	Scan also the data files. Checks integrity between data and index files with checksums. In most cases this should find all possible errors.
EXTENDED	Does a full check to verify every possible error. For MyISAM and Aria we verify for each row that all it keys exists and points to the row. This may take a long time on big tables!

For most cases running

**CHECK TABLE**  
without options or  
**MEDIUM**  
should be good enough.

The [Aria](#) storage engine supports [progress reporting](#) for this statement.

If you want to know if two tables are identical, take a look at

[CHECKSUM TABLE](#)

## InnoDB

If

**CHECK TABLE**  
finds an error in an InnoDB table, MariaDB might shutdown to prevent the error propagation. In this case, the problem will be reported in the error log. Otherwise the table or an index might be marked as corrupted, to prevent use. This does not happen with some minor problems, like a wrong

number of entries in a secondary index. Those problems are reported in the output of  
CHECK TABLE

Each tablespace contains a header with metadata. This header is not checked by this statement.

During the execution of

CHECK TABLE  
, other threads may be blocked.

## 1.1.2.1.5 CHECK VIEW

### Syntax

```
CHECK VIEW view_name
```

### Description

The

CHECK VIEW

statement was introduced in [MariaDB 10.0.18](#) to assist with fixing [MDEV-6916](#) , an issue introduced in [MariaDB 5.2](#) where the view algorithms were swapped. It checks whether the view algorithm is correct. It is run as part of [mysql\\_upgrade](#) , and should not normally be required in regular use.

### See Also

- [REPAIR VIEW](#)

## 1.1.2.1.6 CHECKSUM TABLE

### Syntax

```
CHECKSUM TABLE tbl_name [, tbl_name] ... [ QUICK | EXTENDED ]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Differences Between MariaDB and MySQL](#)

### Description

CHECKSUM TABLE  
reports a table checksum. This is very useful if you want to know if two tables are the same (for example on a master and slave).

With

QUICK  
, the live table checksum is reported if it is available, or  
NULL  
otherwise. This is very fast. A live checksum is enabled by specifying the  
CHECKSUM=1  
table option when you [create the table](#) ; currently, this is supported only for [Aria](#) and [MyISAM](#) tables.

With

EXTENDED  
, the entire table is read row by row and the checksum is calculated. This can be very slow for large tables.

If neither

QUICK  
nor  
EXTENDED  
is specified, MariaDB returns a live checksum if the table storage engine supports it and scans the table otherwise.

CHECKSUM TABLE  
requires the [SELECT privilege](#) for the table.

For a nonexistent table,

```
CHECKSUM TABLE
returns
NULL
and generates a warning.
```

The table row format affects the checksum value. If the row format changes, the checksum will change. This means that when a table created with a MariaDB/MySQL version is upgraded to another version, the checksum value will probably change.

Two identical tables should always match to the same checksum value; however, also for non-identical tables there is a very slight chance that they will return the same value as the hashing algorithm is not completely collision-free.

## Differences Between MariaDB and MySQL

```
CHECKSUM TABLE
may give a different result as MariaDB doesn't ignore
NULL
s in the columns as MySQL 5.1 does (Later MySQL versions should calculate checksums the same way as MariaDB). You can get the 'old style'
checksum in MariaDB by starting mysqld with the
--old
option. Note however that the MyISAM and Aria storage engines in MariaDB are using the new checksum internally, so if you are using
--old
, the
CHECKSUM
command will be slower as it needs to calculate the checksum row by row.
```

### 1.1.2.1.7 CREATE TABLE

#### Syntax

```
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  [(create_definition,...)] [table_options] ... [partition_options]
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  [(create_definition,...)] [table_options] ... [partition_options]
  select_statement
CREATE [OR REPLACE] [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
  { LIKE old_table_name | (LIKE old_table_name) }

select_statement:
  [IGNORE | REPLACE] [AS] SELECT ...  (Some legal select statement)
```

#### Contents

- 1. [Syntax](#)
- 2. [Description](#)
- 3. [Privileges](#)
- 4. [CREATE OR REPLACE](#)
  - 1. [Things to be Aware of With CREATE OR REPLACE](#)
- 5. [CREATE TABLE IF NOT EXISTS](#)
- 6. [CREATE TEMPORARY TABLE](#)
- 7. [CREATE TABLE ... LIKE](#)
- 8. [CREATE TABLE ... SELECT](#)
- 9. [Column Definitions](#)
  - 1. [NULL and NOT NULL](#)
  - 2. [DEFAULT Column Option](#)
  - 3. [AUTO\\_INCREMENT Column Option](#)
  - 4. [ZEROFILL Column Option](#)
  - 5. [PRIMARY KEY Column Option](#)
  - 6. [UNIQUE KEY Column Option](#)
  - 7. [COMMENT Column Option](#)
  - 8. [REF\\_SYSTEM\\_ID](#)
  - 9. [Generated Columns](#)
- 10. [COMPRESSED](#)
- 11. [INVISIBLE](#)
- 12. [WITH SYSTEM VERSIONING Column Option](#)
- 13. [WITHOUT SYSTEM VERSIONING Column Option](#)
- 10. [Index Definitions](#)
  - 1. [Index Categories](#)

- 1. Plain Indexes
- 2. PRIMARY KEY
- 3. UNIQUE
- 4. FOREIGN KEY
- 5. FULLTEXT
- 6. SPATIAL
- 2. Index Options
  - 1. KEY\_BLOCK\_SIZE Index Option
  - 2. Index Types
  - 3. WITH PARSER Index Option
  - 4. COMMENT Index Option
  - 5. CLUSTERING Index Option
  - 6. IGNORED / NOT IGNORED
- 11. Periods
- 12. Constraint Expressions
- 13. Table Options
  - 1. [STORAGE] ENGINE
  - 2. AUTO\_INCREMENT
  - 3. AVG\_ROW\_LENGTH
  - 4. [DEFAULT] CHARACTER SET/CHARSET
  - 5. CHECKSUM/TABLE\_CHECKSUM
  - 6. [DEFAULT] COLLATE
  - 7. COMMENT
  - 8. CONNECTION
  - 9. DATA DIRECTORY/INDEX DIRECTORY
  - 10. DELAY\_KEY\_WRITE
  - 11. ENCRYPTED
  - 12. ENCRYPTION\_KEY\_ID
  - 13. IETF\_QUOTES
  - 14. INSERT\_METHOD
  - 15. KEY\_BLOCK\_SIZE
  - 16. MIN\_ROWS/MAX\_ROWS
  - 17. PACK\_KEYS
  - 18. PAGE\_CHECKSUM
  - 19. PAGE\_COMPRESSED
  - 20. PAGE\_COMPRESSION\_LEVEL
  - 21. PASSWORD
  - 22. RAID\_TYPE
  - 23. ROW\_FORMAT
    - 1. Supported MyISAM Row Formats
    - 2. Supported Aria Row Formats
    - 3. Supported InnoDB Row Formats
    - 4. Other Storage Engines and ROW\_FORMAT
  - 24. SEQUENCE
  - 25. STATS\_AUTO\_RECALC
  - 26. STATS\_PERSISTENT
  - 27. STATS\_SAMPLE\_PAGES
  - 28. TRANSACTIONAL
  - 29. UNION
  - 30. WITH SYSTEM VERSIONING
- 14. Partitions
- 15. Sequences
- 16. Atomic DDL
- 17. Examples
- 18. See Also

## Description

Use the

```
CREATE TABLE
statement to create a table with the given name.
```

In its most basic form, the

```
CREATE TABLE
statement provides a table name followed by a list of columns, indexes, and constraints. By default, the table is created in the default database.
```

Specify a database with

*db\_name*

*tbl\_name*

. If you quote the table name, you must quote the database name and table name separately as

*db\_name*

*tbl\_name*

. This is particularly useful for `CREATE TABLE ... SELECT`, because it allows to create a table into a database, which contains data from other databases. See [Identifier Qualifiers](#).

If a table with the same name exists, error 1050 results. Use `IF NOT EXISTS` to suppress this error and issue a note instead. Use `SHOW WARNINGS` to see notes.

The

`CREATE TABLE`

statement automatically commits the current transaction, except when using the `TEMPORARY` keyword.

For valid identifiers to use as table names, see [Identifier Names](#).

**Note:** if the `default_storage_engine` is set to ColumnStore then it needs setting on all UMs. Otherwise when the tables using the default engine are replicated across UMs they will use the wrong engine. You should therefore not use this option as a session variable with ColumnStore.

`Microsecond precision` can be between 0-6. If no precision is specified it is assumed to be 0, for backward compatibility reasons.

## Privileges

Executing the

`CREATE TABLE`

statement requires the `CREATE` privilege for the table or the database.

## CREATE OR REPLACE

If the

`OR REPLACE`

clause is used and the table already exists, then instead of returning an error, the server will drop the existing table and replace it with the newly defined table.

This syntax was originally added to make `replication` more robust if it has to rollback and repeat statements such as

`CREATE ... SELECT`

on replicas.

```
CREATE OR REPLACE TABLE table_name (a int);
```

is basically the same as:

```
DROP TABLE IF EXISTS table_name;
CREATE TABLE table_name (a int);
```

with the following exceptions:

- If  
*table\_name*  
was locked with `LOCK TABLES` it will continue to be locked after the statement.
- Temporary tables are only dropped if the  
`TEMPORARY`  
keyword was used. (With `DROP TABLE`, temporary tables are preferred to be dropped before normal tables).

## Things to be Aware of With CREATE OR REPLACE

- The table is dropped first (if it existed), after that the  
`CREATE`  
is done. Because of this, if the

- CREATE  
fails, then the table will not exist anymore after the statement. If the table was used with  
LOCK TABLES  
it will be unlocked.
- One can't use  
OR REPLACE  
together with  
IF EXISTS
  - Slaves in replication will by default use  
CREATE OR REPLACE  
when replicating  
CREATE  
statements that don't use  
IF EXISTS  
. This can be changed by setting the variable [slave-ddl-exec-mode](#) to  
STRICT

## CREATE TABLE IF NOT EXISTS

If the  
IF NOT EXISTS  
clause is used, then the table will only be created if a table with the same name does not already exist. If the table already exists, then a warning will be triggered by default.

## CREATE TEMPORARY TABLE

Use the  
TEMPORARY  
keyword to create a temporary table that is only available to the current session. Temporary tables are dropped when the session ends. Temporary table names are specific to the session. They will not conflict with other temporary tables from other sessions even if they share the same name. They will shadow names of non-temporary tables or views, if they are identical. A temporary table can have the same name as a non-temporary table which is located in the same database. In that case, their name will reference the temporary table when used in SQL statements. You must have the [CREATE TEMPORARY TABLES](#) privilege on the database to create temporary tables. If no storage engine is specified, the [default\\_tmp\\_storage\\_engine](#) setting will determine the engine.

[ROCKSDB](#) temporary tables cannot be created by setting the [default\\_tmp\\_storage\\_engine](#) system variable, or using  
CREATE TEMPORARY TABLE LIKE  
. Before [MariaDB 10.7](#), they could be specified, but would silently fail, and a MyISAM table would be created instead. From [MariaDB 10.7](#) an error is returned. Explicitly creating a temporary table with  
ENGINE=ROCKSDB  
has never been permitted.

## CREATE TABLE ... LIKE

Use the  
LIKE  
clause instead of a full table definition to create a table with the same definition as another table, including columns, indexes, and table options. Foreign key definitions, as well as any DATA DIRECTORY or INDEX DIRECTORY table options specified on the original table, will not be created.

## CREATE TABLE ... SELECT

You can create a table containing data from other tables using the

```
CREATE ... SELECT  

statement. Columns will be created in the table for each field returned by the  

SELECT  

query.
```

You can also define some columns normally and add other columns from a

```
SELECT  

. You can also create columns in the normal way and assign them some values using the query, this is done to force a certain type or other field characteristics. The columns that are not named in the query will be placed before the others. For example:
```

```
CREATE TABLE test (a INT NOT NULL, b CHAR(10)) ENGINE=MyISAM  

SELECT 5 AS b, c, d FROM another_table;
```

Remember that the query just returns data. If you want to use the same indexes, or the same columns attributes ( [NOT] NULL

```

,
DEFAULT
,
AUTO_INCREMENT
) in the new table, you need to specify them manually. Types and sizes are not automatically preserved if no data returned by the
SELECT
requires the full size, and
VARCHAR
could be converted into
CHAR
. The CAST\(\) function can be used to forcee the new table to use certain types.

```

Aliases (

```

AS
) are taken into account, and they should always be used when you
SELECT
an expression (function, arithmetical operation, etc).

```

If an error occurs during the query, the table will not be created at all.

If the new table has a primary key or

```

UNIQUE
indexes, you can use the IGNORE or
REPLACE
keywords to handle duplicate key errors during the query.
IGNORE
means that the newer values must not be inserted an identical value exists in the index.
REPLACE
means that older values must be overwritten.

```

If the columns in the new table are more than the rows returned by the query, the columns populated by the query will be placed after other columns. Note that if the strict

```

SQL_MODE
is on, and the columns that are not names in the query do not have a
DEFAULT
value, an error will raise and no rows will be copied.

```

[Concurrent inserts](#) are not used during the execution of a

```
CREATE ... SELECT
```

If the table already exists, an error similar to the following will be returned:

```
ERROR 1050 (42S01): Table 't' already exists
```

If the

```
IF NOT EXISTS
clause is used and the table exists, a note will be produced instead of an error.
```

To insert rows from a query into an existing table, [INSERT ... SELECT](#) can be used.

## Column Definitions

```

create_definition:
{ col_name column_definition | index\_definition | period\_definition | CHECK (expr) }

column_definition:
data\_type
[NOT NULL | NULL] [DEFAULT default_value | (expression)]
[ON UPDATE [NOW | CURRENT_TIMESTAMP] [(precision)]]
[AUTO_INCREMENT] [ZEROFILL] [UNIQUE [KEY] | [PRIMARY] KEY]
[INVISIBLE] [{WITH|WITHOUT} SYSTEM VERSIONING]
[COMMENT 'string'] [REF_SYSTEM_ID = value]
[reference\_definition]
| data\_type [GENERATED ALWAYS]
AS { { ROW {START|END} } | { (expression) [VIRTUAL | PERSISTENT | STORED] } }
[UNIQUE [KEY]] [COMMENT 'string']

constraint_definition:
CONSTRAINT [constraint_name] CHECK (expression)

```

**Note:** Until MariaDB 10.4 , MariaDB accepts the shortcut format with a REFERENCES clause only in ALTER TABLE and CREATE TABLE

statements, but that syntax does nothing. For example:

```
CREATE TABLE b(for_key INT REFERENCES a(not_key));
```

MariaDB simply parses it without returning any error or warning, for compatibility with other DBMS's. Before MariaDB 10.2.1 this was also true for  
CHECK  
constraints. However, only the syntax described below creates foreign keys.

From MariaDB 10.5 , MariaDB will attempt to apply the constraint. See [Foreign Keys examples](#) .

Each definition either creates a column in the table or specifies and index or constraint on one or more columns. See [Indexes](#) below for details on creating indexes.

Create a column by specifying a column name and a data type, optionally followed by column options. See [Data Types](#) for a full list of data types allowed in MariaDB.

## NULL and NOT NULL

Use the

NULL

or

NOT NULL

options to specify that values in the column may or may not be

NULL

, respectively. By default, values may be

NULL

. See also [NULL Values in MariaDB](#) .

## DEFAULT Column Option

MariaDB starting with 10.2.1

The

DEFAULT

clause was enhanced in MariaDB 10.2.1 . Some enhancements include

- BLOB and TEXT columns now support

DEFAULT

- The

DEFAULT

clause can now be used with an expression or function.

Specify a default value using the

DEFAULT

clause. If you don't specify

DEFAULT

then the following rules apply:

- If the column is not defined with

NOT NULL

,

AUTO\_INCREMENT

or

TIMESTAMP

, an explicit

DEFAULT NULL

will be added. Note that in MySQL and in MariaDB before 10.1.6, you may get an explicit

DEFAULT

for primary key parts, if not specified with NOT NULL.

The default value will be used if you [INSERT](#) a row without specifying a value for that column, or if you specify [DEFAULT](#) for that column. Before MariaDB 10.2.1 you couldn't usually provide an expression or function to evaluate at insertion time. You had to provide a constant default value instead. The one exception is that you may use [CURRENT\\_TIMESTAMP](#) as the default value for a [TIMESTAMP](#) column to use the current timestamp at insertion time.

[CURRENT\\_TIMESTAMP](#) may also be used as the default value for a [DATETIME](#)

From MariaDB 10.2.1 you can use most functions in

DEFAULT

. Expressions should have parentheses around them. If you use a non deterministic function in

DEFAULT

then all inserts to the table will be [replicated](#) in [row mode](#) . You can even refer to earlier columns in the

DEFAULT

expression (excluding  
AUTO\_INCREMENT  
columns):

```
CREATE TABLE t1 (a int DEFAULT (1+1), b int DEFAULT (a+1));  
CREATE TABLE t2 (a bigint primary key DEFAULT UUID_SHORT());
```

The  
DEFAULT  
clause cannot contain any [stored functions](#) or [subqueries](#), and a column used in the clause must already have been defined earlier in the statement.

Since [MariaDB 10.2.1](#), it is possible to assign [BLOB](#) or [TEXT](#) columns a  
DEFAULT  
value. In earlier versions, assigning a default to these columns was not possible.

MariaDB starting with [10.3.3](#)

Starting from 10.3.3 you can also use [DEFAULT \( NEXT VALUE FOR sequence \)](#)

## AUTO\_INCREMENT Column Option

Use [AUTO\\_INCREMENT](#) to create a column whose value can be set automatically from a simple counter. You can only use

AUTO\_INCREMENT  
on a column with an integer type. The column must be a key, and there can only be one  
AUTO\_INCREMENT  
column in a table. If you insert a row without specifying a value for that column (or if you specify

0

,

NULL

, or [DEFAULT](#) as the value), the actual value will be taken from the counter, with each insertion incrementing the counter by one. You can still insert a value explicitly. If you insert a value that is greater than the current counter value, the counter is set based on the new value. An

AUTO\_INCREMENT  
column is implicitly  
NOT NULL

. Use [LAST\\_INSERT\\_ID](#) to get the [AUTO\\_INCREMENT](#) value most recently used by an [INSERT](#) statement.

## ZEROFILL Column Option

If the

ZEROFILL  
column option is specified for a column using a [numeric](#) data type, then the column will be set to  
UNSIGNED  
and the spaces used by default to pad the field are replaced with zeros.  
ZEROFILL  
is ignored in expressions or as part of a [UNION](#).  
ZEROFILL  
is a non-standard MySQL and MariaDB enhancement.

## PRIMARY KEY Column Option

Use  
PRIMARY KEY  
to make a column a primary key. A primary key is a special type of a unique key. There can be at most one primary key per table, and it is implicitly  
NOT NULL

Specifying a column as a unique key creates a unique index on that column. See the [Index Definitions](#) section below for more information.

## UNIQUE KEY Column Option

Use  
UNIQUE KEY  
(or just  
UNIQUE  
) to specify that all values in the column must be distinct from each other. Unless the column is  
NOT NULL  
, there may be multiple rows with

NULL  
in the column.

Specifying a column as a unique key creates a unique index on that column. See the [Index Definitions](#) section below for more information.

## COMMENT Column Option

You can provide a comment for each column using the

COMMENT  
clause. The maximum length is 1024 characters. Use the [SHOW FULL COLUMNS](#) statement to see column comments.

## REF\_SYSTEM\_ID

REF\_SYSTEM\_ID  
can be used to specify Spatial Reference System IDs for spatial data type columns.

## Generated Columns

A generated column is a column in a table that cannot explicitly be set to a specific value in a [DML query](#). Instead, its value is automatically generated based on an expression. This expression might generate the value based on the values of other columns in the table, or it might generate the value by calling [built-in functions](#) or [user-defined functions \(UDFs\)](#).

There are two types of generated columns:

- PERSISTENT  
or  
STORED  
: This type's value is actually stored in the table.
- VIRTUAL  
: This type's value is not stored at all. Instead, the value is generated dynamically when the table is queried. This type is the default.

Generated columns are also sometimes called computed columns or virtual columns.

For a complete description about generated columns and their limitations, see [Generated \(Virtual and Persistent/Stored\) Columns](#).

## COMPRESSED

MariaDB starting with 10.3.3

Certain columns may be compressed. See [Storage-Engine Independent Column Compression](#).

## INVISIBLE

MariaDB starting with 10.3.3

Columns may be made invisible, and hidden in certain contexts. See [Invisible Columns](#).

## WITH SYSTEM VERSIONING Column Option

MariaDB starting with 10.3.4

Columns may be explicitly marked as included from system versioning. See [System-versioned tables](#) for details.

## WITHOUT SYSTEM VERSIONING Column Option

MariaDB starting with 10.3.4

Columns may be explicitly marked as excluded from system versioning. See [System-versioned tables](#) for details.

## Index Definitions

```

index_definition:
  {INDEX|KEY} [index_name] [index_type] (index_col_name,...) [index_option] ...
  | {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...) [index_option] ...
  | [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index_name] [index_type] (index_col_name,...) [index_option] ...
  | [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name,...) reference_definition

index_col_name:
  col_name [(length)] [ASC | DESC]

index_type:
  USING {BTREE | HASH | RTREE}

index_option:
  [ KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | CLUSTERING={YES| NO} ]
  [ IGNORED | NOT IGNORED ]

reference_definition:
  REFERENCES tbl_name (index_col_name,...)
  [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
  [ON DELETE reference_option]
  [ON UPDATE reference_option]

reference_option:
  RESTRICT | CASCADE | SET NULL | NO ACTION

```

INDEX  
and  
KEY  
are synonyms.

Index names are optional, if not specified an automatic name will be assigned. Index name are needed to drop indexes and appear in error messages when a constraint is violated.

## Index Categories

### Plain Indexes

Plain indexes are regular indexes that are not unique, and are not acting as a primary key or a foreign key. They are also not the "specialized"

FULLTEXT  
or  
SPATIAL  
indexes.

See [Getting Started with Indexes: Plain Indexes](#) for more information.

### PRIMARY KEY

For

PRIMARY KEY  
indexes, you can specify a name for the index, but it is ignored, and the name of the index is always  
PRIMARY  
. From [MariaDB 10.3.18](#) and [MariaDB 10.4.8](#), a warning is explicitly issued if a name is specified. Before then, the name was silently ignored.

See [Getting Started with Indexes: Primary Key](#) for more information.

### UNIQUE

The

UNIQUE  
keyword means that the index will not accept duplicated values, except for NULLs. An error will raise if you try to insert duplicate values in a  
UNIQUE index.

For

UNIQUE  
indexes, you can specify a name for the constraint, using the  
CONSTRAINT

keyword. That name will be used in error messages.

See [Getting Started with Indexes: Unique Index](#) for more information.

## FOREIGN KEY

For

FOREIGN KEY  
indexes, a reference definition must be provided.

For

FOREIGN KEY  
indexes, you can specify a name for the constraint, using the  
CONSTRAINT  
keyword. That name will be used in error messages.

First, you have to specify the name of the target (parent) table and a column or a column list which must be indexed and whose values must match to the foreign key's values. The

MATCH  
clause is accepted to improve the compatibility with other DBMS's, but has no meaning in MariaDB. The  
ON DELETE  
and  
ON UPDATE  
clauses specify what must be done when a  
DELETE  
(or a  
REPLACE  
) statements attempts to delete a referenced row from the parent table, and when an  
UPDATE  
statement attempts to modify the referenced foreign key columns in a parent table row, respectively. The following options are allowed:

- RESTRICT  
: The delete/update operation is not performed. The statement terminates with a 1451 error (SQLSTATE '2300').
- NO ACTION  
: Synonym for  
RESTRICT
- CASCADE  
: The delete/update operation is performed in both tables.
- SET NULL  
: The update or delete goes ahead in the parent table, and the corresponding foreign key fields in the child table are set to  
NULL  
. (They must not be defined as  
NOT NULL  
for this to succeed).
- SET DEFAULT  
: This option is currently implemented only for the PBXT storage engine, which is disabled by default and no longer maintained. It sets the  
child table's foreign key fields to their  
DEFAULT  
values when the referenced parent table key entries are updated or deleted.

If either clause is omitted, the default behavior for the omitted clause is

RESTRICT

See [Foreign Keys](#) for more information.

## FULLTEXT

Use the

FULLTEXT  
keyword to create full-text indexes.

See [Full-Text Indexes](#) for more information.

## SPATIAL

Use the

SPATIAL

keyword to create geometric indexes.

See [SPATIAL INDEX](#) for more information.

## Index Options

### KEY\_BLOCK\_SIZE Index Option

The

KEY\_BLOCK\_SIZE  
index option is similar to the [KEY\\_BLOCK\\_SIZE](#) table option.

With the [InnoDB](#) storage engine, if you specify a non-zero value for the

KEY\_BLOCK\_SIZE  
table option for the whole table, then the table will implicitly be created with the [ROW\\_FORMAT](#) table option set to  
COMPRESSED

. However, this does not happen if you just set the

KEY\_BLOCK\_SIZE  
index option for one or more indexes in the table. The [InnoDB](#) storage engine ignores the  
KEY\_BLOCK\_SIZE  
index option. However, the [SHOW CREATE TABLE](#) statement may still report it for the index.

For information about the

KEY\_BLOCK\_SIZE  
index option, see the [KEY\\_BLOCK\\_SIZE](#) table option below.

### Index Types

Each storage engine supports some or all index types. See [Storage Engine Index Types](#) for details on permitted index types for each storage engine.

Different index types are optimized for different kind of operations:

- BTREE  
is the default type, and normally is the best choice. It is supported by all storage engines. It can be used to compare a column's value with a value using the =, >, >=, <, <=,  
BETWEEN  
, and  
LIKE  
operators.  
BTREE  
can also be used to find  
NULL  
values. Searches against an index prefix are possible.
- HASH  
is only supported by the MEMORY storage engine.  
HASH  
indexes can only be used for =, <=, and >= comparisons. It can not be used for the  
ORDER BY  
clause. Searches against an index prefix are not possible.
- RTREE  
is the default for  
SPATIAL  
indexes, but if the storage engine does not support it  
BTREE  
can be used.

Index columns names are listed between parenthesis. After each column, a prefix length can be specified. If no length is specified, the whole column will be indexed.

ASC  
and  
DESC  
can be specified for compatibility with are DBMS's, but have no meaning in MariaDB.

### WITH PARSER Index Option

The

WITH PARSER  
index option only applies to [FULLTEXT](#) indexes and contains the fulltext parser name. The fulltext parser must be an installed plugin.

### COMMENT Index Option

A comment of up to 1024 characters is permitted with the

```
COMMENT  
index option.
```

The

```
COMMENT  
index option allows you to specify a comment with user-readable text describing what the index is for. This information is not used by the server itself.
```

## CLUSTERING Index Option

The

```
CLUSTERING  
index option is only valid for tables using the Tokudb storage engine.
```

## IGNORED / NOT IGNORED

MariaDB starting with [10.6.0](#)

From [MariaDB 10.6.0](#), indexes can be specified to be ignored by the optimizer. See [Ignored Indexes](#).

## Periods

MariaDB starting with [10.3.4](#)

```
period_definition:  
    PERIOD FOR SYSTEM_TIME (start_column_name, end_column_name)
```

MariaDB supports a subset of the standard syntax for periods. At the moment it's only used for creating [System-versioned tables](#). Both columns must be created, must be either of a

```
TIMESTAMP(6)  
or  
BIGINT UNSIGNED  
type, and be generated as  
ROW START  
and  
ROW END  
accordingly. See System-versioned tables for details.
```

The table must also have the

```
WITH SYSTEM VERSIONING  
clause.
```

## Constraint Expressions

MariaDB starting with [10.2.1](#)

[MariaDB 10.2.1](#) introduced new ways to define a constraint.

Note: Before [MariaDB 10.2.1](#), constraint expressions were accepted in the syntax but ignored.

[MariaDB 10.2.1](#) introduced two ways to define a constraint:

- ```
CHECK(expression)  
given as part of a column definition.
```
- ```
CONSTRAINT [constraint_name] CHECK (expression)
```

Before a row is inserted or updated, all constraints are evaluated in the order they are defined. If any constraints fails, then the row will not be updated. One can use most deterministic functions in a constraint, including [UDFs](#).

```
create table t1 (a int check(a>0) ,b int check (b> 0), constraint abc check (a>b));
```

If you use the second format and you don't give a name to the constraint, then the constraint will get a auto generated name. This is done so that you can later delete the constraint with [ALTER TABLE DROP constraint\\_name](#).

One can disable all constraint expression checks by setting the variable  
`check_constraint_checks`

to  
OFF  
. This is useful for example when loading a table that violates some constraints that you want to later find and fix in SQL.

See [CONSTRAINT](#) for more information.

## Table Options

For each individual table you create (or alter), you can set some table options. The general syntax for setting options is:

```
<OPTION_NAME> = <option_value>, [<OPTION_NAME> = <option_value> ...]
```

The equal sign is optional.

Some options are supported by the server and can be used for all tables, no matter what storage engine they use; other options can be specified for all storage engines, but have a meaning only for some engines. Also, engines can [extend](#)

```
CREATE TABLE  
  with new options .
```

If the

```
IGNORE_BAD_TABLE_OPTIONS  
SQL_MODE is enabled, wrong table options generate a warning; otherwise, they generate an error.
```

```
table_option:  
  [STORAGE] ENGINE [=] engine_name  
  | AUTO_INCREMENT [=] value  
  | AVG_ROW_LENGTH [=] value  
  | [DEFAULT] CHARACTER SET [=] charset_name  
  | CHECKSUM [=] {0 | 1}  
  | [DEFAULT] COLLATE [=] collation_name  
  | COMMENT [=] 'string'  
  | CONNECTION [=] 'connect_string'  
  | DATA DIRECTORY [=] 'absolute path to directory'  
  | DELAY_KEY_WRITE [=] {0 | 1}  
  | ENCRYPTED [=] {YES | NO}  
  | ENCRYPTION_KEY_ID [=] value  
  | IETF_QUOTES [=] {YES | NO}  
  | INDEX DIRECTORY [=] 'absolute path to directory'  
  | INSERT_METHOD [=] {NO | FIRST | LAST}  
  | KEY_BLOCK_SIZE [=] value  
  | MAX_ROWS [=] value  
  | MIN_ROWS [=] value  
  | PACK_KEYS [=] {0 | 1 | DEFAULT}  
  | PAGE_CHECKSUM [=] {0 | 1}  
  | PAGE_COMPRESSED [=] {0 | 1}  
  | PAGE_COMPRESSION_LEVEL [=] {0 .. 9}  
  | PASSWORD [=] 'string'  
  | ROW_FORMAT [=] {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT|PAGE}  
  | SEQUENCE [=] {0|1}  
  | STATS_AUTO_RECALC [=] {DEFAULT|0|1}  
  | STATS_PERSISTENT [=] {DEFAULT|0|1}  
  | STATS_SAMPLE_PAGES [=] {DEFAULT|value}  
  | TABLESPACE tablespace_name  
  | TRANSACTIONAL [=] {0 | 1}  
  | UNION [=] (tbl_name[,tbl_name]...)  
  | WITH SYSTEM VERSIONING
```

## [STORAGE] ENGINE

```
[STORAGE] ENGINE
```

specifies a [storage engine](#) for the table. If this option is not used, the default storage engine is used instead. That is, the [default\\_storage\\_engine](#) session option value if it is set, or the value specified for the

```
--default-storage-engine
```

[mysqld startup option](#), or the default storage engine, [InnoDB](#). If the specified storage engine is not installed and active, the default value will be used, unless the

```
NO_ENGINE_SUBSTITUTION
```

[SQL MODE](#) is set (default). This is only true for

```
CREATE TABLE
```

, not for

```
ALTER TABLE
```

. For a list of storage engines that are present in your server, issue a [SHOW ENGINES](#).

## AUTO\_INCREMENT

AUTO\_INCREMENT  
specifies the initial value for the  
AUTO\_INCREMENT  
primary key. This works for MyISAM, Aria, InnoDB/XtraDB, MEMORY, and ARCHIVE tables. You can change this option with  
ALTER TABLE  
, but in that case the new value must be higher than the highest value which is present in the  
AUTO\_INCREMENT  
column. If the storage engine does not support this option, you can insert (and then delete) a row having the wanted value - 1 in the  
AUTO\_INCREMENT  
column.

## AVG\_ROW\_LENGTH

AVG\_ROW\_LENGTH  
is the average rows size. It only applies to tables using [MyISAM](#) and [Aria](#) storage engines that have the ROW\_FORMAT table option set to  
FIXED  
format.

MyISAM uses

MAX\_ROWS  
and  
AVG\_ROW\_LENGTH  
to decide the maximum size of a table (default: 256TB, or the maximum file size allowed by the system).

## [DEFAULT] CHARACTER SET/CHARSET

[DEFAULT] CHARACTER SET  
(or  
[DEFAULT] CHARSET  
) is used to set a default character set for the table. This is the character set used for all columns where an explicit character set is not specified.  
If this option is omitted or  
DEFAULT  
is specified, database's default character set will be used. See [Setting Character Sets and Collations](#) for details on setting the character sets .

## CHECKSUM/TABLE\_CHECKSUM

CHECKSUM  
(or  
TABLE\_CHECKSUM  
) can be set to 1 to maintain a live checksum for all table's rows. This makes write operations slower, but  
CHECKSUM TABLE  
will be very fast. This option is only supported for [MyISAM](#) and [Aria tables](#) .

## [DEFAULT] COLLATE

[DEFAULT] COLLATE  
is used to set a default collation for the table. This is the collation used for all columns where an explicit character set is not specified. If this option is omitted or  
DEFAULT  
is specified, database's default option will be used. See [Setting Character Sets and Collations](#) for details on setting the collations

## COMMENT

COMMENT  
is a comment for the table. The maximum length is 2048 characters. Also used to define table parameters when creating a [Spider](#) table.

## CONNECTION

CONNECTION

is used to specify a server name or a connection string for a [Spider](#) , [CONNECT](#) , [Federated](#) or [FederatedX](#) table .

## DATA DIRECTORY/INDEX DIRECTORY

DATA DIRECTORY  
and  
INDEX DIRECTORY

are supported for MyISAM and Aria, and DATA DIRECTORY is also supported by InnoDB if the `innodb_file_per_table` server system variable is enabled, but only in CREATE TABLE, not in [ALTER TABLE](#) . So, carefully choose a path for InnoDB tables at creation time, because it cannot be changed without dropping and re-creating the table. These options specify the paths for data files and index files, respectively. If these options are omitted, the database's directory will be used to store data files and index files. Note that these table options do not work for [partitioned](#) tables (use the partition options instead), or if the server has been invoked with the

```
--skip-symbolic-links  
startup option . To avoid the overwriting of old files with the same name that could be present in the directories, you can use the  
--keep_files_on_create  
option (an error will be issued if files already exist). These options are ignored if the  
NO_DIR_IN_CREATE  
SQL_MODE is enabled (useful for replication slaves). Also note that symbolic links cannot be used for InnoDB tables.
```

DATA DIRECTORY

works by creating symlinks from where the table would normally have been (inside the `datadir` ) to where the option specifies. For security reasons, to avoid bypassing the privilege system, the server does not permit symlinks inside the datadir. Therefore,

DATA DIRECTORY

cannot be used to specify a location inside the datadir. An attempt to do so will result in an error

1210 (HY000) Incorrect arguments to DATA DIRECTORY

## DELAY\_KEY\_WRITE

DELAY\_KEY\_WRITE

is supported by MyISAM and Aria, and can be set to 1 to speed up write operations. In that case, when data are modified, the indexes are not updated until the table is closed. Writing the changes to the index file altogether can be much faster. However, note that this option is applied only if the `delay_key_write` server variable is set to 'ON'. If it is 'OFF' the delayed index writes are always disabled, and if it is 'ALL' the delayed index writes are always used, disregarding the value of

DELAY\_KEY\_WRITE

## ENCRYPTED

The

ENCRYPTED

table option can be used to manually set the encryption status of an InnoDB table. See [InnoDB Encryption](#) for more information.

Aria does not support the

ENCRYPTED

table option. See [MDEV-18049](#) .

See [Data-at-Rest Encryption](#) for more information.

## ENCRYPTION\_KEY\_ID

The

ENCRYPTION\_KEY\_ID

table option can be used to manually set the encryption key of an InnoDB table. See [InnoDB Encryption](#) for more information.

Aria does not support the

ENCRYPTION\_KEY\_ID

table option. See [MDEV-18049](#) .

See [Data-at-Rest Encryption](#) for more information.

## IETF\_QUOTES

For the `CSV` storage engine, the

IETF\_QUOTES

option, when set to

YES

, enables IETF-compatible parsing of embedded quote and comma characters. Enabling this option for a table improves compatibility with other tools that use CSV, but is not compatible with MySQL CSV tables, or MariaDB CSV tables created without this option. Disabled by default.

## INSERT\_METHOD

INSERT\_METHOD

is only used with [MERGE](#) tables. This option determines in which underlying table the new rows should be inserted. If you set it to 'NO' (which is the default) no new rows can be added to the table (but you will still be able to perform

INSERT

s directly against the underlying tables).

FIRST

means that the rows are inserted into the first table, and

LAST

means that they are inserted into the last table.

## KEY\_BLOCK\_SIZE

KEY\_BLOCK\_SIZE

is used to determine the size of key blocks, in bytes or kilobytes. However, this value is just a hint, and the storage engine could modify or ignore it. If

KEY\_BLOCK\_SIZE

is set to 0, the storage engine's default value will be used.

With the [InnoDB](#) storage engine, if you specify a non-zero value for the

KEY\_BLOCK\_SIZE

table option for the whole table, then the table will implicitly be created with the [ROW\\_FORMAT](#) table option set to COMPRESSED

## MIN\_ROWS/MAX\_ROWS

MIN\_ROWS

and

MAX\_ROWS

let the storage engine know how many rows you are planning to store as a minimum and as a maximum. These values will not be used as real limits, but they help the storage engine to optimize the table.

MIN\_ROWS

is only used by [MEMORY](#) storage engine to decide the minimum memory that is always allocated.

MAX\_ROWS

is used to decide the minimum size for indexes.

## PACK\_KEYS

PACK\_KEYS

can be used to determine whether the indexes will be compressed. Set it to 1 to compress all keys. With a value of 0, compression will not be used. With the

DEFAULT

value, only long strings will be compressed. Uncompressed keys are faster.

## PAGE\_CHECKSUM

PAGE\_CHECKSUM

is only applicable to [Aria](#) tables, and determines whether indexes and data should use page checksums for extra safety.

## PAGE\_COMPRESSED

PAGE\_COMPRESSED

is used to enable [InnoDB](#) page compression for [InnoDB](#) tables.

## PAGE\_COMPRESSION\_LEVEL

#### PAGE\_COMPRESSION\_LEVEL

is used to set the compression level for [InnoDB page compression](#) for [InnoDB](#) tables. The table must also have the [PAGE\\_COMPRESSED](#) table option set to

1

Valid values for

#### PAGE\_COMPRESSION\_LEVEL

are 1 (the best speed) through 9 (the best compression), .

## PASSWORD

#### PASSWORD

is unused.

## RAID\_TYPE

#### RAID\_TYPE

is an obsolete option, as the raid support has been disabled since MySQL 5.0.

## ROW\_FORMAT

The

#### ROW\_FORMAT

table option specifies the row format for the data file. Possible values are engine-dependent.

### Supported MyISAM Row Formats

For [MyISAM](#), the supported row formats are:

- FIXED
- DYNAMIC
- COMPRESSED

The

#### COMPRESSED

row format can only be set by the [myisampack](#) command line tool.

See [MyISAM Storage Formats](#) for more information.

### Supported Aria Row Formats

For [Aria](#), the supported row formats are:

- PAGE
- FIXED
- DYNAMIC

See [Aria Storage Formats](#) for more information.

### Supported InnoDB Row Formats

For [InnoDB](#), the supported row formats are:

- COMPACT
- REDUNDANT

- COMPRESSED
- DYNAMIC

If the

`ROW_FORMAT`

table option is set to

`FIXED`

for an InnoDB table, then the server will either return an error or a warning depending on the value of the `innodb_strict_mode` system variable. If the `innodb_strict_mode` system variable is set to

`OFF`

, then a warning is issued, and MariaDB will create the table using the default row format for the specific MariaDB server version. If the `innodb_strict_mode` system variable is set to

`ON`

, then an error will be raised.

See [InnoDB Storage Formats](#) for more information.

## Other Storage Engines and ROW\_FORMAT

Other storage engines do not support the

`ROW_FORMAT`

table option.

## SEQUENCE

MariaDB starting with [10.3](#)

If the table is a `sequence`, then it will have the

`SEQUENCE`

set to

`1`

## STATS\_AUTO\_RECALC

`STATS_AUTO_RECALC`

indicates whether to automatically recalculate persistent statistics (see

`STATS_PERSISTENT`

, below) for an InnoDB table. If set to

`1`

, statistics will be recalculated when more than 10% of the data has changed. When set to

`0`

, stats will be recalculated only when an `ANALYZE TABLE` is run. If set to

`DEFAULT`

, or left out, the value set by the `innodb_stats_auto_recalc` system variable applies. See [InnoDB Persistent Statistics](#).

## STATS\_PERSISTENT

`STATS_PERSISTENT`

indicates whether the InnoDB statistics created by `ANALYZE TABLE` will remain on disk or not. It can be set to

`1`

(on disk),

`0`

(not on disk, the pre-MariaDB 10 behavior), or

`DEFAULT`

(the same as leaving out the option), in which case the value set by the `innodb_stats_persistent` system variable will apply. Persistent statistics stored on disk allow the statistics to survive server restarts, and provide better query plan stability. See [InnoDB Persistent Statistics](#).

## STATS\_SAMPLE\_PAGES

`STATS_SAMPLE_PAGES`

indicates how many pages are used to sample index statistics. If 0 or DEFAULT, the default value, the `innodb_stats_sample_pages` value is used. See [InnoDB Persistent Statistics](#).

## TRANSACTIONAL

TRANSACTIONAL

is only applicable for Aria tables. In future Aria tables created with this option will be fully transactional, but currently this provides a form of crash protection. See [Aria Storage Engine](#) for more details.

## UNION

UNION

must be specified when you create a MERGE table. This option contains a comma-separated list of MyISAM tables which are accessed by the new table. The list is enclosed between parenthesis. Example:

```
UNION = (t1,t2)
```

## WITH SYSTEM VERSIONING

WITH SYSTEM VERSIONING

is used for creating [System-versioned tables](#).

## Partitions

```
partition_options:  
  PARTITION BY  
    { [LINEAR] HASH(expr)  
    | [LINEAR] KEY(column_list)  
    | RANGE(expr)  
    | LIST(expr)  
    | SYSTEM_TIME [INTERVAL time_quantity time_unit] [LIMIT num] }  
  [PARTITIONS num]  
  [SUBPARTITION BY  
    { [LINEAR] HASH(expr)  
    | [LINEAR] KEY(column_list) }  
    [SUBPARTITIONS num]  
  ]  
  [(partition_definition [, partition_definition] ...)]  
  
partition_definition:  
  PARTITION partition_name  
    [VALUES {LESS THAN {(expr) | MAXVALUE} | IN (value_list)}]  
    [[STORAGE] ENGINE [=] engine_name]  
    [COMMENT [=] 'comment_text']  
    [DATA DIRECTORY [=] 'data_dir']  
    [INDEX DIRECTORY [=] 'index_dir']  
    [MAX_ROWS [=] max_number_of_rows]  
    [MIN_ROWS [=] min_number_of_rows]  
    [TABLESPACE [=] tablespace_name]  
    [NODEGROUP [=] node_group_id]  
    [(subpartition_definition [, subpartition_definition] ...)]  
  
subpartition_definition:  
  SUBPARTITION logical_name  
    [[STORAGE] ENGINE [=] engine_name]  
    [COMMENT [=] 'comment_text']  
    [DATA DIRECTORY [=] 'data_dir']  
    [INDEX DIRECTORY [=] 'index_dir']  
    [MAX_ROWS [=] max_number_of_rows]  
    [MIN_ROWS [=] min_number_of_rows]  
    [TABLESPACE [=] tablespace_name]  
    [NODEGROUP [=] node_group_id]
```

If the

PARTITION BY

clause is used, the table will be [partitioned](#). A partition method must be explicitly indicated for partitions and subpartitions. Partition methods are:

- [LINEAR] HASH  
creates a hash key which will be used to read and write rows. The partition function can be any valid SQL expression which returns an INTEGER  
number. Thus, it is possible to use the HASH  
method on an integer column, or on functions which accept integer columns as an argument. However, VALUES LESS THAN  
and  
VALUES IN  
clauses can not be used with HASH  
. An example:

```
CREATE TABLE t1 (a INT, b CHAR(5), c DATETIME)
PARTITION BY HASH ( YEAR(c) );
```

[LINEAR] HASH  
can be used for subpartitions, too.

- [LINEAR] KEY  
is similar to  
HASH  
, but the index has an even distribution of data. Also, the expression can only be a column or a list of columns.  
VALUES LESS THAN  
and  
VALUES IN  
clauses can not be used with  
KEY

- RANGE partitions the rows using on a range of values, using the  
VALUES LESS THAN  
operator.  
VALUES IN  
is not allowed with  
RANGE  
. The partition function can be any valid SQL expression which returns a single value.
- LIST assigns partitions based on a table's column with a restricted set of possible values. It is similar to  
RANGE  
, but  
VALUES IN  
must be used for at least 1 columns, and  
VALUES LESS THAN  
is disallowed.

- SYSTEM\_TIME  
partitioning is used for [System-versioned tables](#) to store historical data separately from current data.

Only

HASH  
and  
KEY  
can be used for subpartitions, and they can be  
[LINEAR]

It is possible to define up to 1024 partitions and subpartitions.

The number of defined partitions can be optionally specified as

PARTITION count  
. This can be done to avoid specifying all partitions individually. But you can also declare each individual partition and, additionally, specify a PARTITIONS count  
clause; in the case, the number of PARTITION  
s must equal count.

Also see [Partitioning Types Overview](#).

## Sequences

MariaDB starting with [10.3](#)

```
CREATE TABLE  
can also be used to create a SEQUENCE . See CREATE SEQUENCE and Sequence Overview .
```

## Atomic DDL

MariaDB starting with [10.6.1](#)

[MariaDB 10.6.1](#) supports [Atomic DDL](#) .

```
CREATE TABLE  
is atomic, except for  
CREATE OR REPLACE  
, which is only crash safe.
```

## Examples

```
create table if not exists test (  
a bigint auto_increment primary key,  
name varchar(128) charset utf8,  
key name (name(32))  
) engine=InnoDB default charset latin1;
```

This example shows a couple of things:

- Usage of
  - IF NOT EXISTS
    - ; If the table already existed, it will not be created. There will not be any error for the client, just a warning.
- How to create a
  - PRIMARY KEY
    - that is [automatically generated](#) .
- How to specify a table-specific [character set](#) and another for a column.
- How to create an index (
  - name
    - ) that is only partly indexed (to save space).

The following clauses will work from [MariaDB 10.2.1](#) only.

```
CREATE TABLE t1(  
a int DEFAULT (1+1),  
b int DEFAULT (a+1),  
expires DATETIME DEFAULT(NOW() + INTERVAL 1 YEAR),  
x BLOB DEFAULT USER()  
);
```

## See Also

- [Identifier Names](#)
- [ALTER TABLE](#)
- [DROP TABLE](#)
- [Character Sets and Collations](#)
- [SHOW CREATE TABLE](#)
- Storage engines can add their own [attributes for columns, indexes and tables](#) .
- Variable [slave-ddl-exec-mode](#) .

## 1.1.2.1.8 DELETE

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [PARTITION](#)
  2. [FOR PORTION OF](#)
  3. [RETURNING](#)
  4. [Same Source and Target Table](#)
  5. [DELETE HISTORY](#)
3. [Examples](#)
  1. [Deleting from the Same Source and Target](#)
  4. [See Also](#)

## Syntax

Single-table syntax:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
  FROM tbl_name [PARTITION (partition_list)]
  [FOR PORTION OF period FROM expr1 TO expr2]
  [WHERE where_condition]
  [ORDER BY ...]
  [LIMIT row_count]
  [RETURNING select_expr
   [, select_expr ...]]
```

Multiple-table syntax:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
  tbl_name[.*] [, tbl_name[.*]] ...
  FROM table_references
  [WHERE where_condition]
```

Or:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]
  FROM tbl_name[.*] [, tbl_name[.*]] ...
  USING table_references
  [WHERE where_condition]
```

Trimming history:

```
DELETE HISTORY
  FROM tbl_name [PARTITION (partition_list)]
  [BEFORE SYSTEM_TIME [TIMESTAMP|TRANSACTION] expression]
```

## Description

Option	Description
LOW_PRIORITY	Wait until all SELECT's are done before starting the statement. Used with storage engines that uses table locking (MyISAM, Aria etc). See <a href="#">HIGH_PRIORITY and LOW_PRIORITY clauses</a> for details.
QUICK	Signal the storage engine that it should expect that a lot of rows are deleted. The storage engine can do things to speed up the DELETE like ignoring merging of data blocks until all rows are deleted from the block (instead of when a block is half full). This speeds up things at the expense of lost space in data blocks. At least <a href="#">MyISAM</a> and <a href="#">Aria</a> support this feature.
IGNORE	Don't stop the query even if a not-critical error occurs (like data overflow). See <a href="#">How IGNORE works</a> for a full description.

For the single-table syntax, the

```
DELETE
  statement deletes rows from tbl_name and returns a count of the number of deleted rows. This count can be obtained by calling the
ROW_COUNT() function. The
  WHERE
  clause, if given, specifies the conditions that identify which rows to delete. With no
  WHERE
  clause, all rows are deleted. If the ORDER BY clause is specified, the rows are deleted in the order that is specified. The LIMIT clause places a
  limit on the number of rows that can be deleted.
```

For the multiple-table syntax,

```
DELETE
```

deletes from each  
tbl\_name  
the rows that satisfy the conditions. In this case, [ORDER BY](#) and [LIMIT](#) > cannot be used. A  
[DELETE](#)  
can also reference tables which are located in different databases; see [Identifier Qualifiers](#) for the syntax.

where\_condition  
is an expression that evaluates to true for each row to be deleted. It is specified as described in [SELECT](#).

Currently, you cannot delete from a table and select from the same table in a subquery.

You need the

[DELETE](#)  
privilege on a table to delete rows from it. You need only the  
[SELECT](#)  
privilege for any columns that are only read, such as those named in the  
[WHERE](#)  
clause. See [GRANT](#).

As stated, a

[DELETE](#)  
statement with no  
[WHERE](#)  
clause deletes all rows. A faster way to do this, when you do not need to know the number of deleted rows, is to use  
[TRUNCATE TABLE](#)  
. However, within a transaction or if you have a lock on the table,  
[TRUNCATE TABLE](#)  
cannot be used whereas

[DELETE](#)

can. See

[TRUNCATE TABLE](#)

, and

[LOCK](#)

## PARTITION

See [Partition Pruning and Selection](#) for details.

## FOR PORTION OF

MariaDB starting with [10.4.3](#)

See [Application Time Periods - Deletion by Portion](#).

## RETURNING

It is possible to return a resultset of the deleted rows for a single table to the client by using the syntax

`DELETE ... RETURNING select_expr [, select_expr2 ...]`

Any of SQL expression that can be calculated from a single row fields is allowed. Subqueries are allowed. The AS keyword is allowed, so it is possible to use aliases.

The use of aggregate functions is not allowed. RETURNING cannot be used in multi-table DELETEs.

MariaDB starting with [10.3.1](#)

## Same Source and Target Table

Until [MariaDB 10.3.1](#), deleting from a table with the same source and target was not possible. From [MariaDB 10.3.1](#), this is now possible. For example:

```
DELETE FROM t1 WHERE c1 IN (SELECT b.c1 FROM t1 b WHERE b.c2=0);
```

MariaDB starting with 10.3.4

## DELETE HISTORY

One can use

```
DELETE HISTORY  
to delete historical information from System-versioned tables .
```

## Examples

How to use the ORDER BY and LIMIT clauses:

```
DELETE FROM page_hit ORDER BY timestamp LIMIT 1000000;
```

How to use the RETURNING clause:

```
DELETE FROM t RETURNING f1;  
+-----+  
| f1 |  
+-----+  
| 5 |  
| 50 |  
| 500 |  
+-----+
```

The following statement joins two tables: one is only used to satisfy a WHERE condition, but no row is deleted from it; rows from the other table are deleted, instead.

```
DELETE post FROM blog INNER JOIN post WHERE blog.id = post.blog_id;
```

## Deleting from the Same Source and Target

```
CREATE TABLE t1 (c1 INT, c2 INT);  
DELETE FROM t1 WHERE c1 IN (SELECT b.c1 FROM t1 b WHERE b.c2=0);
```

Until MariaDB 10.3.1 , this returned:

```
ERROR 1093 (HY000): Table 't1' is specified twice, both as a target for 'DELETE'  
and as a separate source for
```

From MariaDB 10.3.1 :

```
Query OK, 0 rows affected (0.00 sec)
```

## See Also

- How IGNORE works
- SELECT
- ORDER BY
- LIMIT
- REPLACE ... RETURNING
- INSERT ... RETURNING
- Returning clause (video)

## 1.1.2.1.9 DROP TABLE

### Syntax

```
DROP [TEMPORARY] TABLE [IF EXISTS] /*COMMENT TO SAVE*/  
tbl_name [, tbl_name] ...  
[WAIT n|NOWAIT]  
[RESTRICT | CASCADE]
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [WAIT/NOWAIT](#)
3. [DROP TABLE in replication](#)
4. [Dropping an Internal #sql... Table](#)
5. [Dropping All Tables in a Database](#)
6. [Atomic DROP TABLE](#)
7. [Examples](#)
8. [Notes](#)
9. [See Also](#)

## Description

DROP TABLE

removes one or more tables. You must have the

DROP

privilege for each table. All table data and the table definition are removed, as well as [triggers](#) associated to the table, so be careful with this statement! If any of the tables named in the argument list do not exist, MariaDB returns an error indicating by name which non-existing tables it was unable to drop, but it also drops all of the tables in the list that do exist.

**Important** : When a table is dropped, user privileges on the table are not automatically dropped. See [GRANT](#).

If another thread is using the table in an explicit transaction or an autocommit transaction, then the thread acquires a [metadata lock \(MDL\)](#) on the table. The

DROP TABLE

statement will wait in the "Waiting for table metadata lock" [thread state](#) until the MDL is released. MDLs are released in the following cases:

- If an MDL is acquired in an explicit transaction, then the MDL will be released when the transaction ends.
- If an MDL is acquired in an autocommit transaction, then the MDL will be released when the statement ends.
- Transactional and non-transactional tables are handled the same.

Note that for a partitioned table,

DROP TABLE

permanently removes the table definition, all of its partitions, and all of the data which was stored in those partitions. It also removes the partitioning definition (.par) file associated with the dropped table.

For each referenced table,

DROP TABLE

drops a temporary table with that name, if it exists. If it does not exist, and the

TEMPORARY

keyword is not used, it drops a non-temporary table with the same name, if it exists. The

TEMPORARY

keyword ensures that a non-temporary table will not accidentally be dropped.

Use

IF EXISTS

to prevent an error from occurring for tables that do not exist. A

NOTE

is generated for each non-existent table when using

IF EXISTS

. See [SHOW WARNINGS](#).

If a [foreign key](#) references this table, the table cannot be dropped. In this case, it is necessary to drop the foreign key first.

RESTRICT

and

CASCADE

are allowed to make porting from other database systems easier. In MariaDB, they do nothing.

The comment before the table names (

/\*COMMENT TO SAVE\*/

) is stored in the [binary log](#). That feature can be used by replication tools to send their internal messages.

It is possible to specify table names as

db\_name

.

tab\_name

. This is useful to delete tables from multiple databases with one statement. See [Identifier Qualifiers](#) for details.

The [DROP privilege](#) is required to use

DROP TABLE

on non-temporary tables. For temporary tables, no privilege is required, because such tables are only visible for the current session.

#### Note:

```
DROP TABLE  
automatically commits the current active transaction, unless you use the  
TEMPORARY  
keyword.
```

MariaDB starting with 10.5.4

From MariaDB 10.5.4 ,

```
DROP TABLE  
reliably deletes table remnants inside a storage engine even if the  
.frm  
file is missing. Before then, a missing  
.frm  
file would result in the statement failing.
```

MariaDB starting with 10.3.1

## WAIT/NOWAIT

Set the lock wait timeout. See [WAIT and NOWAIT](#) .

## DROP TABLE in replication

DROP TABLE  
has the following characteristics in [replication](#) :

- DROP TABLE IF EXISTS  
are always logged.
- DROP TABLE  
without  
IF EXISTS  
for tables that don't exist are not written to the [binary log](#) .
- Dropping of  
TEMPORARY  
tables are prefixed in the log with  
TEMPORARY  
. These drops are only logged when running [statement](#) or [mixed mode](#) replication.
- One  
DROP TABLE  
statement can be logged with up to 3 different  
DROP  
statements:
  - DROP TEMPORARY TABLE list\_of\_non\_transactional\_temporary\_tables
  - DROP TEMPORARY TABLE list\_of\_transactional\_temporary\_tables
  - DROP TABLE list\_of\_normal\_tables

DROP TABLE  
on the primary is treated on the replica as  
DROP TABLE IF EXISTS  
. You can change that by setting [slave-ddl-exec-mode](#) to  
STRICT

## Dropping an Internal #sql-... Table

From MariaDB 10.6 , [DROP TABLE](#) is [atomic](#) and the following does not apply. Until MariaDB 10.5 , if the [mariadb/mysqld process](#) is killed during an [ALTER TABLE](#) you may find a table named #sql... in your data directory. In MariaDB 10.3 , InnoDB tables with this prefix will be deleted automatically during startup. From MariaDB 10.4 , these temporary tables will always be deleted automatically.

If you want to delete one of these tables explicitly you can do so by using the following syntax:

```
DROP TABLE `#mysql150##sql-...`;
```

When running an

```
ALTER TABLE..ALGORITHM=INPLACE  
that rebuilds the table, InnoDB will create an internal  
#sql-ib  
table. Until MariaDB 10.3.2 , for these tables, the  
.frm  
file will be called something else. In order to drop such a table after a server crash, you must rename the  
#sql*.frm  
file to match the  
#sql-ib*.ibd  
file.
```

From MariaDB 10.3.3 , the same name as the .frm file is used for the intermediate copy of the table. The #sql-ib names are used by TRUNCATE and delayed DROP.

From MariaDB 10.2.19 and MariaDB 10.3.10 , the #sql-ib tables will be deleted automatically.

## Dropping All Tables in a Database

The best way to drop all tables in a database is by executing

```
DROP DATABASE
```

, which will drop the database itself, and all tables in it.

However, if you want to drop all tables in the database, but you also want to keep the database itself and any other non-table objects in it, then you would need to execute

```
DROP TABLE  
to drop each individual table. You can construct these  
DROP TABLE  
commands by querying the
```

```
TABLES
```

table in the

```
information_schema
```

database. For example:

```
SELECT CONCAT('DROP TABLE IF EXISTS `', TABLE_SCHEMA, '`.', TABLE_NAME, '`;')  
FROM information_schema.TABLES  
WHERE TABLE_SCHEMA = 'mydb';
```

## Atomic DROP TABLE

MariaDB starting with 10.6.1

From MariaDB 10.6 ,

```
DROP TABLE  
for a single table is atomic ( MDEV-25180 ) for most engines, including InnoDB, MyRocks, MyISAM and Aria.
```

This means that if there is a crash (server down or power outage) during

```
DROP TABLE  
, all tables that have been processed so far will be completely dropped, including related trigger files and status entries, and the binary log will include a
```

```
DROP TABLE  
statement for the dropped tables. Tables for which the drop had not started will be left intact.
```

In older MariaDB versions, there was a small chance that, during a server crash happening in the middle of

```
DROP TABLE  
, some storage engines that were using multiple storage files, like MyISAM , could have only a part of its internal files dropped.
```

In MariaDB 10.5 ,

```
DROP TABLE  
was extended to be able to delete a table that was only partly dropped ( MDEV-11412 ) as explained above. Atomic  
DROP TABLE  
is the final piece to make
```

```
DROP TABLE  
    fully reliable.  
  
Dropping multiple tables is crash-safe.  
See Atomic DDL for more information.
```

## Examples

```
DROP TABLE Employees, Customers;
```

## Notes

Beware that

```
DROP TABLE  
can drop both tables and sequences. This is mainly done to allow old tools like mysqldump to work with sequences.
```

## See Also

- [CREATE TABLE](#)
- [ALTER TABLE](#)
- [SHOW CREATE TABLE](#)
- [DROP SEQUENCE](#)
- Variable [slave-ddl-exec-mode](#).

## 1.1.2.1.10 Installing System Tables (`mysql_install_db`)

`mysql_install_db`  
initializes the MariaDB data directory and creates the [system tables](#) in the

`mysql`

database, if they do not exist. MariaDB uses these tables to manage [privileges](#), [roles](#), and [plugins](#). It also uses them to provide the data for the

`help`

command in the

`mysql`

client.

`mysql_install_db`

works by starting MariaDB Server's  
`mysqld`  
process in

`--bootstrap`

mode and sending commands to create the [system tables](#) and their content.

There is a version specifically for Windows,

`mysql_install_db.exe`

To invoke

`mysql_install_db`  
, use the following syntax:

```
mysql_install_db --user=mysql
```

For the options supported by

`mysql_install_db`

, see [mysql\\_install\\_db: Options](#) .

For the option groups read by

`mysql_install_db`

, see [mysql\\_install\\_db: Option Groups](#) .

See [mysql\\_install\\_db: Installing System Tables](#) for information on the installation process.

See [mysql\\_install\\_db: Troubleshooting Issues](#) for information on how to troubleshoot the installation process.

## See also:

- [mysql\\_install\\_db](#)
- The Windows version of  
`mysql_install_db`  
:

`mysql_install_db.exe`

## 1.1.2.1.11 mysqlcheck

MariaDB starting with 10.4.6

From [MariaDB 10.4.6](#),

`mariadb-check`  
is a symlink to  
`mysqlcheck`

.

MariaDB starting with 10.5.2

From [MariaDB 10.5.2](#),

`mariadb-check`  
is the name of the tool, with  
`mysqlcheck`  
a symlink .

### Contents

1. [Using mysqlcheck](#)
  1. [Options](#)
  2. [Option Files](#)
    1. [Option Groups](#)
2. [Notes](#)
  1. [Default Values](#)
  2. [mysqlcheck and auto-repair](#)
  3. [mysqlcheck and all-databases](#)
  4. [mysqlcheck and verbose](#)

`mysqlcheck`

is a maintenance tool that allows you to check, repair, analyze and optimize multiple tables from the command line.

It is essentially a commandline interface to the [CHECK TABLE](#) , [REPAIR TABLE](#) , [ANALYZE TABLE](#) and [OPTIMIZE TABLE](#) commands, and so, unlike [myisamchk](#) and [aria\\_chk](#) , requires the server to be running.

This tool does not work with partitioned tables.

## Using mysqlcheck

```
./client/mysqlcheck [OPTIONS] database [tables]
```

OR

```
./client/mysqlcheck [OPTIONS] --databases DB1 [DB2 DB3...]
```

OR

```
./client/mysqlcheck [OPTIONS] --all-databases
```

#### mysqlcheck

can be used to CHECK (-c, -m, -C), REPAIR (-r), ANALYZE (-a), or OPTIMIZE (-o) tables. Some of the options (like -e or -q) can be used at the same time. Not all options are supported by all storage engines.

The -c, -r, -a and -o options are exclusive to each other.

The option

--check

will be used by default, if no other options were specified. You can change the default behavior by making a symbolic link to the binary, or copying it somewhere with another name, the alternatives are:

mysqlrepair	The default option will be -r ( --repair )
mysqlanalyze	The default option will be -a ( --analyze )
mysqloptimize	The default option will be -o ( --optimize )

## Options

#### mysqlcheck

supports the following options:

Option	Description
-A ,-- all-databases	Check all the databases. This is the same as --databases with all databases selected.
-1 ,-- all-in-1	Instead of issuing one query for each table, use one query per database, naming all tables in the database in a comma-separated list.
-a ,-- analyze	Analyze given tables.
-- auto-repair	If a checked table is corrupted, automatically fix it. Repairing will be done after all tables have been checked.

--character-sets-dir=name	Directory where <b>character set</b> files are installed.
-c , --check	Check table for errors.
-C , --check-only-changed	Check only tables that have changed since last check or haven't been closed properly.
-g , --check-upgrade	Check tables for version-dependent changes. May be used with --auto-repair to correct tables requiring version-dependent updates. Automatically enables the --fix-db-names and --fix-table-names options. Used <a href="#">when upgrading</a>
--compress	Compress all information sent between the client and server if both support compression.
-B , --databases	Check several databases. Note that normally <i>mysqlcheck</i> treats the first argument as a database name, and following arguments as table names. With this option, no tables are given, and all name arguments are regarded as database names.
-# , --debug[=name]	Output debug log. Often this is 'd:t:o,filename'.
--debug-check	Check memory and open file usage at exit.
--debug-info	Print some debug info at exit.
--default-auth=plugin	Default authentication client-side plugin to use.
--default-character-set=name	Set the default <b>character set</b> .

	<code>-e</code> , <code>--</code>	If you are using this option with <code>--check</code> , it will ensure that the table is 100 percent consistent, but will take a long time. If you are using this option with <code>--repair</code> , it will force using the old, slow, repair with keycache method, instead of the much faster repair by sorting.
	<code>-F</code> , <code>--</code>	Check only tables that haven't been closed properly.
	<code>--fix-db-names</code>	Convert database names to the format used since MySQL 5.1. Only database names that contain special characters are affected. Used <a href="#">when upgrading</a> from an old MySQL version.
	<code>--fix-table-names</code>	Convert table names (including <a href="#">views</a> ) to the format used since MySQL 5.1. Only table names that contain special characters are affected. Used <a href="#">when upgrading</a> from an old MySQL version.
	<code>--flush</code>	Flush each table after check. This is useful if you don't want to have the checked tables take up space in the caches after the check.
	<code>-f</code> , <code>--</code>	Continue even if we get an SQL error.
	<code>-?</code> , <code>--</code>	Display this help message and exit.
	<code>-h</code> , <code>--host=name</code>	Connect to the given host.
	<code>-m</code> , <code>--</code>	Faster than extended-check, but only finds 99.99 percent of all errors. Should be good enough for most cases.
	<code>-o</code> , <code>--</code>	Optimize tables.
	<code>-p</code> , <code>--password[=name]</code>	Password to use when connecting to the server. If you use the short option form ( <code>-p</code> ), you cannot have a space between the option and the password. If you omit the password value following the <code>--password</code> or <code>-p</code> option on the command line, mysqlcheck prompts for one. Specifying a password on the command line should be considered insecure. You can use an option file to avoid giving the password on the command line.

	<b>-z</b> ,--	When using ANALYZE TABLE ( --analyze ), uses the PERSISTENT FOR ALL option, which forces <a href="#">Engine-independent Statistics</a> for this table to be updated. Added in <a href="#">MariaDB 10.1.10</a>
	<b>-W</b> ,--	On Windows, connect to the server via a named pipe. This option applies only if the server supports named-pipe connections.
	<b>--plugin-dir</b>	Directory for client-side plugins.
<b>num</b>  <b>port=num</b>	<b>-P</b> ,--	Port number to use for connection or 0 for default to, in order of preference, my.cnf, \$MYSQL_TCP_PORT, /etc/services, built-in default (3306).
	<b>--process-tables</b>	Perform the requested operation (check, repair, analyze, optimize) on tables. Enabled by default. Use --skip-process-tables to disable.
<b>process-views[=val]</b>	<b>--</b>	Perform the requested operation (only <a href="#">CHECK VIEW</a> or <a href="#">REPAIR VIEW</a> ). Possible values are NO, YES (correct the checksum, if necessary, add the mariadb-version field), UPGRADE_FROM_MYSQL (same as YES and toggle the algorithm MERGE->TEMPTABLE).
	<b>--protocol=name</b>	The connection protocol (tcp, socket, pipe, memory) to use for connecting to the server. Useful when other connection parameters would cause a protocol to be used other than the one you want.
<b>quick</b>	<b>-q</b> ,--	If you are using this option with CHECK TABLE, it prevents the check from scanning the rows to check for wrong links. This is the fastest check. If you are using this option with REPAIR TABLE, it will try to repair only the index tree. This is the fastest repair method for a table.
<b>repair</b>	<b>-r</b> ,--	Can fix almost anything except unique keys that aren't unique.
<b>shared-memory-base-name</b>	<b>--</b>	Shared-memory name to use for Windows connections using shared memory to a local server (started with the --shared-memory option). Case-sensitive.
<b>silent</b>	<b>-s</b> ,--	Print only error messages.
<b>--skip-database</b>	<b>--</b>	Don't process the database (case-sensitive) specified as argument.

name -S , --socket=name	For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.
ssl --ssl	Enables <a href="#">TLS</a> . TLS is also enabled even without setting this option when certain other TLS options are set. Starting with <a href="#">MariaDB 10.2</a> , the <code>--ssl</code> option will not enable <a href="#">verifying the server certificate</a> by default. In order to verify the server certificate, the user must specify the <code>--ssl-verify-server-cert</code> option.
ssl-ca=name --ssl-ca=name	Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. See <a href="#">Secure Connections Overview: Certificate Authorities (CAs)</a> for more information. This option implies the <code>--ssl</code> option.
ssl-capath=name --ssl-capath=name	Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the <code>openssl rehash</code> command. See <a href="#">Secure Connections Overview: Certificate Authorities (CAs)</a> for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See <a href="#">TLS and Cryptography Libraries Used by MariaDB</a> for more information about which libraries are used on which platforms. This option implies the <code>--ssl</code> option.
ssl-cert=name --ssl-cert=name	Defines a path to the X509 certificate file to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. This option implies the <code>--ssl</code> option.
ssl-cipher=name --ssl-cipher=name	List of permitted ciphers or cipher suites to use for <a href="#">TLS</a> . This option implies the <code>--ssl</code> option.
ssl-crl=name --ssl-crl=name	Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. See <a href="#">Secure Connections Overview: Certificate Revocation Lists (CRLs)</a> for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See <a href="#">TLS and Cryptography Libraries Used by MariaDB</a> for more information about which libraries are used on which platforms.
ssl-crlpath=name --ssl-crlpath=name	Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the <code>openssl rehash</code> command. See <a href="#">Secure Connections Overview: Certificate Revocation Lists (CRLs)</a> for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See <a href="#">TLS and Cryptography Libraries Used by MariaDB</a> for more information about which libraries are used on which platforms.
ssl-key=name --ssl-key=name	Defines a path to a private key file to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. This option implies the <code>--ssl</code> option.
ssl-verify-server-cert --ssl-verify-server-cert	Enables <a href="#">server certificate verification</a> . This option is disabled by default.

tables	--	Overrides the --databases or -B option such that all name arguments following the option are regarded as table names.
usefrm	--	For repair operations on MyISAM tables, get table structure from .frm file, so the table can be repaired even if the .MYI header is corrupted.
username	-u ,	User for login if not current user.
verbose	-v ,	Print info about the various stages. You can give this option several times to get even more information. See <a href="#">mysqlcheck and verbose</a> , below.
version	-V ,	Output version information and exit.
write-binlog	--	Write ANALYZE, OPTIMIZE and REPAIR TABLE commands to the <a href="#">binary log</a> . Enabled by default; use --skip-write-binlog when commands should not be sent to replication slaves.

## Option Files

In addition to reading options from the command-line,

```
mysqlcheck
can also read options from option files. If an unknown option is provided to
mysqlcheck
in an option file, then it is ignored.
```

The following options relate to how MariaDB command-line tools handles option files. They must be given as the first argument on the command-line:

Option	Description
--print-defaults	Print the program argument list and exit.
--no-defaults	Don't read default options from any option file.
--defaults-file=#	Only read default options from the given file #.
--defaults-extra-file=#	Read this file after the global files are read.
--defaults-group-suffix=#	In addition to the default option groups, also read option groups with this suffix.

In [MariaDB 10.2](#) and later,

```
mysqlcheck
is linked with MariaDB Connector/C. However, MariaDB Connector/C does not yet handle the parsing of option files for this client. That is still
performed by the server option file parsing code. See MDEV-19035 for more information.
```

## Option Groups

`mysqlcheck`  
reads options from the following [option groups](#) from [option files](#) :

Group	Description
[mysqlcheck]	Options read by <code>mysqlcheck</code> , which includes both MariaDB Server and MySQL Server.
[mariadb-check]	Options read by <code>mysqlcheck</code> . Available starting with <a href="#">MariaDB 10.4.6</a> .
[client]	Options read by all MariaDB and MySQL <a href="#">client programs</a> , which includes both MariaDB and MySQL clients. For example, <code>mysqldump</code>
[client-server]	Options read by all MariaDB <a href="#">client programs</a> and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients.
[client-mariadb]	Options read by all MariaDB <a href="#">client programs</a> .

## Notes

### Default Values

To see the default values for the options and also to see the arguments you get from configuration files you can do:

```
./client/mysqlcheck --print-defaults
./client/mysqlcheck --help
```

### mysqlcheck and auto-repair

When running

```
mysqlcheck
with
--auto-repair
(as done by mysql\_upgrade),
mysqlcheck
will first check all tables and then in a separate phase repair those that failed the check.
```

### mysqlcheck and all-databases

```
mysqlcheck --all-databases
will ignore the internal log tables general\_log and slow\_log as these can't be checked, repaired or optimized.
```

### mysqlcheck and verbose

Using one

```
--verbose
option will give you more information about what mysqlcheck is doing.
```

Using two

```
--verbose
options will also give you connection information.
```

If you use three

```
--verbose
options you will also get, on stdout, all ALTER , RENAME , and CHECK commands that mysqlcheck executes.
```

## 1.1.2.1.12 OPTIMIZE TABLE

# Syntax

```
OPTIMIZE [NO_WRITE_TO_BINLOG | LOCAL] TABLE
  tbl_name [, tbl_name] ...
  [WAIT n | NOWAIT]
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [WAIT/NOWAIT](#)
  2. [Defragmenting](#)
  3. [Updating an InnoDB fulltext index](#)
  4. [Defragmenting InnoDB tablespaces](#)
3. [See Also](#)

## Description

OPTIMIZE TABLE

has two main functions. It can either be used to defragment tables, or to update the InnoDB fulltext index.

MariaDB starting with [10.3.0](#)

### WAIT/NOWAIT

Set the lock wait timeout. See [WAIT and NOWAIT](#).

## Defragmenting

OPTIMIZE TABLE

works for [InnoDB](#) (before [MariaDB 10.1.1](#), only if the `innodb_file_per_table` server system variable is set), [Aria](#), [MyISAM](#) and [ARCHIVE](#) tables, and should be used if you have deleted a large part of a table or if you have made many changes to a table with variable-length rows (tables that have `VARCHAR`, `VARBINARY`, `BLOB`, or `TEXT` columns). Deleted rows are maintained in a linked list and subsequent

`INSERT`

operations reuse old row positions.

This statement requires [SELECT](#) and [INSERT](#) privileges for the table.

By default,

OPTIMIZE TABLE

statements are written to the [binary log](#) and will be [replicated](#). The

`NO_WRITE_TO_BINLOG`

keyword (

`LOCAL`

is an alias) will ensure the statement is not written to the binary log.

From [MariaDB 10.3.19](#),

OPTIMIZE TABLE

statements are not logged to the binary log if `read_only` is set. See also [Read-Only Replicas](#).

OPTIMIZE TABLE

is also supported for partitioned tables. You can use

[ALTER TABLE](#)

... `OPTIMIZE PARTITION`

to optimize one or more partitions.

You can use

OPTIMIZE TABLE

to reclaim the unused space and to defragment the data file. With other storage engines,

OPTIMIZE TABLE

does nothing by default, and returns this message: "The storage engine for the table doesn't support optimize". However, if the server has been started with the

`--skip-new`

option,

OPTIMIZE TABLE

is linked to [ALTER TABLE](#), and recreates the table. This operation frees the unused space and updates index statistics.

The [Aria](#) storage engine supports [progress reporting](#) for this statement.

If a [MyISAM](#) table is fragmented, [concurrent inserts](#) will not be performed until an

```
OPTIMIZE TABLE  
statement is executed on that table, unless the concurrent\_insert server system variable is set to  
ALWAYS
```

## Updating an InnoDB fulltext index

When rows are added or deleted to an InnoDB [fulltext index](#), the index is not immediately re-organized, as this can be an expensive operation. Change statistics are stored in a separate location. The fulltext index is only fully re-organized when an

```
OPTIMIZE TABLE  
statement is run.
```

By default, an OPTIMIZE TABLE will defragment a table. In order to use it to update fulltext index statistics, the [innodb\\_optimize\\_fulltext\\_only](#) system variable must be set to

```
1  
. This is intended to be a temporary setting, and should be reset to  
0  
once the fulltext index has been re-organized.
```

Since fulltext re-organization can take a long time, the [innodb\\_ft\\_num\\_word\\_optimize](#) variable limits the re-organization to a number of words (2000 by default). You can run multiple OPTIMIZE statements to fully re-organize the index.

## Defragmenting InnoDB tablespaces

[MariaDB 10.1.1](#) merged the Facebook/Kakao defragmentation patch, allowing one to use

```
OPTIMIZE TABLE  
to defragment InnoDB tablespaces. For this functionality to be enabled, the innodb\_defragment system variable must be enabled. No new tables  
are created and there is no need to copy data from old tables to new tables. Instead, this feature loads  
n  
pages (determined by innodb-defragment-n-pages) and tries to move records so that pages would be full of records and then frees pages that  
are fully empty after the operation. Note that tablespace files (including ibdata1) will not shrink as the result of defragmentation, but one will get better  
memory utilization in the InnoDB buffer pool as there are fewer data pages in use.
```

See [Defragmenting InnoDB Tablespaces](#) for more details.

## See Also

- [Optimize Table in InnoDB with ALGORITHM set to INPLACE](#)
- [Optimize Table in InnoDB with ALGORITHM set to NOCOPY](#)
- [Optimize Table in InnoDB with ALGORITHM set to INSTANT](#)

## 1.1.2.1.13 RENAME TABLE

### Syntax

```
RENAME TABLE[S] [IF EXISTS] tbl_name  
[WAIT n | NOWAIT]  
TO new_tbl_name  
[, tbl_name2 TO new_tbl_name2] ...
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [IF EXISTS](#)
  2. [WAIT/NOWAIT](#)
  3. [Privileges](#)
  4. [Atomic RENAME TABLE](#)

### Description

This statement renames one or more tables or [views](#), but not the privileges associated with them.

#### IF EXISTS

MariaDB starting with [10.5.2](#)

If this directive is used, one will not get an error if the table to be renamed doesn't exist.

The rename operation is done atomically, which means that no other session can access any of the tables while the rename is running. For example, if you have an existing table

```
old_table  
, you can create another table  
new_table  
that has the same structure but is empty, and then replace the existing table with the empty one as follows (assuming that  
backup_table  
does not already exist):
```

```
CREATE TABLE new_table (...);  
RENAME TABLE old_table TO backup_table, new_table TO old_table;
```

tbl\_name  
can optionally be specified as  
db\_name  
. .  
tbl\_name  
. See [Identifier Qualifiers](#). This allows to use  
RENAME  
to move a table from a database to another (as long as they are on the same filesystem):

```
RENAME TABLE db1.t TO db2.t;
```

Note that moving a table to another database is not possible if it has some [triggers](#). Trying to do so produces the following error:

```
ERROR 1435 (HY000): Trigger in wrong schema
```

Also, views cannot be moved to another database:

```
ERROR 1450 (HY000): Changing schema from 'old_db' to 'new_db' is not allowed.
```

Multiple tables can be renamed in a single statement. The presence or absence of the optional

```
S  
(  
RENAME TABLE  
or  
RENAME TABLES  
) has no impact, whether a single or multiple tables are being renamed.
```

If a

```
RENAME TABLE  
renames more than one table and one renaming fails, all renames executed by the same statement are rolled back.
```

Renames are always executed in the specified order. Knowing this, it is also possible to swap two tables' names:

```
RENAME TABLE t1 TO tmp_table,  
t2 TO t1,  
tmp_table TO t2;
```

## WAIT/NOWAIT

MariaDB starting with [10.3.0](#)

Set the lock wait timeout. See [WAIT](#) and [NOWAIT](#).

## Privileges

Executing the

```
RENAME TABLE  
statement requires the DROP, CREATE and INSERT privileges for the table or the database.
```

## Atomic RENAME TABLE

MariaDB starting with [10.6.1](#)

From MariaDB 10.6 ,  
 RENAME TABLE  
 is atomic for most engines, including InnoDB, MyRocks, MyISAM and Aria ( [MDEV-23842](#) ). This means that if there is a crash (server down or power outage) during  
 RENAME TABLE  
 , all tables will revert to their original names and any changes to trigger files will be reverted.  
 In older MariaDB version there was a small chance that, during a server crash happening in the middle of  
 RENAME TABLE  
 , some tables could have been renamed (in the worst case partly) while others would not be renamed.  
 See [Atomic DDL](#) for more information.

## 1.1.2.1.14 REPAIR TABLE

### Syntax

```
REPAIR [NO_WRITE_TO_BINLOG | LOCAL] TABLE
tbl_name [, tbl_name] ...
[QUICK] [EXTENDED] [USE_FRM]
```

### Description

`REPAIR TABLE`  
 repairs a possibly corrupted table. By default, it has the same effect as

```
myisamchk --recover tbl_name
```

or

```
aria_chk --recover tbl_name
```

See [aria\\_chk](#) and [myisamchk](#) for more.

`REPAIR TABLE`  
 works for [Archive](#) , [Aria](#) , [CSV](#) and [MyISAM](#) tables. For [InnoDB](#) , see [recovery modes](#) . For CSV, see also [Checking and Repairing CSV Tables](#) . For Archive, this statement also improves compression. If the storage engine does not support this statement, a warning is issued.

This statement requires [SELECT](#) and [INSERT](#) privileges for the table.

By default,

```
REPAIR TABLE
statements are written to the binary log and will be replicated . The
NO_WRITE_TO_BINLOG
keyword (
LOCAL
is an alias) will ensure the statement is not written to the binary log.
```

From MariaDB 10.3.19 ,

```
REPAIR TABLE
statements are not logged to the binary log if read\_only is set. See also Read-Only Replicas .
```

When an index is recreated, the storage engine may use a configurable buffer in the process. Incrementing the buffer speeds up the index creation. [Aria](#) and [MyISAM](#) allocate a buffer whose size is defined by

```
aria_sort_buffer_size
```

or

```
myisam_sort_buffer_size
```

, also used for

```
ALTER TABLE
```

```
.
```

```
REPAIR TABLE
```

is also supported for partitioned tables. However, the `USE_FRM` option cannot be used with this statement on a partitioned table.

```
ALTER TABLE ... REPAIR PARTITION
```

can be used to repair one or more partitions.

The [Aria](#) storage engine supports [progress reporting](#) for this statement.

## 1.1.2.1.15 REPAIR VIEW

### Syntax

```
REPAIR [NO_WRITE_TO_BINLOG | LOCAL] VIEW view_name[, view_name] ... [FROM MYSQL]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

### Description

The

```
REPAIR VIEW
```

statement was introduced to assist with fixing [MDEV-6916](#), an issue introduced in [MariaDB 5.2](#) where the view algorithms were swapped compared to their MySQL on disk representation. It checks whether the view algorithm is correct. It is run as part of [mysql\\_upgrade](#), and should not normally be required in regular use.

By default it corrects the checksum and if necessary adds the mariadb-version field. If the optional

```
FROM MYSQL
```

clause is used, and no mariadb-version field is present, the MERGE and TEMPTABLE algorithms are toggled.

By default,

```
REPAIR VIEW  
statements are written to the binary log and will be replicated. The  
NO_WRITE_TO_BINLOG  
keyword (  
LOCAL  
is an alias) will ensure the statement is not written to the binary log.
```

### See Also

- [CHECK VIEW](#)

## 1.1.2.1.16 REPLACE

### Syntax

```
REPLACE [LOW_PRIORITY | DELAYED]  
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]  
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...  
[RETURNING select_expr  
[, select_expr ...]]
```

Or:

```
REPLACE [LOW_PRIORITY | DELAYED]  
[INTO] tbl_name [PARTITION (partition_list)]  
SET col={expr | DEFAULT}, ...  
[RETURNING select_expr  
[, select_expr ...]]
```

Or:

```
REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
SELECT ...
[RETURNING select_expr
[, select_expr ...]]
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [PARTITION](#)
  2. [REPLACE RETURNING](#)
    1. [Examples](#)
3. [Examples](#)
4. [See Also](#)

## Description

REPLACE  
works exactly like

[INSERT](#)

, except that if an old row in the table has the same value as a new row for a  
PRIMARY KEY  
or a  
UNIQUE  
index, the old row is deleted before the new row is inserted. If the table has more than one  
UNIQUE  
keys, it is possible that the new row conflicts with more than one row. In this case, all conflicting rows will be deleted.

The table name can be specified in the form

db\_name  
.tbl\_name  
or, if a default database is selected, in the form  
tbl\_name  
(see [Identifier Qualifiers](#)). This allows to use

[REPLACE ... SELECT](#)

to copy rows between different databases.

MariaDB starting with [10.5.0](#)

The RETURNING clause was introduced in [MariaDB 10.5.0](#)

Basically it works like this:

```
BEGIN;
SELECT 1 FROM t1 WHERE key=# FOR UPDATE;
IF found-row
  DELETE FROM t1 WHERE key=# ;
ENDIF
INSERT INTO t1 VALUES (...);
END;
```

The above can be replaced with:

```
REPLACE INTO t1 VALUES (...)
```

REPLACE  
is a MariaDB/MySQL extension to the SQL standard. It either inserts, or deletes and inserts. For other MariaDB/MySQL extensions to standard SQL --- that also handle duplicate values --- see [IGNORE](#) and [INSERT ON DUPLICATE KEY UPDATE](#).

Note that unless the table has a

PRIMARY KEY  
or

UNIQUE  
index, using a  
REPLACE  
statement makes no sense. It becomes equivalent to  
INSERT  
, because there is no index to be used to determine whether a new row duplicates another.

Values for all columns are taken from the values specified in the

REPLACE  
statement. Any missing columns are set to their default values, just as happens for  
INSERT  
. You cannot refer to values from the current row and use them in the new row. If you use an assignment such as  
'SET col = col + 1'  
, the reference to the column name on the right hand side is treated as  
DEFAULT(col)  
, so the assignment is equivalent to  
'SET col = DEFAULT(col) + 1'

To use

REPLACE  
, you must have both the  
INSERT  
and  
DELETE  
privileges for the table.

There are some gotchas you should be aware of, before using

REPLACE  
:

- If there is an

[AUTO\\_INCREMENT](#)

field, a new value will be generated.

- If there are foreign keys,

ON DELETE  
action will be activated by  
REPLACE

- Triggers on

DELETE  
and  
INSERT  
will be activated by  
REPLACE

To avoid some of these behaviors, you can use

INSERT ... ON DUPLICATE KEY UPDATE

This statement activates INSERT and DELETE triggers. See [Trigger Overview](#) for details.

## PARTITION

See [Partition Pruning and Selection](#) for details.

## REPLACE RETURNING

REPLACE ... RETURNING  
returns a resultset of the replaced rows.

This returns the listed columns for all the rows that are replaced, or alternatively, the specified SELECT expression. Any SQL expressions which can be calculated can be used in the select expression for the RETURNING clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

### Examples

Simple REPLACE statement

```

REPLACE INTO t2 VALUES (1,'Leopard'),(2,'Dog') RETURNING id2, id2+id2
as Total ,id2|id2, id2&&id2;
+-----+-----+-----+
| id2 | Total | id2|id2 | id2&&id2 |
+-----+-----+-----+
|  1 |     2 |     1 |     1 |
|  2 |     4 |     2 |     1 |
+-----+-----+-----+

```

Using stored functions in RETURNING

```

DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
BEGIN
    RETURN (SELECT arg+arg);
END |

DELIMITER ;
PREPARE stmt FROM "REPLACE INTO t2 SET id2=3, animal2='Fox' RETURNING f2(id2),
UPPER(animal2)";

EXECUTE stmt;
+-----+
| f2(id2) | UPPER(animal2) |
+-----+
|      6 | FOX             |
+-----+

```

Subqueries in the statement

```

REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT id2 FROM t2 WHERE
id2 IN (SELECT id2 FROM t2 WHERE id2=1)) AS new_id;
+-----+
| new_id |
+-----+
|   1 |
|   1 |
|   1 |
|   1 |
+-----+

```

Subqueries in the RETURNING clause that return more than one row or column cannot be used..

Aggregate functions cannot be used in the RETURNING clause. Since aggregate functions work on a set of values and if the purpose is to get the row count, ROW\_COUNT() with SELECT can be used, or it can be used in REPLACE...SEL== Description

```

REPLACE ... RETURNING
returns a resultset of the replaced rows.

```

This returns the listed columns for all the rows that are replaced, or alternatively, the specified SELECT expression. Any SQL expressions which can be calculated can be used in the select expression for the RETURNING clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

## Examples

Simple REPLACE statement

```

REPLACE INTO t2 VALUES (1,'Leopard'),(2,'Dog') RETURNING id2, id2+id2
as Total ,id2|id2, id2&&id2;
+-----+-----+-----+
| id2 | Total | id2|id2 | id2&&id2 |
+-----+-----+-----+
|  1 |     2 |     1 |     1 |
|  2 |     4 |     2 |     1 |
+-----+-----+-----+

```

Using stored functions in RETURNING

```

DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
BEGIN
    RETURN (SELECT arg+arg);
END |

DELIMITER ;
PREPARE stmt FROM "REPLACE INTO t2 SET id2=3, animal2='Fox' RETURNING f2(id2),
UPPER(animal2)";

EXECUTE stmt;
+-----+
| f2(id2) | UPPER(animal2) |
+-----+
|      6   | FOX           |
+-----+

```

Subqueries in the statement

```

REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT id2 FROM t2 WHERE
id2 IN (SELECT id2 FROM t2 WHERE id2=1)) AS new_id;
+-----+
| new_id |
+-----+
|      1  |
|      1  |
|      1  |
|      1  |
+-----+

```

Subqueries in the RETURNING clause that return more than one row or column cannot be used..

Aggregate functions cannot be used in the RETURNING clause. Since aggregate functions work on a set of values and if the purpose is to get the row count, ROW\_COUNT() with SELECT can be used, or it can be used in REPLACE...SELECT...RETURNING if the table in the RETURNING clause is not the same as the REPLACE table. ECT...RETURNING if the table in the RETURNING clause is not the same as the REPLACE table.

## See Also

- [INSERT](#)
- [HIGH\\_PRIORITY and LOW\\_PRIORITY clauses](#)
- [INSERT DELAYED](#) for details on the  
DELAYED  
clause

### 1.1.2.1.17 SHOW COLUMNS

#### Syntax

```

SHOW [FULL] {COLUMNS | FIELDS} FROM tbl_name [FROM db_name]
[LIKE 'pattern' | WHERE expr]

```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

#### Description

SHOW COLUMNS  
 displays information about the columns in a given table. It also works for views. The  
 LIKE  
 clause, if present on its own, indicates which column names to match. The  
 WHERE  
 and  
 LIKE  
 clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

If the data types differ from what you expect them to be based on a

CREATE TABLE

statement, note that MariaDB sometimes changes data types when you create or alter a table. The conditions under which this occurs are described in the [Silent Column Changes](#) article.

The

FULL

keyword causes the output to include the column collation and comments, as well as the privileges you have for each column.

You can use

db\_name.tbl\_name

as an alternative to the

tbl\_name FROM db\_name

syntax. In other words, these two statements are equivalent:

```
SHOW COLUMNS FROM mytable FROM mydb;
SHOW COLUMNS FROM mydb.mytable;
```

SHOW COLUMNS

displays the following values for each table column:

**Field** indicates the column name.

**Type** indicates the column data type.

**Collation** indicates the collation for non-binary string columns, or NULL for other columns. This value is displayed only if you use the FULL keyword.

The **Null** field contains YES if NULL values can be stored in the column, NO if not.

The **Key** field indicates whether the column is indexed:

- If **Key** is empty, the column either is not indexed or is indexed only as a secondary column in a multiple-column, non-unique index.
- If **Key** is **PRI**, the column is a

PRIMARY KEY

or is one of the columns in a multiple-column

PRIMARY KEY

- If **Key** is **UNI**, the column is the first column of a unique-valued index that cannot contain

NULL

values.

- If **Key** is **MUL**, multiple occurrences of a given value are allowed within the column. The column is the first column of a non-unique index or a unique-valued index that can contain

NULL

values.

If more than one of the **Key** values applies to a given column of a table, **Key** displays the one with the highest priority, in the order PRI, UNI, MUL.

A

UNIQUE

index may be displayed as

PRI

if it cannot contain

NULL

values and there is no

PRIMARY KEY

in the table. A

UNIQUE

index may display as

MUL

if several columns form a composite

UNIQUE

index; although the combination of the columns is unique, each column can still hold multiple occurrences of a given value.

The **Default** field indicates the default value that is assigned to the column.

The **Extra** field contains any additional information that is available about a given column.

Value	Description
AUTO_INCREMENT	The column was created with the AUTO_INCREMENT keyword.
PERSISTENT	The column was created with the PERSISTENT keyword. (New in 5.3)

VIRTUAL	The column was created with the VIRTUAL keyword. (New in 5.3)
on update CURRENT_TIMESTAMP	The column is a TIMESTAMP column that is automatically updated on INSERT and UPDATE.

**Privileges** indicates the privileges you have for the column. This value is displayed only if you use the FULL keyword.

**Comment** indicates any comment the column has. This value is displayed only if you use the FULL keyword.

SHOW FIELDS  
is a synonym for  
SHOW COLUMNS  
. Also

[DESCRIBE](#)

and

[EXPLAIN](#)

can be used as shortcuts.

You can also list a table's columns with:

```
mysqlshow db_name tbl_name
```

See the

[mysqlshow](#)

command for more details.

The

[DESCRIBE](#)

statement provides information similar to  
SHOW COLUMNS  
. The

[information\\_schema.COLUMNS](#)

table provides similar, but more complete, information.

The

[SHOW CREATE TABLE](#)

,

[SHOW TABLE STATUS](#)

, and

[SHOW INDEX](#)

statements also provide information about tables.

## Examples

```
SHOW COLUMNS FROM city;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| Id | int(11) | NO | PRI | NULL | auto_increment |
| Name | char(35) | NO | | | |
| Country | char(3) | NO | UNI | | |
| District | char(20) | YES | MUL | | |
| Population | int(11) | NO | | 0 | |
+-----+-----+-----+-----+
```

```
SHOW COLUMNS FROM employees WHERE Type LIKE 'Varchar%';
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| first_name | varchar(30) | NO | MUL | NULL | |
| last_name | varchar(40) | NO | | NULL | |
| position | varchar(25) | NO | | NULL | |
| home_address | varchar(50) | NO | | NULL | |
| home_phone | varchar(12) | NO | | NULL | |
| employee_code | varchar(25) | NO | UNI | NULL | |
+-----+-----+-----+-----+
```

## See Also

- [DESCRIBE](#)
- [mysqlshow](#)
- [SHOW CREATE TABLE](#)
- [SHOW TABLE STATUS](#)
- [SHOW INDEX](#)
- [Extended SHOW](#)
- [Silent Column Changes](#)

### 1.1.2.1.18 SHOW CREATE TABLE

#### Syntax

```
SHOW CREATE TABLE tbl_name
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

#### Description

Shows the [CREATE TABLE](#) statement that created the given table. The statement requires the [SELECT](#) privilege for the table. This statement also works with [views](#) and [SEQUENCE](#).

```
SHOW CREATE TABLE
quotes table and column names according to the value of the sql_quote_show_create server system variable.
```

Certain [SQL\\_MODE](#) values can result in parts of the original [CREATE](#) statement not being included in the output. MariaDB-specific table options, column options, and index options are not included in the output of this statement if the [NO\\_TABLE\\_OPTIONS](#), [NO\\_FIELD\\_OPTIONS](#) and [NO\\_KEY\\_OPTIONS](#) [SQL\\_MODE](#) flags are used. All MariaDB-specific table attributes are also not shown when a non-MariaDB/MySQL emulation mode is used, which includes [ANSI](#), [DB2](#), [POSTGRESQL](#), [MSSQL](#), [MAXDB](#) or [ORACLE](#).

Invalid table options, column options and index options are normally commented out (note, that it is possible to create a table with invalid options, by altering a table of a different engine, where these options were valid). To have them uncommented, enable the [IGNORE\\_BAD\\_TABLE\\_OPTIONS](#) [SQL\\_MODE](#). Remember that replaying a [CREATE TABLE](#) statement with uncommented invalid options will fail with an error, unless the [IGNORE\\_BAD\\_TABLE\\_OPTIONS](#) [SQL\\_MODE](#) is in effect.

Note that

```
SHOW CREATE TABLE
is not meant to provide metadata about a table. It provides information about how the table was declared, but the real table structure could differ
```

a bit. For example, if an index has been declared as

```
HASH
, the
CREATE TABLE
statement returned by
SHOW CREATE TABLE
will declare that index as
HASH
; however, it is possible that the index is in fact a
BTREE
, because the storage engine does not support
HASH
```

#### MariaDB starting with 10.2.1

MariaDB 10.2.1 permits TEXT and BLOB data types to be assigned a DEFAULT value. As a result, from MariaDB 10.2.1 ,

```
SHOW CREATE TABLE
will append a
DEFAULT NULL
to nullable TEXT or BLOB fields if no specific default is provided.
```

#### MariaDB starting with 10.2.2

From MariaDB 10.2.2 , numbers are no longer quoted in the

```
DEFAULT
clause in
SHOW CREATE
statement. Previously, MariaDB quoted numbers.
```

## Examples

```
SHOW CREATE TABLE t\G
***** 1. row *****
Table: t
Create Table: CREATE TABLE `t` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `s` char(60) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

With `sql_quote_show_create` off:

```
SHOW CREATE TABLE t\G
***** 1. row *****
Table: t
Create Table: CREATE TABLE t (
  id int(11) NOT NULL AUTO_INCREMENT,
  s char(60) DEFAULT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Unquoted numeric DEFAULTs, from MariaDB 10.2.2 :

```
CREATE TABLE td (link TINYINT DEFAULT 1);

SHOW CREATE TABLE td\G
***** 1. row *****
Table: td
Create Table: CREATE TABLE `td` (
  `link` tinyint(4) DEFAULT 1
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Quoted numeric DEFAULTs, until MariaDB 10.2.1 :

```

CREATE TABLE td (link TINYINT DEFAULT 1);

SHOW CREATE TABLE td\G
***** 1. row *****
    Table: td
Create Table: CREATE TABLE `td` (
  `link` tinyint(4) DEFAULT '1'
) ENGINE=InnoDB DEFAULT CHARSET=latin1

```

[SQL\\_MODE](#) impacting the output:

```

SELECT @@sql_mode;
+-----+
| @@sql_mode |
+-----+
| STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----+

CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `msg` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
;

SHOW CREATE TABLE t1\G
***** 1. row *****
    Table: t1
Create Table: CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `msg` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1

SET SQL_MODE=ORACLE;

SHOW CREATE TABLE t1\G
***** 1. row *****
    Table: t1
Create Table: CREATE TABLE "t1" (
  "id" int(11) NOT NULL,
  "msg" varchar(100) DEFAULT NULL,
  PRIMARY KEY ("id")
)

```

## See Also

- [SHOW CREATE SEQUENCE](#)
- [SHOW CREATE VIEW](#)

## 1.1.2.1.19 SHOW INDEX

### Syntax

```

SHOW {INDEX | INDEXES | KEYS}
  FROM tbl_name [FROM db_name]
  [WHERE expr]

```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

```

SHOW INDEX
returns table index information. The format resembles that of the SQLStatistics call in ODBC.

```

You can use

db\_name.tbl\_name  
as an alternative to the  
tbl\_name FROM db\_name  
syntax. These two statements are equivalent:

```
SHOW INDEX FROM mytable FROM mydb;
SHOW INDEX FROM mydb.mytable;
```

SHOW KEYS  
and  
SHOW INDEXES  
are synonyms for  
SHOW INDEX

You can also list a table's indexes with the [mariadb-show/mysqlshow](#) command:

```
mysqlshow -k db_name tbl_name
```

The `information_schema.STATISTICS` table stores similar information.

The following fields are returned by

```
SHOW INDEX
```

Field	Description
Table	Table name
Non_unique	1 if the index permits duplicate values, 0 if values must be unique.
Key_name	Index name. The primary key is always named PRIMARY
Seq_in_index	The column's sequence in the index, beginning with 1
Column_name	Column name.
Collation	Either A , if the column is sorted in ascending order in the index, or NULL if it's not sorted.
Cardinality	Estimated number of unique values in the index. The cardinality statistics are calculated at various times, and can help the optimizer make improved decisions.
Sub_part	NULL if the entire column is included in the index, or the number of included characters if not.
Packed	NULL if the index is not packed, otherwise how the index is packed.

Null	<p>NULL if NULL values are permitted in the column, an empty string if NULL 's are not permitted.</p>
Index_type	<p>The index type, which can be</p> <p>BTREE , FULLTEXT , HASH or RTREE . See <a href="#">Storage Engine Index Types</a> .</p>
Comment	Other information, such as whether the index is disabled.
Index_comment	Contents of the COMMENT attribute when the index was created.
Ignored	Whether or not an index will be ignored by the optimizer. See <a href="#">Ignored Indexes</a> . From <a href="#">MariaDB 10.6.0</a> .

The

WHERE  
and  
LIKE

clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#) .

## Examples

```

CREATE TABLE IF NOT EXISTS `employees_example` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`first_name` varchar(30) NOT NULL,
`last_name` varchar(40) NOT NULL,
`position` varchar(25) NOT NULL,
`home_address` varchar(50) NOT NULL,
`home_phone` varchar(12) NOT NULL,
`employee_code` varchar(25) NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `employee_code` (`employee_code`),
KEY `first_name` (`first_name`,`last_name`)
) ENGINE=Aria;

INSERT INTO `employees_example`(`first_name`, `last_name`, `position`, `home_address`, `home_phone`, `employee_code`)
VALUES
('Mustapha', 'Mond', 'Chief Executive Officer', '692 Promiscuous Plaza', '326-555-3492', 'MM1'),
('Henry', 'Foster', 'Store Manager', '314 Savage Circle', '326-555-3847', 'HF1'),
('Bernard', 'Marx', 'Cashier', '1240 Ambient Avenue', '326-555-8456', 'BM1'),
('Lenina', 'Crowne', 'Cashier', '281 Bumblepuppy Boulevard', '328-555-2349', 'LC1'),
('Fanny', 'Crowne', 'Restocker', '1023 Bokanovsky Lane', '326-555-6329', 'FC1'),
('Helmholtz', 'Watson', 'Janitor', '944 Soma Court', '329-555-2478', 'HW1');

```

```

SHOW INDEXES FROM employees_example\G
***** 1. row *****
    Table: employees_example
  Non_unique: 0
    Key_name: PRIMARY
Seq_in_index: 1
Column_name: id
  Collation: A
Cardinality: 6
  Sub_part: NULL
    Packed: NULL
      Null:
Index_type: BTREE
  Comment:
Index_comment:
  Ignored: NO
***** 2. row *****
    Table: employees_example
  Non_unique: 0
    Key_name: employee_code
Seq_in_index: 1
Column_name: employee_code
  Collation: A
Cardinality: 6
  Sub_part: NULL
    Packed: NULL
      Null:
Index_type: BTREE
  Comment:
Index_comment:
  Ignored: NO
***** 3. row *****
    Table: employees_example
  Non_unique: 1
    Key_name: first_name
Seq_in_index: 1
Column_name: first_name
  Collation: A
Cardinality: NULL
  Sub_part: NULL
    Packed: NULL
      Null:
Index_type: BTREE
  Comment:
Index_comment:
  Ignored: NO
***** 4. row *****
    Table: employees_example
  Non_unique: 1
    Key_name: first_name
Seq_in_index: 2
Column_name: last_name
  Collation: A
Cardinality: NULL
  Sub_part: NULL
    Packed: NULL
      Null:
Index_type: BTREE
  Comment:
Index_comment:
  Ignored: NO

```

## See Also

- [Ignored Indexes](#)

## 1.1.2.1.20 TRUNCATE TABLE

### Syntax

```

TRUNCATE [TABLE] tbl_name
[WAIT n | NOWAIT]

```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [WAIT/NOWAIT](#)
  2. [Oracle-mode](#)
  3. [Performance](#)
3. [See Also](#)

## Description

TRUNCATE TABLE  
empties a table completely. It requires the  
DROP  
privilege. See [GRANT](#).

tbl\_name  
can also be specified in the form  
db\_name.  
  
tbl\_name  
(see [Identifier Qualifiers](#)).

Logically,

TRUNCATE TABLE  
is equivalent to a [DELETE](#) statement that deletes all rows, but there are practical differences under some circumstances.

TRUNCATE TABLE  
will fail for an [InnoDB table](#) if any FOREIGN KEY constraints from other tables reference the table, returning the error:

ERROR 1701 (42000): Cannot truncate a table referenced in a foreign key constraint

Foreign Key constraints between columns in the same table are permitted.

For an InnoDB table, if there are no

FOREIGN KEY  
constraints, InnoDB performs fast truncation by dropping the original table and creating an empty one with the same definition, which is much faster than deleting rows one by one. The [AUTO\\_INCREMENT](#) counter is reset by

TRUNCATE TABLE  
, regardless of whether there is a  
FOREIGN KEY  
constraint.

The count of rows affected by

TRUNCATE TABLE  
is accurate only when it is mapped to a  
DELETE  
statement.

For other storage engines,

TRUNCATE TABLE  
differs from  
DELETE  
in the following ways:

- Truncate operations drop and re-create the table, which is much faster than deleting rows one by one, particularly for large tables.
- Truncate operations cause an implicit commit.
- Truncation operations cannot be performed if the session holds an active table lock.
- Truncation operations do not return a meaningful value for the number of deleted rows. The usual result is "0 rows affected," which should be interpreted as "no information."
- As long as the table format file  
  
tbl\_name.frm  
is valid, the table can be re-created as an empty table with  
  
TRUNCATE TABLE  
, even if the data or index files have become corrupted.
- The table handler does not remember the last used [AUTO\\_INCREMENT](#) value, but starts counting from the beginning. This is true even for MyISAM and InnoDB, which normally do not reuse sequence values.
- When used with partitioned tables,  
  
TRUNCATE TABLE  
preserves the partitioning; that is, the data and index files are dropped and re-created, while the partition definitions (.par) file is unaffected.

- Since truncation of a table does not make any use of
 

```
DELETE
, the
TRUNCATE
```

 statement does not invoke
 

```
ON DELETE
```

 triggers.
- ```
TRUNCATE TABLE
```

 will only reset the values in the [Performance Schema summary tables](#) to zero or null, and will not remove the rows.

For the purposes of binary logging and [replication](#) ,

```
TRUNCATE TABLE
```

is treated as [DROP TABLE](#) followed by [CREATE TABLE](#) (DDL rather than DML).

```
TRUNCATE TABLE
does not work on views . Currently,
TRUNCATE TABLE
drops all historical records from a system-versioned table .
```

MariaDB starting with [10.3.0](#)

#### [WAIT/NOWAIT](#)

Set the lock wait timeout. See [WAIT](#) and [NOWAIT](#) .

#### Oracle-mode

[Oracle-mode](#) from [MariaDB 10.3](#) permits the optional keywords REUSE STORAGE or DROP STORAGE to be used.

```
TRUNCATE [TABLE] tbl_name [{DROP | REUSE} STORAGE] [WAIT n | NOWAIT]
```

These have no effect on the operation.

## Performance

```
TRUNCATE TABLE
is faster than DELETE , because it drops and re-creates a table.
```

With [InnoDB](#) ,

```
TRUNCATE TABLE
is slower if innodb\_file\_per\_table=ON is set (the default). This is because
TRUNCATE TABLE
unlinks the underlying tablespace file, which can be an expensive operation. See MDEV-8069 for more details.
```

The performance issues with [innodb\\_file\\_per\\_table=ON](#) can be exacerbated in cases where the [InnoDB buffer pool](#) is very large and [innodb\\_adaptive\\_hash\\_index=ON](#) is set. In that case, using [DROP TABLE](#) followed by [CREATE TABLE](#) instead of

```
TRUNCATE TABLE
may perform better. Setting innodb\_adaptive\_hash\_index=OFF (it defaults to ON before MariaDB 10.5 ) can also help. In MariaDB 10.2 only, from MariaDB 10.2.19 , this performance can also be improved by setting innodb\_safe\_truncate=OFF . See MDEV-9459 for more details.
```

Setting [innodb\\_adaptive\\_hash\\_index=OFF](#) can also improve

```
TRUNCATE TABLE
performance in general. See MDEV-16796 for more details.
```

## See Also

- [TRUNCATE function](#)
- [innodb\\_safe\\_truncate](#) system variable
- [Oracle mode from MariaDB 10.3](#)

## 1.1.2.1.21 UPDATE

### Syntax

Single-table syntax:

```

UPDATE [LOW_PRIORITY] [IGNORE] table_reference
[PARTITION (partition_list)]
[FOR PORTION OF period FROM expr1 TO expr2]
SET col1={expr1|DEFAULT} [,col2={expr2|DEFAULT}] ...
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]

```

Multiple-table syntax:

```

UPDATE [LOW_PRIORITY] [IGNORE] table_references
      SET col1={expr1|DEFAULT} [, col2={expr2|DEFAULT}] ...
      [WHERE where_condition]

```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [PARTITION](#)
  2. [FOR PORTION OF](#)
  3. [UPDATE Statements With the Same Source and Target](#)
3. [Example](#)
4. [See Also](#)

## Description

For the single-table syntax, the

```

UPDATE
statement updates columns of existing rows in the named table with new values. The
SET
clause indicates which columns to modify and the values they should be given. Each value can be given as an expression, or the keyword
DEFAULT
to set a column explicitly to its default value. The
WHERE
clause, if given, specifies the conditions that identify which rows to update. With no
WHERE
clause, all rows are updated. If the ORDER BY clause is specified, the rows are updated in the order that is specified. The LIMIT clause places a
limit on the number of rows that can be updated.

```

Until [MariaDB 10.3.2](#), for the multiple-table syntax,

```

UPDATE
updates rows in each table named in table_references that satisfy the conditions. In this case, ORDER BY and LIMIT cannot be used. This
restriction was lifted in MariaDB 10.3.2 and both clauses can be used with multiple-table updates. An
UPDATE
can also reference tables which are located in different databases; see Identifier Qualifiers for the syntax.

```

where\_condition  
is an expression that evaluates to true for each row to be updated.

```

table_references
and
where_condition
are as specified as described in SELECT.

```

For single-table updates, assignments are evaluated in left-to-right order, while for multi-table updates, there is no guarantee of a particular order. If the SIMULTANEOUS\_ASSIGNMENT sql\_mode (available from [MariaDB 10.3.5](#)) is set, UPDATE statements evaluate all assignments simultaneously.

You need the

```

UPDATE
privilege only for columns referenced in an
UPDATE
that are actually updated. You need only the SELECT privilege for any columns that are read but not modified. See GRANT.

```

The

```

UPDATE
statement supports the following modifiers:

```

- If you use the  
LOW\_PRIORITY

keyword, execution of the  
UPDATE  
is delayed until no other clients are reading from the table. This affects only storage engines that use only table-level locking (MyISAM, MEMORY, MERGE). See [HIGH\\_PRIORITY](#) and [LOW\\_PRIORITY](#) clauses for details.

- If you use the  
IGNORE  
keyword, the update statement does not abort even if errors occur during the update. Rows for which duplicate-key conflicts occur are not updated. Rows for which columns are updated to values that would cause data conversion errors are updated to the closest valid values instead.

## PARTITION

See [Partition Pruning and Selection](#) for details.

## FOR PORTION OF

MariaDB starting with 10.4.3  
See [Application Time Periods - Updating by Portion](#).

## UPDATE Statements With the Same Source and Target

MariaDB starting with 10.3.2  
From MariaDB 10.3.2 , UPDATE statements may have the same source and target.

For example, given the following table:

```
DROP TABLE t1;
CREATE TABLE t1 (c1 INT, c2 INT);
INSERT INTO t1 VALUES (10,10), (20,20);
```

Until MariaDB 10.3.1 , the following UPDATE statement would not work:

```
UPDATE t1 SET c1=c1+1 WHERE c2=(SELECT MAX(c2) FROM t1);
ERROR 1093 (HY000): Table 't1' is specified twice,
both as a target for 'UPDATE' and as a separate source for data
```

From MariaDB 10.3.2 , the statement executes successfully:

```
UPDATE t1 SET c1=c1+1 WHERE c2=(SELECT MAX(c2) FROM t1);

SELECT * FROM t1;
+-----+
| c1 | c2 |
+-----+
| 10 | 10 |
| 21 | 20 |
+-----+
```

## Example

Single-table syntax:

```
UPDATE table_name SET column1 = value1, column2 = value2 WHERE id=100;
```

Multiple-table syntax:

```
UPDATE tab1, tab2 SET tab1.column1 = value1, tab1.column2 = value2 WHERE tab1.id = tab2.id;
```

## See Also

- [How IGNORE works](#)
- [SELECT](#)
- [ORDER BY](#)
- [LIMIT](#)
- [Identifier Qualifiers](#)

## 1.1.2.1.22 IGNORE

The

IGNORE

option tells the server to ignore some common errors.

IGNORE

can be used with the following statements:

- [DELETE](#)
- [INSERT](#) (see also [INSERT IGNORE](#) )
- [LOAD DATA INFILE](#)
- [UPDATE](#)
- [ALTER TABLE](#)
- [CREATE TABLE ... SELECT](#)
- [INSERT ... SELECT](#)

The logic used:

- Variables out of ranges are replaced with the maximum/minimum value.

- [SQL\\_MODEs](#)

```
STRICT_TRANS_TABLES  
,  
STRICT_ALL_TABLES  
,  
NO_ZERO_IN_DATE  
,  
NO_ZERO_DATE  
are ignored.
```

- Inserting

```
NULL  
in a  
NOT NULL  
field will insert 0 ( in a numerical field), 0000-00-00 ( in a date field) or an empty string ( in a character field).
```

- Rows that cause a duplicate key error or break a foreign key constraint are not inserted, updated, or deleted.

The following errors are ignored:

| Error number | Symbolic error name                | Description                                   |
|--------------|------------------------------------|-----------------------------------------------|
| 1022         | ER_DUP_KEY                         | Can't write; duplicate key in table '%s'      |
| 1048         | ER_BAD_NULL_ERROR                  | Column '%s' cannot be null                    |
| 1062         | ER_DUP_ENTRY                       | Duplicate entry '%s' for key %d               |
| 1242         | ER_SUBQUERY_NO_1_ROW               | Subquery returns more than 1 row              |
| 1264         | ER_WARN_DATA_OUT_OF_RANGE          | Out of range value for column '%s' at row %ld |
| 1265         | WARN_DATA_TRUNCATED                | Data truncated for column '%s' at row %ld     |
| 1292         | ER_TRUNCATED_WRONG_VALUE           | Truncated incorrect %s value: '%s'            |
| 1366         | ER_TRUNCATED_WRONG_VALUE_FOR_FIELD | Incorrect integer value                       |
| 1369         | ER_VIEW_CHECK_FAILED               | CHECK OPTION failed '%s.%s'                   |

|      |                                           |                                                                       |
|------|-------------------------------------------|-----------------------------------------------------------------------|
| 1451 | ER_ROW_IS_REFERENCED_2                    | Cannot delete or update a parent row                                  |
| 1452 | ER_NO_REFERENCED_ROW_2                    | Cannot add or update a child row: a foreign key constraint fails (%s) |
| 1526 | ER_NO_PARTITION_FOR_GIVEN_VALUE           | Table has no partition for value %s                                   |
| 1586 | ER_DUP_ENTRY_WITH_KEY_NAME                | Duplicate entry '%s' for key '%s'                                     |
| 1591 | ER_NO_PARTITION_FOR_GIVEN_VALUE_SILENT    | Table has no partition for some existing values                       |
| 1748 | ER_ROW_DOES_NOT_MATCH_GIVEN_PARTITION_SET | Found a row not matching the given partition set                      |

Ignored errors normally generate a warning.

A property of the

IGNORE

clause consists in causing transactional engines and non-transactional engines (like XtraDB and Aria) to behave the same way. For example, normally a multi-row insert which tries to violate a

UNIQUE

constraint is completely rolled back on XtraDB/InnoDB, but might be partially executed on Aria. With the

IGNORE

clause, the statement will be partially executed in both engines.

Duplicate key errors also generate warnings. The `OLD_MODE` server variable can be used to prevent this.

## 1.1.2.1.23 System-Versioned Tables

### 1.1.2.2 ANALYZE and EXPLAIN Statements

#### 1.1.2.2.1 ANALYZE FORMAT=JSON

##### Contents

- 1. Basic Execution Data
- 2. Advanced Execution Data
- 3. Data About Individual Query Plan Nodes
- 4. Use Cases

`ANALYZE FORMAT=JSON` is a mix of the `EXPLAIN FORMAT=JSON` and `ANALYZE` statement features. The `ANALYZE FORMAT=JSON $statement` will execute `$statement`, and then print the output of `EXPLAIN FORMAT=JSON`, amended with data from the query execution.

## Basic Execution Data

You can get the following also from tabular

`ANALYZE`

statement form:

- `r_rows` is provided for any node that reads rows. It shows how many rows were read, on average
- `r_filtered` is provided whenever there is a condition that is checked. It shows the percentage of rows left after checking the condition.

# Advanced Execution Data

The most important data not available in the regular tabula

ANALYZE  
statement are:

- **r\_loops**  
field. This shows how many times the node was executed. Most query plan elements have this field.
- **r\_total\_time\_ms**  
field. It shows how much time in total was spent executing this node. If the node has subnodes, their execution time is included.
- **r\_buffer\_size**  
field. Query plan nodes that make use of buffers report the size of buffer that was used.

## Data About Individual Query Plan Nodes

- **filesort**  
node reports whether sorting was done with  
`LIMIT n`  
parameter, and how many rows were in the sort result.
- **block-nl-join**  
node has  
**r\_loops**  
field, which allows to tell whether  
Using join buffer  
was efficient
- **range-checked-for-each-record**  
reports counters that show the result of the check.
- **expression-cache**  
is used for subqueries, and it reports how many times the cache was used, and what cache hit ratio was.
- **union\_result**  
node has  
**r\_rows**  
so one can see how many rows were produced after UNION operation
- and so forth

## Use Cases

See [Examples of ANALYZE FORMAT=JSON](#).

### 1.1.2.2 ANALYZE FORMAT=JSON Examples

#### Example #1

Customers who have ordered more than 1M goods.

```
ANALYZE FORMAT=JSON
SELECT COUNT(*)
FROM customer
WHERE
  (SELECT SUM(o_totalprice) FROM orders WHERE o_custkey=c_custkey) > 1000*1000;
```

The query takes 40 seconds over cold cache

```

EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "r_loops": 1,
    "r_total_time_ms": 39872,
    "table": {
      "table_name": "customer",
      "access_type": "index",
      "key": "i_c_nationkey",
      "key_length": "5",
      "used_key_parts": ["c_nationkey"],
      "r_loops": 1,
      "rows": 150303,
      "r_rows": 150000,
      "r_total_time_ms": 270.3,
      "filtered": 100,
      "r_filtered": 60.691,
      "attached_condition": "((subquery#2) > <cache>((1000 * 1000)))",
      "using_index": true
    },
    "subqueries": [
      {
        "query_block": {
          "select_id": 2,
          "r_loops": 150000,
          "r_total_time_ms": 39531,
          "table": {
            "table_name": "orders",
            "access_type": "ref",
            "possible_keys": ["i_o_custkey"],
            "key": "i_o_custkey",
            "key_length": "5",
            "used_key_parts": ["o_custkey"],
            "ref": ["dbt3sf1.customer.c_custkey"],
            "r_loops": 150000,
            "rows": 7,
            "r_rows": 10,
            "r_total_time_ms": 39208,
            "filtered": 100,
            "r_filtered": 100
          }
        }
      }
    ]
  }
}

```

#### ANALYZE

shows that 39.2 seconds were spent in the subquery, which was executed 150K times (for every row of outer table).

### 1.1.2.2.3 ANALYZE Statement

#### Contents

1. [Description](#)
2. [Command Output](#)
3. [Interpreting the Output](#)
  1. [Joins](#)
  2. [Meaning of NULL in r\\_rows and r\\_filtered](#)
4. [ANALYZE FORMAT=JSON](#)
5. [Notes](#)
6. [See Also](#)

### Description

The

ANALYZE statement  
is similar to the  
EXPLAIN statement

```
ANALYZE statement  
will invoke the optimizer, execute the statement, and then produce  
EXPLAIN  
output instead of the result set. The  
EXPLAIN  
output will be annotated with statistics from statement execution.
```

This lets one check how close the optimizer's estimates about the query plan are to the reality.

```
ANALYZE  
produces an overview, while the
```

```
ANALYZE FORMAT=JSON
```

command provides a more detailed view of the query plan and the query execution.

The syntax is

```
ANALYZE explainable_statement;
```

where the statement is any statement for which one can run

```
EXPLAIN
```

## Command Output

Consider an example:

```
ANALYZE SELECT * FROM tbl1  
WHERE key1  
    BETWEEN 10 AND 200 AND  
col1 LIKE 'foo%'\\G
```

```
***** 1. row *****  
    id: 1  
select_type: SIMPLE  
    table: tbl1  
        type: range  
possible_keys: key1  
    key: key1  
key_len: 5  
    ref: NULL  
    rows: 181  
r_rows: 181  
filtered: 100.00  
r_filtered: 10.50  
Extra: Using index condition; Using where
```

Compared to

```
EXPLAIN  
,  
ANALYZE  
produces two extra columns:
```

- **r\_rows**  
is an observation-based counterpart of the **rows** column. It shows how many rows were actually read from the table.
- **r\_filtered**  
is an observation-based counterpart of the **filtered** column. It shows which fraction of rows was left after applying the WHERE condition.

## Interpreting the Output

### Joins

Let's consider a more complicated example.

```

ANALYZE SELECT *
FROM orders, customer
WHERE
customer.c_custkey=orders.o_custkey AND
customer.c_acctbal < 0 AND
orders.o_totalprice > 200*1000

```

| id   select_type   table   type   possible_keys   key   key_len   ref   rows   r_rows   filtered |
|--------------------------------------------------------------------------------------------------|
| 1   SIMPLE   customer   ALL   PRIMARY,...   NULL   NULL   NULL   149095   150000   18.08         |
| 1   SIMPLE   orders   ref   i_o_custkey   i_o_custkey   5   customer.c_custkey   7   10   100.00 |
|                                                                                                  |

Here, one can see that

- For table customer, **customer.rows=149095, customer.r\_rows=150000**. The estimate for number of rows we will read was fairly precise
- **customer.filtered=18.08, customer.r\_filtered=9.13**. The optimizer somewhat overestimated the number of records that will match selectivity of condition attached to `customer` table (in general, when you have a full scan and r\_filtered is less than 15%, it's time to consider adding an appropriate index).
- For table orders, **orders.rows=7, orders.r\_rows=10**. This means that on average, there are 7 orders for a given c\_custkey, but in our case there were 10, which is close to the expectation (when this number is consistently far from the expectation, it may be time to run ANALYZE TABLE, or even edit the table statistics manually to get better query plans).
- **orders.filtered=100, orders.r\_filtered=30.03**. The optimizer didn't have any way to estimate which fraction of records will be left after it checks the condition that is attached to table orders (it's orders.o\_totalprice > 200\*1000). So, it used 100%. In reality, it is 30%. 30% is typically not selective enough to warrant adding new indexes. For joins with many tables, it might be worth to collect and use [column statistics](#) for columns in question, this may help the optimizer to pick a better query plan.

## Meaning of NULL in r\_rows and r\_filtered

Let's modify the previous example slightly

```

ANALYZE SELECT *
FROM orders, customer
WHERE
customer.c_custkey=orders.o_custkey AND
customer.c_acctbal < -0 AND
customer.c_comment LIKE '%foo%' AND
orders.o_totalprice > 200*1000;

```

| id   select_type   table   type   possible_keys   key   key_len   ref   rows   r_rows   filtered   |
|----------------------------------------------------------------------------------------------------|
| 1   SIMPLE   customer   ALL   PRIMARY,...   NULL   NULL   NULL   149095   150000   18.08           |
| 1   SIMPLE   orders   ref   i_o_custkey   i_o_custkey   5   customer.c_custkey   7   NULL   100.00 |

Here, one can see that **orders.r\_rows=NULL** and **orders.r\_filtered=NULL**. This means that table orders was not scanned even once. Indeed, we can also see **customer.r\_filtered=0.00**. This shows that a part of WHERE attached to table `customer` was never satisfied (or, satisfied in less than 0.01% of cases).

## ANALYZE FORMAT=JSON

[ANALYZE FORMAT=JSON](#) produces JSON output. It produces much more information than tabular

ANALYZE

## Notes

- - ANALYZE UPDATE  
or
  - ANALYZE DELETE  
will actually make updates/deletes (
  - ANALYZE SELECT  
will perform the select operation and then discard the resultset).
- PostgreSQL has a similar command,  
EXPLAIN ANALYZE

- The [EXPLAIN in the slow query log](#) feature allows MariaDB to have  
ANALYZE  
output of slow queries printed into the [slow query log](#) (see [MDEV-6388](#) ).

## See Also

- [ANALYZE FORMAT=JSON](#)
- [ANALYZE TABLE](#)
- JIRA task for ANALYZE statement, [MDEV-406](#)

## 1.1.2.2.4 EXPLAIN

### Syntax

```
EXPLAIN tbl_name
```

Or

```
EXPLAIN [EXTENDED | PARTITIONS]
{SELECT select_options | UPDATE update_options | DELETE delete_options}
```

### Contents

- [Syntax](#)
- [Description](#)
- [Columns in EXPLAIN ... SELECT](#)
  - "Select\_type" Column
  - "Type" Column
  - "Extra" Column
- [EXPLAIN EXTENDED](#)
- [Examples](#)
  - [Example of ref\\_or\\_null Optimization](#)
- [See Also](#)

### Description

The

```
EXPLAIN
statement can be used either as a synonym for
DESCRIBE
or as a way to obtain information about how MariaDB executes a
SELECT
,
UPDATE
or
DELETE
statement:
```

- 'EXPLAIN tbl\_name'  
is synonymous with  
,

```
DESCRIBE
```

```
tbl_name'
or
'
```

```
SHOW COLUMNS
```

```
FROM tbl_name'
```

- When you precede a  
SELECT  
,
- UPDATE  
or a

DELETE  
statement with the keyword  
EXPLAIN  
, MariaDB displays information from the optimizer about the query execution plan. That is, MariaDB explains how it would process the  
SELECT  
,

UPDATE  
or  
DELETE  
, including information about how tables are joined and in which order.  
EXPLAIN EXTENDED  
can be used to provide additional information.

- EXPLAIN PARTITIONS  
is useful only when examining queries involving partitioned tables.

For details, see [Partition pruning and selection](#).

- **ANALYZE statement** performs the query as well as producing EXPLAIN output, and provides actual as well as estimated statistics.

- EXPLAIN  
output can be printed in the [slow query log](#). See [EXPLAIN in the Slow Query Log](#) for details.

**SHOW EXPLAIN** shows the output of a running statement. In some cases, its output can be closer to reality than

EXPLAIN

The **ANALYZE statement** runs a statement and returns information about its execution plan. It also shows additional columns, to check how much the optimizer's estimation about filtering and found rows are close to reality.

There is an online [EXPLAIN Analyzer](#) that you can use to share

EXPLAIN

and

EXPLAIN EXTENDED

output with others.

EXPLAIN  
can acquire metadata locks in the same way that  
SELECT  
does, as it needs to know table metadata and, sometimes, data as well.

## Columns in EXPLAIN ... SELECT

| Column name   | Description                                                                                         |
|---------------|-----------------------------------------------------------------------------------------------------|
| id            | Sequence number that shows in which order tables are joined.                                        |
| select_type   | What kind of<br>SELECT<br>the table comes from.                                                     |
| table         | Alias name of table. Materialized temporary tables for sub queries are named <subquery#>            |
| type          | How rows are found from the table (join type).                                                      |
| possible_keys | keys in table that could be used to find rows in the table                                          |
| key           | The name of the key that is used to retrieve rows.<br>NULL<br>is no key was used.                   |
| key_len       | How many bytes of the key that was used (shows if we are using only parts of the multi-column key). |

|       |                                                                             |
|-------|-----------------------------------------------------------------------------|
| ref   | The reference that is used as the key value.                                |
| rows  | An estimate of how many rows we will find in the table for each key lookup. |
| Extra | Extra information about this join.                                          |

Here are descriptions of the values for some of the more complex columns in

```
EXPLAIN ... SELECT
```

:

### "Select\_type" Column

The

`select_type`

column can have the following values:

| Value                 | Description                                            | Comment                                                                                                                                                                            |
|-----------------------|--------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DEPENDENT<br>SUBQUERY | The<br>SUBQUERY<br>is<br>DEPENDENT                     |                                                                                                                                                                                    |
| DEPENDENT UNION       | The<br>UNION<br>is<br>DEPENDENT                        |                                                                                                                                                                                    |
| DERIVED               | The<br>SELECT<br>is<br>DERIVED<br>from the<br>PRIMARY  |                                                                                                                                                                                    |
| MATERIALIZED          | The<br>SUBQUERY<br>is<br><b>MATERIALIZED</b>           | Materialized tables will be populated at first access and will be accessed by the primary key (= one key lookup). Number of rows in EXPLAIN shows the cost of populating the table |
| PRIMARY               | The<br>SELECT<br>is a<br>PRIMARY<br>one.               |                                                                                                                                                                                    |
| SIMPLE                | The<br>SELECT<br>is a<br>SIMPLE<br>one.                |                                                                                                                                                                                    |
| SUBQUERY              | The<br>SELECT<br>is a<br>SUBQUERY<br>of the<br>PRIMARY |                                                                                                                                                                                    |

|                      |                                                |
|----------------------|------------------------------------------------|
| UNCACHEABLE SUBQUERY | The SUBQUERY is UNCACHEABLE                    |
| UNCACHEABLE UNION    | The UNION is UNCACHEABLE                       |
| UNION                | The SELECT is a UNION of the PRIMARY           |
| RESULT               | The result of the UNION                        |
| LATERAL DERIVED      | The SELECT uses a Lateral Derived optimization |

## "Type" Column

This column contains information on how the table is accessed.

| Value          | Description                                                                                                                                                                                                                      |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ALL            | A full table scan is done for the table (all rows are read). This is bad if the table is large and the table is joined against a previous table! This happens when the optimizer could not find any usable index to access rows. |
| const          | There is only one possibly matching row in the table. The row is read before the optimization phase and all columns in the table are treated as constants.                                                                       |
| eq_ref         | A unique index is used to find the rows. This is the best possible plan to find the row.                                                                                                                                         |
| fulltext       | A fulltext index is used to access the rows.                                                                                                                                                                                     |
| index_merge    | A 'range' access is done for several index and the found rows are merged. The key column shows which keys are used.                                                                                                              |
| index_subquery | This is similar as ref, but used for sub queries that are transformed to key lookups.                                                                                                                                            |
| index          | A full scan over the used index. Better than ALL but still bad if index is large and the table is joined against a previous table.                                                                                               |
| range          | The table will be accessed with a key over one or more value ranges.                                                                                                                                                             |

|                 |                                                                                                                                                 |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| ref_or_null     | Like 'ref' but in addition another search for the 'null' value is done if the first value was not found. This happens usually with sub queries. |
| ref             | A non unique index or prefix of an unique index is used to find the rows. Good if the prefix doesn't match many rows.                           |
| system          | The table has 0 or 1 rows.                                                                                                                      |
| unique_subquery | This is similar as eq_ref, but used for sub queries that are transformed to key lookups                                                         |

## "Extra" Column

This column consists of one or more of the following values, separated by ','

*Note that some of these values are detected after the optimization phase.*

The optimization phase can do the following changes to the

WHERE  
clause:

- Add the expressions from the  
ON  
and  
USING  
clauses to the  
WHERE  
clause.
- Constant propagation: If there is  
column=constant  
, replace all column instances with this constant.
- Replace all columns from '  
const  
' tables with their values.
- Remove the used key columns from the  
WHERE  
(as this will be tested as part of the key lookup).
- Remove impossible constant sub expressions. For example  
WHERE '(a=1 and a=2) OR b=1'  
becomes  
'b=1'
- Replace columns with other columns that has identical values: Example:  
WHERE  
  
a=b  
and  
a=c  
may be treated as  
'WHERE a=b and a=c and b=c'

- Add extra conditions to detect impossible row conditions earlier. This happens mainly with  
OUTER JOIN  
where we in some cases add detection of  
NULL  
values in the  
WHERE  
(Part of'  
Not exists  
'optimization). This can cause an unexpected'  
Using where  
' in the Extra column.
- For each table level we remove expressions that have already been tested when we read the previous row. Example: When joining tables  
t1  
with  
t2

using the following  
 WHERE 't1.a=1 and t1.a=t2.b'  
 , we don't have to test  
 't1.a=1'  
 when checking rows in  
 t2  
 as we already know that this expression is true.

| Value                                                               | Description                                                                                                                                                                                                                                                                         |
|---------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| const<br>row not found                                              | The table was a system table (a table with exactly one row), but no row was found.                                                                                                                                                                                                  |
| Distinct                                                            | If distinct optimization (remove duplicates) was used. This is marked only for the last table in the<br><pre>SELECT .</pre>                                                                                                                                                         |
| Full<br>scan on NULL key                                            | The table is a part of the sub query and if the value that is used to match the sub query will be<br><pre>NULL , we will do a full table scan.</pre>                                                                                                                                |
| Impossible HAVING                                                   | The used<br><pre>HAVING clause is always false so the SELECT will return no rows.</pre>                                                                                                                                                                                             |
| Impossible WHERE<br>noticed<br>after<br>reading<br>const<br>tables. | The used<br><pre>WHERE clause is always false so the SELECT will return no rows. This case was detected after we had read all 'const' tables and used the column values as constant in the WHERE clause. For example: WHERE const_column=5 and const_column had a value of 4.</pre> |
| Impossible WHERE                                                    | The used<br><pre>WHERE clause is always false so the SELECT will return no rows. For example: WHERE 1=2</pre>                                                                                                                                                                       |
| No<br>matching min/max<br>row                                       | During early optimization of<br><pre>MIN() / MAX() values it was detected that no row could match the WHERE clause. The MIN() / MAX() function will return NULL .</pre>                                                                                                             |

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| no matching row<br>in const table                 | The table was a const table (a table with only one possible matching row), but no row was found.                                                                                                                                                                                                                                                                                                                                                                     |
| No tables used                                    | The<br>SELECT<br>was a sub query that did not use any tables. For example a there was no<br>FROM<br>clause or a<br>FROM DUAL<br>clause.                                                                                                                                                                                                                                                                                                                              |
| Not exists                                        | Stop searching after more row if we find one single matching row. This optimization is used with<br>LEFT JOIN<br>where one is explicitly searching for rows that doesn't exists in the<br>LEFT JOIN TABLE<br>. Example:<br>SELECT * FROM t1 LEFT JOIN t2 on (...) WHERE t2.not_null_column IS NULL<br>. As<br>t2.not_null_column<br>can only be<br>NULL<br>if there was no matching row for on condition, we can stop searching if we find a single matching row.    |
| Open_frm_only                                     | For<br>information_schema<br>tables. Only the<br>frm<br>(table definition file was opened) was opened for each matching row.                                                                                                                                                                                                                                                                                                                                         |
| Open_full_table                                   | For<br>information_schema<br>tables. A full table open for each matching row is done to retrieve the requested information. (Slow)                                                                                                                                                                                                                                                                                                                                   |
| Open_trigger_only                                 | For<br>information_schema<br>tables. Only the trigger file definition was opened for each matching row.                                                                                                                                                                                                                                                                                                                                                              |
| Range checked for each record<br>(index map: ...) | This only happens when there was no good default index to use but there may some index that could be used when we can treat all columns from previous table as constants. For each row combination the optimizer will decide which index to use (if any) to fetch a row from this table. This is not fast, but faster than a full table scan that is the only other choice. The index map is a bitmask that shows which index are considered for each row condition. |
| Scanned 0/1/all databases                         | For<br>information_schema<br>tables. Shows how many times we had to do a directory scan.                                                                                                                                                                                                                                                                                                                                                                             |

|                              |                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Select tables optimized away | All tables in the join was optimized away. This happens when we are only using COUNT(*) , MIN() and MAX() functions in the SELECT and we were able to replace all of these with constants.                                                                                                                                                              |
| Skip_open_table              | For information_schema tables. The queried table didn't need to be opened.                                                                                                                                                                                                                                                                              |
| unique row not found         | The table was detected to be a const table (a table with only one possible matching row) during the early optimization phase, but no row was found.                                                                                                                                                                                                     |
| Using filesort               | Filesort is needed to resolve the query. This means an extra phase where we first collect all columns to sort, sort them with a disk based merge sort and then use the sorted set to retrieve the rows in sorted order. If the column set is small, we store all the columns in the sort file to not have to go to the database to retrieve them again. |
| Using index                  | Only the index is used to retrieve the needed information from the table. There is no need to perform an extra seek to retrieve the actual record.                                                                                                                                                                                                      |
| Using index condition        | Like ' Using where ' but the where condition is pushed down to the table engine for internal optimization at the index level.                                                                                                                                                                                                                           |
| Using index condition(BKA)   | Like ' Using index condition ' but in addition we use batch key access to retrieve rows.                                                                                                                                                                                                                                                                |
| Using index for group-by     | The index is being used to resolve a GROUP BY or DISTINCT query. The rows are not read. This is very efficient if the table has a lot of identical index entries as duplicates are quickly jumped over.                                                                                                                                                 |
| Using intersect(...)         | For index_merge joins. Shows which index are part of the intersect.                                                                                                                                                                                                                                                                                     |
| Using join buffer            | We store previous row combinations in a row buffer to be able to match each row against all of the rows combinations in the join buffer at one go.                                                                                                                                                                                                      |
| Using sort_union(...)        | For index_merge joins. Shows which index are part of the union.                                                                                                                                                                                                                                                                                         |
| Using temporary              | A temporary table is created to hold the result. This typically happens if you are using GROUP BY , DISTINCT or ORDER BY .                                                                                                                                                                                                                              |

|                                      |                                                                                                                                                                                                                                               |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| where                                | Using<br>A<br>WHERE<br>expression (in additional to the possible key lookup) is used to check if the row should be accepted. If you don't have 'Using where' together with a join type of<br>ALL<br>, you are probably doing something wrong! |
| where<br>with<br>pushed<br>condition | Like '<br>Using where<br>' but the where condition is pushed down to the table engine for internal optimization at the row level.                                                                                                             |
| buffer                               | Using<br>The<br>UPDATE<br>statement will first buffer the rows, and then run the updates, rather than do updates on the fly. See <a href="#">Using Buffer UPDATE Algorithm</a> for a detailed explanation.                                    |

## EXPLAIN EXTENDED

The  
EXTENDED  
keyword adds another column, *filtered* , to the output. This is a percentage estimate of the table rows that will be filtered by the condition.

An  
EXPLAIN EXTENDED  
will always throw a warning, as it adds extra *Message* information to a subsequent

`SHOW WARNINGS`

statement. This includes what the  
`SELECT`  
query would look like after optimizing and rewriting rules are applied and how the optimizer qualifies columns and tables.

## Examples

As synonym for

`DESCRIBE`  
or  
`SHOW COLUMNS FROM`  
:

```
DESCRIBE city;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra          |
+-----+-----+-----+-----+-----+
Id	int(11)	NO	PRI	NULL	auto_increment
Name	char(35)	YES		NULL	
Country	char(3)	NO	UNI		
District	char(20)	YES	MUL		
Population	int(11)	YES		NULL	
+-----+-----+-----+-----+-----+
```

A simple set of examples to see how

`EXPLAIN`  
can identify poor index usage:

```

CREATE TABLE IF NOT EXISTS `employees_example` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `first_name` varchar(30) NOT NULL,
  `last_name` varchar(40) NOT NULL,
  `position` varchar(25) NOT NULL,
  `home_address` varchar(50) NOT NULL,
  `home_phone` varchar(12) NOT NULL,
  `employee_code` varchar(25) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `employee_code` (`employee_code`),
  KEY `first_name` (`first_name`,`last_name`)
) ENGINE=Aria;

INSERT INTO `employees_example` (`first_name`, `last_name`, `position`, `home_address`, `home_phone`, `employee_code`)
VALUES
('Mustapha', 'Mond', 'Chief Executive Officer', '692 Promiscuous Plaza', '326-555-3492', 'MM1'),
('Henry', 'Foster', 'Store Manager', '314 Savage Circle', '326-555-3847', 'HF1'),
('Bernard', 'Marx', 'Cashier', '1240 Ambient Avenue', '326-555-8456', 'BM1'),
('Lenina', 'Crowne', 'Cashier', '281 Bumblepuppy Boulevard', '328-555-2349', 'LC1'),
('Fanny', 'Crowne', 'Restocker', '1023 Bokanovsky Lane', '326-555-6329', 'FC1'),
('Helmholtz', 'Watson', 'Janitor', '944 Soma Court', '329-555-2478', 'HW1');

```

```
SHOW INDEXES FROM employees_example;
```

| Table             | Non_unique | Key_name      | Seq_in_index | Column_name   | Collation | Cardinality | Sub_part | Packed |
|-------------------|------------|---------------|--------------|---------------|-----------|-------------|----------|--------|
| employees_example | 0          | PRIMARY       | 1            | id            | A         | 7           | NULL     | NULL   |
| employees_example | 0          | employee_code | 1            | employee_code | A         | 7           | NULL     | NULL   |
| employees_example | 1          | first_name    | 1            | first_name    | A         | NULL        | NULL     | NULL   |
| employees_example | 1          | first_name    | 2            | last_name     | A         | NULL        | NULL     | NULL   |

```
SELECT
```

```
on a primary key:
```

```
EXPLAIN SELECT * FROM employees_example WHERE id=1;
```

| id | select_type | table             | type  | possible_keys | key     | key_len | ref   | rows | Extra |
|----|-------------|-------------------|-------|---------------|---------|---------|-------|------|-------|
| 1  | SIMPLE      | employees_example | const | PRIMARY       | PRIMARY | 4       | const | 1    |       |

The type is `const`, which means that only one possible result could be returned. Now, returning the same record but searching by their phone number:

```
EXPLAIN SELECT * FROM employees_example WHERE home_phone='326-555-3492';
```

| id | select_type | table             | type | possible_keys | key  | key_len | ref  | rows | Extra       |
|----|-------------|-------------------|------|---------------|------|---------|------|------|-------------|
| 1  | SIMPLE      | employees_example | ALL  | NULL          | NULL | NULL    | NULL | 6    | Using where |

Here, the type is `All`, which means no index could be used. Looking at the rows count, a full table scan (all six rows) had to be performed in order to retrieve the record. If it's a requirement to search by phone number, an index will have to be created.

```
SHOW EXPLAIN
```

```
example:
```

```
SHOW EXPLAIN FOR 1;
```

| id | select_type | table | type  | possible_keys | key | key_len | ref  | rows    | Extra       |
|----|-------------|-------|-------|---------------|-----|---------|------|---------|-------------|
| 1  | SIMPLE      | tbl   | index | NULL          | a   | 5       | NULL | 1000107 | Using index |

```
1 row in set, 1 warning (0.00 sec)
```

## Example of

### ref\_or\_null

## Optimization

```
SELECT * FROM table_name
WHERE key_column=expr OR key_column IS NULL;
```

ref\_or\_null  
is something that often happens when you use subqueries with  
NOT IN  
as then one has to do an extra check for  
NULL  
values if the first value didn't have a matching row.

## See Also

- [SHOW EXPLAIN](#)
- [Ignored Indexes](#)

## 1.1.2.2.5 EXPLAIN ANALYZE

The syntax for the

EXPLAIN ANALYZE  
feature was changed to  
ANALYZE statement  
, available since MariaDB 10.1.0 . See [ANALYZE statement](#) .

## 1.1.2.2.6 EXPLAIN FORMAT=JSON

MariaDB starting with 10.1.2

Starting from version 10.1.2, MariaDB supports the EXPLAIN FORMAT=JSON syntax.

### Contents

1. [Synopsis](#)
2. [Output is different from MySQL](#)
3. [Output format](#)
4. [See also](#)

## Synopsis

EXPLAIN FORMAT=JSON  
is a variant of

[EXPLAIN](#)

command that produces output in JSON form. The output always has one row which has only one column titled "  
JSON  
". The contents are a JSON representation of the query plan, formatted for readability:

```
EXPLAIN FORMAT=JSON SELECT * FROM t1 WHERE col1=1\G
```

```
***** 1. row *****
EXPLAIN: {
  "query_block": {
    "select_id": 1,
    "table": {
      "table_name": "t1",
      "access_type": "ALL",
      "rows": 1000,
      "filtered": 100,
      "attached_condition": "(t1.col1 = 1)"
    }
  }
}
```

# Output is different from MySQL

The output of MariaDB's

`EXPLAIN FORMAT=JSON`

is different from

`EXPLAIN FORMAT=JSON`

in MySQL. The reasons for that are:

- MySQL's output has deficiencies. Some are listed here:

`EXPLAIN FORMAT=JSON in MySQL`

- The output of MySQL's

`EXPLAIN FORMAT=JSON`

is not defined. Even MySQL Workbench has trouble parsing it (see this [blog post](#)).

- MariaDB has query optimizations that MySQL does not have. Ergo, MariaDB generates query plans that MySQL does not generate.

A (as yet incomplete) list of how MariaDB's output is different from MySQL can be found here: [EXPLAIN FORMAT=JSON differences from MySQL](#).

## Output format

TODO: MariaDB's output format description.

## See also

- 

`ANALYZE FORMAT=JSON`

produces output like

`EXPLAIN FORMAT=JSON`

, but amended with the data from query execution.

## 1.1.2.2.7 SHOW EXPLAIN

### Contents

- [Syntax](#)
- [Description](#)
  - [Possible Errors](#)
  - [Differences Between SHOW EXPLAIN and EXPLAIN Outputs](#)
    - [Background](#)
    - [List of Recorded Differences](#)
    - [Required Permissions](#)
  - [See Also](#)

## Syntax

```
SHOW EXPLAIN FOR <thread_id>;
```

## Description

The

`SHOW EXPLAIN`

command allows one to get an [EXPLAIN](#) (that is, a description of a query plan) of a query running in a certain thread.

```
SHOW EXPLAIN FOR <thread_id>;
```

will produce an

`EXPLAIN`

output for the query that thread number

`thread_id`

is running. The thread id can be obtained with [SHOW PROCESSLIST](#).

```
SHOW EXPLAIN FOR 1;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | tbl | index | NULL | a | 5 | NULL | 1000107 | Using index |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

The output is always accompanied with a warning which shows the query the target thread is running (this shows what the EXPLAIN is for):

```
SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note | 1003 | select sum(a) from tbl |
+-----+-----+
1 row in set (0.00 sec)
```

## Possible Errors

The output can be only produced if the target thread is *currently* running a query, which has a ready query plan. If this is not the case, the output will be:

```
SHOW EXPLAIN FOR 2;
ERROR 1932 (HY000): Target is not running an EXPLAINable command
```

You will get this error when:

- the target thread is not running a command for which one can run EXPLAIN
- the target thread is running a command for which one can run EXPLAIN, but
  - there is no query plan yet (for example, tables are open and locks are acquired before the query plan is produced)

## Differences Between SHOW EXPLAIN and EXPLAIN Outputs

### Background

In MySQL,

```
EXPLAIN
execution takes a slightly different route from the way the real query (typically the
SELECT
) is optimized. This is unfortunate, and has caused a number of bugs in
EXPLAIN
. (For example, see MDEV-326 , MDEV-410 , and lp:1013343 . lp:992942 is not directly about
EXPLAIN
, but it also would not have existed if MySQL didn't try to delete parts of a query plan in the middle of the query)
```

```
SHOW EXPLAIN
examines a running
SELECT
, and hence its output may be slightly different from what
EXPLAIN SELECT
would produce. We did our best to make sure that either the difference is negligible, or
SHOW EXPLAIN
's output is closer to reality than
EXPLAIN
's output.
```

### List of Recorded Differences

- SHOW EXPLAIN may have Extra='no matching row in const table ', where

- ```

EXPLAIN
would produce Extra=
Impossible WHERE ...
'

• For queries with subqueries,
SHOW EXPLAIN
may print
select_type==PRIMARY
where regular
EXPLAIN
used to print
select_type==SIMPLE
, or vice versa.

```

## Required Permissions

Running

```

SHOW EXPLAIN
requires the same permissions as running
SHOW PROCESSLIST
would.

```

## See Also

- [EXPLAIN ANALYZE](#) , which will perform a query and outputs enhanced EXPLAIN results.
- It is also possible to save EXPLAIN into the slow query log .

## 1.1.2.2.8 Using Buffer UPDATE Algorithm

This article explains the [UPDATE](#) statement's *Using Buffer* algorithm.

Take the following table and query:

Name	Salary
Babatunde	1000
Jolana	1050
Pankaja	1300

```
UPDATE employees SET salary = salary+100 WHERE salary < 2000;
```

Suppose the *employees* table has an index on the *salary* column, and the optimizer decides to use a range scan on that index.

The optimizer starts a range scan on the *salary* index. We find the first record *Babatunde, 1000* . If we do an on-the-fly update, we immediately instruct the storage engine to change this record to be *Babatunde, 1000+100=1100* .

Then we proceed to search for the next record, and find *Jolana, 1050* . We instruct the storage engine to update it to be *Jolana, 1050+100=1150* .

Then we proceed to search for the next record ... and what happens next depends on the storage engine. In some storage engines, data changes are visible immediately, so we will find the *Babatunde, 1100* record that we wrote at the first step, modifying it again, giving Babatunde an undeserved raise. Then we will see Babatunde again and again, looping continually.

In order to prevent such situations, the optimizer checks whether the UPDATE statement is going to change key values for the keys it is using. In that case, it will use a different algorithm:

1. Scan everyone with "salary<2000", remembering the rowids of the rows in a buffer.
2. Read the buffer and apply the updates.

This way, each row will be updated only once.

The

```

Using buffer
EXPLAIN output indicates that the buffer as described above will be used.

```

## 1.1.2.3 BACKUP Commands

### 1.1.2.3.1 BACKUP STAGE

MariaDB starting with [10.4.1](#)

The

BACKUP STAGE  
commands were introduced in [MariaDB 10.4.1](#).

## Contents

1. [Syntax](#)
2. [Goals with BACKUP STAGE Commands](#)
3. [BACKUP STAGE Commands](#)
  1. [BACKUP STAGE START](#)
  2. [BACKUP STAGE FLUSH](#)
  3. [BACKUP STAGE BLOCK\\_DDL](#)
  4. [BACKUP STAGE BLOCK\\_COMMIT](#)
  5. [BACKUP STAGE END](#)
4. [Using BACKUP STAGE Commands with Backup Tools](#)
  1. [Using BACKUP STAGE Commands with mariabackup](#)
  2. [Using BACKUP STAGE Commands with Storage Snapshots](#)
5. [Privileges](#)
6. [Notes](#)
7. [See Also](#)

The

BACKUP STAGE  
commands are a set of commands to make it possible to make an efficient external backup tool.

## Syntax

```
BACKUP STAGE [START | FLUSH | BLOCK_DDL | BLOCK_COMMIT | END ]
```

In the following text, a transactional table means InnoDB or "InnoDB-like engine with redo log that can lock redo purges and can be copied without locks by an outside process".

## Goals with BACKUP STAGE Commands

- To be able to do a majority of the backup with the minimum possible server locks. Especially for transactional tables (InnoDB, MyRocks etc) there is only need for a very short block of new commits while copying statistics and log tables.
- DDL are only needed to be blocked for a very short duration of the backup while [mariabackup](#) is copying the tables affected by DDL during the initial part of the backup.
- Most non transactional tables (those that are not in use) will be copied during
  - BACKUP STAGE START
    - . The exceptions are system statistic and log tables that are not blocked during the backup until
    - BLOCK\_COMMIT
  - Should work efficiently with backup tools that use disk snapshots.
  - Should work as efficiently as possible for all table types that store data on the local disks.
  - As little copying as possible under higher level stages/locks. For example, .frm (dictionary) and .trn (trigger) files should be copying while copying the table data.

## BACKUP STAGE Commands

### BACKUP STAGE START

The

START  
stage is designed for the following tasks:

- Blocks purge of redo files for storage engines that needs this (Aria)
- Start logging of DDL commands into 'datadir'/ddl.log. This may take a short time as the command has to wait until there are no active DDL commands.

## BACKUP STAGE FLUSH

The

FLUSH

stage is designed for the following tasks:

- FLUSH all changes for inactive non-transactional tables, except for statistics and log tables.
- Close all tables that are not in use, to ensure they are marked as closed for the backup.
- BLOCK all new write locks for all non transactional tables (except statistics and log tables). The command will not wait for tables that are in use by read-only transactions.

DDLs don't have to be blocked at this stage as they can't cause the table to be in an inconsistent state. This is true also for non-transactional tables.

## BACKUP STAGE BLOCK\_DDL

The

BLOCK\_DDL

stage is designed for the following tasks:

- Wait for all statements using write locked non-transactional tables to end.
- Blocks [CREATE TABLE](#), [DROP TABLE](#), [TRUNCATE TABLE](#), and [RENAME TABLE](#).
- Blocks also start off a new [ALTER TABLE](#) and the **final rename phase** of [ALTER TABLE](#). Running ALTER TABLES are not blocked.

## BACKUP STAGE BLOCK\_COMMIT

The

BLOCK\_COMMIT

stage is designed for the following tasks:

- Lock the binary log and commit/rollback to ensure that no changes are committed to any tables. If there are active commits or data to be copied to the binary log this will be allowed to finish. Active transactions will not affect

BLOCK\_COMMIT

- This doesn't lock temporary tables that are not used by replication. However these will be blocked when it's time to write to the binary log.
- Lock system log tables and statistics tables, flush them and mark them closed.

When the

BLOCK\_COMMIT

's stages return, this is the 'backup time'. Everything committed will be in the backup and everything not committed will roll back.

Transactional engines will continue to do changes to the redo log during the

BLOCK\_COMMIT

stage, but this is not important as all of these will roll back later as the changes will not be committed.

## BACKUP STAGE END

The

END

stage is designed for the following tasks:

- End DDL logging
- Free resources

## Using

### BACKUP STAGE Commands with Backup Tools

## Using

### BACKUP STAGE

## Commands with Mariabackup

The

`BACKUP STAGE`

commands are a set of commands to make it possible to make an efficient external backup tool. How [Mariabackup](#) uses these commands depends on whether you are using the version that is bundled with MariaDB Community Server or the version that is bundled with [MariaDB Enterprise Server](#). See [Mariabackup and BACKUP STAGE Commands](#) for some examples on how [Mariabackup](#) uses these commands.

If you would like to use a version of [Mariabackup](#) that uses the

`BACKUP STAGE`

commands in an efficient way, then one option is to use [MariaDB Enterprise Backup](#) that is bundled with [MariaDB Enterprise Server](#).

## Using

`BACKUP STAGE`

Commands with Storage Snapshots

The

`BACKUP STAGE`

commands are a set of commands to make it possible to make an efficient external backup tool. These commands could even be used by tools that perform backups by taking a snapshot of a file system, SAN, or some other kind of storage device. See [Storage Snapshots and BACKUP STAGE Commands](#) for some examples on how to use each

`BACKUP STAGE`

command in an efficient way.

## Privileges

`BACKUP STAGE`

requires the

`RELOAD`

privilege.

## Notes

- Only one connection can run

`BACKUP STAGE START`

. If a second connection tries, it will wait until the first one has executed

`BACKUP STAGE END`

- If the user skips a

`BACKUP STAGE`

, then all intermediate backup stages will automatically be run. This will allow us to add new stages within the

`BACKUP STAGE`

hierarchy in the future with even more precise locks without causing problems for tools using an earlier version of the

`BACKUP STAGE`

implementation.

- One can use the

`max_statement_time`

or

`lock_wait_timeout`

system variables to ensure that a

`BACKUP STAGE`

command doesn't block the server too long.

- DDL logging will only be available in [MariaDB Enterprise Server](#) 10.2 and later.

## See Also

- [BACKUP LOCK](#) Locking a table from DDL's.
- [MDEV-5336](#) . Implement BACKUP STAGE for safe external backups.

## 1.1.2.3.2 BACKUP LOCK

MariaDB starting with [10.4.2](#)

The BACKUP LOCK command was introduced in [MariaDB 10.4.2](#).

### Contents

1. [Syntax](#)
2. [Usage in a Backup Tool](#)
3. [Privileges](#)
4. [Notes](#)
5. [Implementation](#)
6. [See Also](#)

BACKUP LOCK blocks a table from DDL statements. This is mainly intended to be used by tools like [mariabackup](#) that need to ensure there are no DDLs on a table while the table files are opened. For example, for an Aria table that stores data in 3 files with extensions .frm, .MAI and .MAD. Normal read/write operations can continue as normal.

### Syntax

To lock a table:

```
BACKUP LOCK table_name
```

To unlock a table:

```
BACKUP UNLOCK
```

### Usage in a Backup Tool

```
BACKUP LOCK [database.]table_name;
- Open all files related to a table (for example, t.frm, t.MAI and t.MYD)
BACKUP UNLOCK;
- Copy data
- Close files
```

This ensures that all files are from the same generation, that is created at the same time by the MariaDB server. This works, because the open files will point to the original table files which will not be affected if there is any ALTER TABLE while copying the files.

### Privileges

BACKUP LOCK requires the

```
RELOAD
```

privilege.

### Notes

- The idea is that the  

```
BACKUP LOCK
```

should be held for as short a time as possible by the backup tool. The time to take an uncontested lock is very short! One can easily do 50,000 locks/unlocks per second on low end hardware.
- One should use different connections for [BACKUP STAGE](#) commands and  

```
BACKUP LOCK
```

### Implementation

- Internally, BACKUP LOCK is implemented by taking an  

```
MDL_SHARED_HIGH_PRIO
```

MDL lock on the table object, which protects the table from any DDL operations.

### See Also

- [BACKUP STAGE](#)
- [MDEV-17309](#) - BACKUP LOCK: DDL locking of tables during backup

## 1.1.2.3.3 Mariabackup and BACKUP STAGE Commands

MariaDB starting with [10.4.1](#)

The

[BACKUP STAGE](#)

commands were introduced in [MariaDB 10.4.1](#).

### Contents

1. [Mariabackup and BACKUP STAGE Commands in MariaDB Community Server](#)
  1. [Tasks Performed Prior to BACKUP STAGE in MariaDB Community Server](#)
  2. [BACKUP STAGE START in MariaDB Community Server](#)
  3. [BACKUP STAGE FLUSH in MariaDB Community Server](#)
  4. [BACKUP STAGE BLOCK\\_DDL in MariaDB Community Server](#)
  5. [BACKUP STAGE BLOCK\\_COMMIT in MariaDB Community Server](#)
  6. [BACKUP STAGE END in MariaDB Community Server](#)
2. [Mariabackup and BACKUP STAGE Commands in MariaDB Enterprise Server](#)
  1. [BACKUP STAGE START in MariaDB Enterprise Server](#)
  2. [BACKUP STAGE FLUSH in MariaDB Enterprise Server](#)
  3. [BACKUP STAGE BLOCK\\_DDL in MariaDB Enterprise Server](#)
  4. [BACKUP STAGE BLOCK\\_COMMIT in MariaDB Enterprise Server](#)
  5. [BACKUP STAGE END in MariaDB Enterprise Server](#)

The

[BACKUP STAGE](#)

commands are a set of commands to make it possible to make an efficient external backup tool. How Mariabackup uses these commands depends on whether you are using the version that is bundled with MariaDB Community Server or the version that is bundled with [MariaDB Enterprise Server](#).

## Mariabackup and BACKUP STAGE Commands in MariaDB Community Server

MariaDB starting with [10.4.1](#)

In MariaDB Community Server, Mariabackup first supported

[BACKUP STAGE](#)

commands in [MariaDB 10.4.1](#).

In [MariaDB 10.3](#) and before, the

[BACKUP STAGE](#)

commands are **not** supported, so Mariabackup executes the

## FLUSH TABLES WITH READ LOCK

command to lock the database. When the backup is complete, it executes the

## UNLOCK TABLES

command to unlock the database.

In [MariaDB 10.4](#) and later, the

## BACKUP STAGE

commands are supported. However, the version of Mariabackup that is bundled with MariaDB Community Server does not yet use the

## BACKUP STAGE

commands in the most efficient way. Mariabackup simply executes the following

## BACKUP STAGE

commands to lock the database:

```
BACKUP STAGE START;  
BACKUP STAGE BLOCK_COMMIT;
```

When the backup is complete, it executes the following

## BACKUP STAGE

command to unlock the database:

```
BACKUP STAGE END;
```

If you would like to use a version of Mariabackup that uses the

## BACKUP STAGE

commands in the most efficient way, then your best option is to use [MariaDB Enterprise Backup](#) that is bundled with [MariaDB Enterprise Server](#).

## Tasks Performed Prior to

### BACKUP STAGE in MariaDB Community Server

- Copy some transactional tables.
  - [InnoDB](#) (i.e.
    - ibdataN
    - and file extensions
    - .ibd
    - and
    - .isl
    - )
- Copy the tail of some transaction logs.
  - The tail of the [InnoDB redo log](#) (i.e.
    - ib\_logfileN
    - files) will be copied for [InnoDB](#) tables.

### BACKUP STAGE START in MariaDB Community Server

Mariabackup from MariaDB Community Server does not currently perform any tasks in the

START  
stage.

## BACKUP STAGE FLUSH in MariaDB Community Server

Mariabackup from MariaDB Community Server does not currently perform any tasks in the FLUSH stage.

## BACKUP STAGE BLOCK\_DDL in MariaDB Community Server

Mariabackup from MariaDB Community Server does not currently perform any tasks in the BLOCK\_DDL stage.

## BACKUP STAGE BLOCK\_COMMIT in MariaDB Community Server

Mariabackup from MariaDB Community Server performs the following tasks in the BLOCK\_COMMIT stage:

- Copy other files.
  - i.e. file extensions
    - .frm
    - ,
    - .isl
    - ,
    - .TRG
    - ,
    - .TRN
    - ,
    - .opt
    - ,
    - .par
- Copy some transactional tables.
  - [Aria](#) (i.e.
    - aria\_log\_control
    - and file extensions
    - .MAD
    - and
    - .MAI
    - )
- Copy the non-transactional tables.
  - [MyISAM](#) (i.e. file extensions
    - .MYD
    - and
    - .MYI
    - )
  - [MERGE](#) (i.e. file extensions
    - .MRG
    - )
  - [ARCHIVE](#) (i.e. file extensions
    - .ARM
    - and
    - .ARZ
    - )
  - [CSV](#) (i.e. file extensions
    - .CSM
    - and
    - .CSV
    - )
- Create a [MyRocks](#) checkpoint using the `rocksdb_create_checkpoint` system variable.
- Copy the tail of some transaction logs.

- The tail of the [InnoDB redo log](#) (i.e. `ib_logfileN` files) will be copied for [InnoDB](#) tables.
- Save the [binary log](#) position to

`xtrabackup_binlog_info`

- Save the [Galera Cluster](#) state information to

`xtrabackup_galera_info`

## BACKUP STAGE END in MariaDB Community Server

Mariabackup from MariaDB Community Server performs the following tasks in the

END  
stage:

- Copy the [MyRocks](#) checkpoint into the backup.

## Mariabackup and BACKUP STAGE Commands in MariaDB Enterprise Server

MariaDB starting with [10.2.25](#)

[MariaDB Enterprise Backup](#) first supported

`BACKUP STAGE`

commands in [MariaDB Enterprise Server 10.4.6-1](#) , [MariaDB Enterprise Server 10.3.16-1](#) , and [MariaDB Enterprise Server 10.2.25-1](#) .

The following sections describe how the [MariaDB Enterprise Backup](#) version of Mariabackup that is bundled with [MariaDB Enterprise Server](#) uses each

`BACKUP STAGE`

command in an efficient way.

## BACKUP STAGE START in MariaDB Enterprise Server

Mariabackup from MariaDB Enterprise Server performs the following tasks in the

START  
stage:

- Copy all transactional tables.
  - [InnoDB](#) (i.e.
    - `ibdataN`
    - and file extensions
    - `.ibd`
    - and
    - `.isl`
    - )
  - [Aria](#) (i.e.
    - `aria_log_control`
    - and file extensions
    - `.MAD`
    - and
    - `.MAI`
    - )
- Copy the tail of all transaction logs.
  - The tail of the [InnoDB redo log](#) (i.e. `ib_logfileN`)

- files) will be copied for [InnoDB](#) tables.
- The tail of the Aria redo log (i.e.  
aria\_log.N  
files) will be copied for [Aria](#) tables.

## BACKUP STAGE FLUSH in MariaDB Enterprise Server

Mariabackup from MariaDB Enterprise Server performs the following tasks in the

FLUSH

stage:

- Copy all non-transactional tables that are not in use. This list of used tables is found with

[SHOW OPEN TABLES](#)

- [MyISAM](#) (i.e. file extensions  
.MYD  
and  
.MYI  
)
- [MERGE](#) (i.e. file extensions  
.MRG  
)
- [ARCHIVE](#) (i.e. file extensions  
.ARM  
and  
.ARZ  
)
- [CSV](#) (i.e. file extensions  
.CSM  
and  
.CSV  
)
- Copy the tail of all transaction logs.
  - The tail of the [InnoDB redo log](#) (i.e.  
ib\_logfileN  
files) will be copied for [InnoDB](#) tables.
  - The tail of the Aria redo log (i.e.  
aria\_log.N  
files) will be copied for [Aria](#) tables.

## BACKUP STAGE BLOCK\_DDL in MariaDB Enterprise Server

Mariabackup from MariaDB Enterprise Server performs the following tasks in the

BLOCK\_DDL

stage:

- Copy other files.
  - i.e. file extensions  
.frm  
,  
.isl  
,  
.TRG  
,  
.TRN  
,  
.opt  
,  
.par
- Copy the non-transactional tables that were in use during  
BACKUP STAGE FLUSH
  - [MyISAM](#) (i.e. file extensions  
.MYD

- **MERGE** (i.e. file extensions
  - .MRG
  - )
- **ARCHIVE** (i.e. file extensions
  - .ARM
  - and
  - .ARZ
  - )
- **CSV** (i.e. file extensions
  - .CSM
  - and
  - .CSV
  - )

- Check

ddl.log  
for DDL executed before the  
BLOCK\_DDL  
stage.

- The file names of newly created tables can be read from

ddl.log

- The file names of dropped tables can also be read from

ddl.log

- The file names of renamed tables can also be read from

ddl.log

, so the files can be renamed instead of re-copying them.

- Copy changes to system log tables.

- 

## mysql.general\_log

○

## mysql.slow\_log

- This is easy as these are append only.
  - Copy the tail of all transaction logs.

- The tail of the InnoDB redo log (i.e.

ib logfileN

files) will be copied for InnoDB tables.

- The tail of the Aria redo log (i.e.

aria\_log.N

files) will be copied for [Aria](#) tables.

## BACKUP STAGE BLOCK\_COMMIT in MariaDB Enterprise Server

Mariabackup from MariaDB Enterprise Server performs the following tasks in the `BLOCK_COMMIT` stage:

- Create a `MyRocks` checkpoint using the

`rocksdb create_checkpoint`

system variable.

- Copy changes to system log tables.

8

## mysql.general log

8

`mysql.slow_log`

- This is easy as these are append only.
- Copy changes to statistics tables.
  -

`mysql.table_stats`

◦

`mysql.column_stats`

◦

`mysql.index_stats`

- Copy the tail of all transaction logs.

- The tail of the [InnoDB redo log](#) (i.e. `ib_logfileN` files) will be copied for [InnoDB](#) tables.
- The tail of the [Aria redo log](#) (i.e. `aria_log.N` files) will be copied for [Aria](#) tables.

- Save the [binary log](#) position to

`xtrabackup_binlog_info`

- Save the [Galera Cluster](#) state information to

`xtrabackup_galera_info`

## BACKUP STAGE END in MariaDB Enterprise Server

Mariabackup from MariaDB Enterprise Server performs the following tasks in the

`END`  
stage:

- Copy the [MyRocks](#) checkpoint into the backup.

### 1.1.2.3.4 Storage Snapshots and BACKUP STAGE Commands

MariaDB starting with [10.4.1](#)

The

`BACKUP STAGE`  
commands were introduced in [MariaDB 10.4.1](#).

## Contents

1. [Generic Backup Process with Storage Snapshots](#)

The

`BACKUP STAGE`

commands are a set of commands to make it possible to make an efficient external backup tool. These commands could even be used by tools that perform backups by taking a snapshot of a file system, SAN, or some other kind of storage device.

# Generic Backup Process with Storage Snapshots

A tool that backs up MariaDB by taking a snapshot of a file system, SAN, or some other kind of storage device could use each

```
BACKUP STAGE  
command in the following way:
```

- First, execute the following:

```
BACKUP STAGE START  
BACKUP STAGE BLOCK_COMMIT
```

- Then, take the snapshot.

- Then, execute the following:

```
BACKUP STAGE END
```

The above ensures that all non-transactional tables are properly flushed to disk before the snapshot is done. Using

```
BACKUP STAGE  
commands is also more efficient than using the
```

```
FLUSH TABLES WITH READ LOCK
```

command as the above set of commands will not block or be blocked by write operations to transactional tables.

Note that when the backup is completed, one should delete all files with the "#sql" prefix, as these are files used by concurrent running

```
ALTER TABLE  
. Note that InnoDB will on server restart automatically delete any tables with the "#sql" prefix.
```

## 1.1.2.4 FLUSH Commands

### 1.1.2.4.1 FLUSH

#### Syntax

```
FLUSH [NO_WRITE_TO_BINLOG | LOCAL]  
flush_option [, flush_option] ...
```

or when flushing tables:

```
FLUSH [NO_WRITE_TO_BINLOG | LOCAL] TABLES [table_list] [table_flush_option]
```

## Contents

1. Syntax
2. Description
3. FLUSH STATUS
4. Compatibility with MySQL
  1. Global Status Variables that Support FLUSH STATUS
5. The different usage of FLUSH TABLES
  1. The purpose of FLUSH TABLES
  2. The purpose of FLUSH TABLES WITH READ LOCK
  3. The purpose of FLUSH TABLES table\_list
  4. The purpose of FLUSH TABLES table\_list WITH READ LOCK
6. Implementation of FLUSH TABLES commands in MariaDB 10.4.8 and above
  1. Implementation of FLUSH TABLES
  2. Implementation of FLUSH TABLES WITH READ LOCK
  3. Implementation of FLUSH TABLES table\_list
  4. Implementation of FLUSH TABLES table\_list FOR EXPORT
7. FLUSH SSL
8. Reducing Memory Usage

where

table\_list  
is a list of tables separated by  
,

(comma).

## Description

The

FLUSH  
statement clears or reloads various internal caches used by MariaDB. To execute  
FLUSH  
, you must have the  
RELOAD  
privilege. See [GRANT](#).

The

RESET  
statement is similar to  
FLUSH  
. See [RESET](#).

You cannot issue a FLUSH statement from within a [stored function](#) or a [trigger](#). Doing so within a stored procedure is permitted, as long as it is not called by a stored function or trigger. See [Stored Routine Limitations](#), [Stored Function Limitations](#) and [Trigger Limitations](#).

If a listed table is a view, an error like the following will be produced:

```
ERROR 1347 (HY000): 'test.v' is not BASE TABLE
```

By default,

FLUSH  
statements are written to the [binary log](#) and will be [replicated](#). The  
NO\_WRITE\_TO\_BINLOG  
keyword (  
LOCAL  
is an alias) will ensure the statement is not written to the binary log.

The different flush options are:

Option	Description
CHANGED_PAGE_BITMAPS	Internal command used for backup purposes. See the <a href="#">Information Schema CHANGED_PAGE_BITMAPS Table</a> .

CLIENT_STATISTICS	Reset client statistics (see <a href="#">SHOW CLIENT_STATISTICS</a> ).
DES_KEY_FILE	Reloads the DES key file (Specified with the <a>--des-key-file</a> startup option ).
HOSTS	Flush the hostname cache (used for converting ip to host names and for unblocking blocked hosts. See <a>max_connect_errors</a> )
INDEX_STATISTICS	Reset index statistics (see <a>SHOW INDEX_STATISTICS</a> ).
[ERROR   ENGINE   GENERAL   SLOW   BINARY   RELAY] LOGS	<p>Close and reopen the specified log type, or all log types if none are specified.</p> <p><code>FLUSH RELAY LOGS [connection-name]</code>  can be used to flush the relay logs for a specific connection. Only one connection can be specified per <code>FLUSH</code> command. See <a>Multi-source replication</a> .</p> <p><code>FLUSH ENGINE LOGS</code>  will delete all unneeded Aria redo logs. Since MariaDB 10.1.30 and MariaDB 10.2.11 , <code>FLUSH BINARY LOGS DELETE_DOMAIN_ID=(list-of-domains)</code>  can be used to discard obsolete GTID domains from the server's <a>binary log</a> state. In order for this to be successful, no event group from the listed GTID domains can be present in existing <a>binary log</a> files. If some still exist, then they must be purged prior to executing this command. If the command completes successfully, then it also rotates the <a>binary log</a> .</p>
MASTER	Deprecated option, use <a>RESET MASTER</a> instead.
PRIVILEGES	Reload all privileges from the privilege tables in the mysql database. If the server is started with <code>--skip-grant-table</code> option, this will activate the privilege tables again.
QUERY CACHE	Defragment the <a>query cache</a> to better utilize its memory. If you want to reset the query cache, you can do it with <a>RESET QUERY CACHE</a> .
QUERY_RESPONSE_TIME	See the <a>QUERY_RESPONSE_TIME</a> plugin.
SLAVE	Deprecated option, use <a>RESET REPLICA</a> or <a>RESET SLAVE</a> instead.
SSL	Used to dynamically reinitialize the server's <a>TLS</a> context by reloading the files defined by several <a>TLS system variables</a> . See <a>FLUSH SSL</a> for more information. This command was first added in <a>MariaDB 10.4.1</a> .
STATUS	Resets all <a>server status variables</a> that can be reset to 0. Not all global status variables support this, so not all global values are reset. See <a>FLUSH STATUS</a> for more information.
TABLE	Close tables given as options or all open tables if no table list was used. From <a>MariaDB 10.4.1</a> , using without any table list will only close tables not in use, and tables not locked by the FLUSH TABLES connection. If there are no locked tables, FLUSH TABLES will be instant and will not cause any waits, as it no longer waits for tables in use. When a table list is provided, from <a>MariaDB 10.4.1</a> , the server will wait for the end of any transactions that are using the tables. Previously, FLUSH TABLES only waited for the statements to complete.
TABLES	Same as <code>FLUSH TABLE</code> -
TABLES ... FOR EXPORT	For InnoDB tables, flushes table changes to disk to permit binary table copies while the server is running. Introduced in <a>MariaDB 10.0.8</a> . See <a>FLUSH TABLES ... FOR EXPORT</a> for more.

TABLES WITH READ LOCK	Closes all open tables. New tables are only allowed to be opened with read locks until an <a href="#">UNLOCK TABLES</a> is given.
TABLES WITH READ LOCK AND DISABLE CHECKPOINT	As TABLES WITH READ LOCK but also disable all checkpoint writes by transactional table engines. This is useful when doing a disk snapshot of all tables.
TABLE_STATISTICS	Reset table statistics (see <a href="#">SHOW TABLE_STATISTICS</a> ).
USER_RESOURCES	Resets all per hour <a href="#">user resources</a> . This enables clients that have exhausted their resources to connect again.
USER_STATISTICS	Reset user statistics (see <a href="#">SHOW USER_STATISTICS</a> ).

You can also use the [mysqladmin](#) client to flush things. Use

```
mysqladmin --help
to examine what flush commands it supports.
```

## FLUSH STATUS

[Server status variables](#) can be reset by executing the following:

```
FLUSH STATUS;
```

## Compatibility with MySQL

MariaDB starting with [10.7.0](#)

The

```
FOR CHANNEL
```

keyword was added for MySQL compatibility. This is identical as using the channel\_name directly after the FLUSH command

For example, one can now use:

```
FLUSH RELAY LOGS 'connection_name';
FLUSH RELAY LOGS FOR CHANNEL 'connection_name';
```

## Global Status Variables that Support FLUSH STATUS

Not all global status variables support being reset by

```
FLUSH STATUS
. Currently, the following status variables are reset by
FLUSH STATUS
:
```

- [Aborted\\_clients](#)
- [Aborted\\_connects](#)
- [Binlog\\_cache\\_disk\\_use](#)
- [Binlog\\_cache\\_use](#)
- [Binlog\\_stmt\\_cache\\_disk\\_use](#)

- Binlog\_stmt\_cache\_use
- Connection\_errors\_accept
- Connection\_errors\_internal
- Connection\_errors\_max\_connections
- Connection\_errors\_peer\_address
- Connection\_errors\_select
- Connection\_errors\_tcpwrap
- Created\_tmp\_files
- Delayed\_errors
- Delayed\_writes
- Feature\_check\_constraint
- Feature\_delay\_key\_write
- Max\_used\_connections
- Opened\_plugin\_libraries
- Performance\_schema\_accounts\_lost
- Performance\_schema\_cond\_instances\_lost
- Performance\_schema\_digest\_lost
- Performance\_schema\_file\_handles\_lost
- Performance\_schema\_file\_instances\_lost
- Performance\_schema\_hosts\_lost
- Performance\_schema\_locker\_lost
- Performance\_schema\_mutex\_instances\_lost
- Performance\_schema\_rwlock\_instances\_lost
- Performance\_schema\_session\_connect\_attrs\_lost
- Performance\_schema\_socket\_instances\_lost
- Performance\_schema\_stage\_classes\_lost
- Performance\_schema\_statement\_classes\_lost
- Performance\_schema\_table\_handles\_lost
- Performance\_schema\_table\_instances\_lost
- Performance\_schema\_thread\_instances\_lost
- Performance\_schema\_users\_lost
- Qcache\_hits
- Qcache\_inserts
- Qcache\_lowmem\_prunes
- Qcache\_not\_cached
- Rpl\_semi\_sync\_master\_no\_times
- Rpl\_semi\_sync\_master\_no\_tx
- Rpl\_semi\_sync\_master\_timefunc\_failures
- Rpl\_semi\_sync\_master\_wait\_pos\_backtraverse
- Rpl\_semi\_sync\_master\_yes\_tx
- Rpl\_transactions\_multi\_engine
- Server\_audit\_writes\_failed
- Slave\_retried\_transactions
- Slow\_launch\_threads
- Ssl\_accept\_renegotiates
- Ssl\_accepts
- Ssl\_callback\_cache\_hits
- Ssl\_client\_connects
- Ssl\_connect\_renegotiates
- Ssl\_ctx\_verify\_depth
- Ssl\_ctx\_verify\_mode
- Ssl\_finished\_accepts
- Ssl\_finished\_connects
- Ssl\_session\_cache\_hits
- Ssl\_session\_cache\_misses
- Ssl\_session\_cache\_overflows
- Ssl\_session\_cache\_size
- Ssl\_session\_cache\_timeouts
- Ssl\_sessions\_reused
- Ssl\_used\_session\_cache\_entries
- Subquery\_cache\_hit
- Subquery\_cache\_miss
- Table\_locks\_immediate
- Table\_locks\_waited
- Tc\_log\_max\_pages\_used
- Tc\_log\_page\_waits
- Transactions\_gtid\_foreign\_engine
- Transactions\_multi\_engine

## The different usage of

# FLUSH TABLES

## The purpose of FLUSH TABLES

The purpose of

FLUSH TABLES

is to clean up the open table cache and table definition cache from not in use tables. This frees up memory and file descriptors. Normally this is not needed as the caches works on a FIFO bases, but can be useful if the server seams to use up to much memory for some reason.

## The purpose of FLUSH TABLES WITH READ LOCK

FLUSH TABLES WITH READ LOCK

is useful if you want to take a backup of some tables. When

FLUSH TABLES WITH READ LOCK

returns, all write access to tables are blocked and all tables are marked as 'properly closed' on disk. The tables can still be used for read operations.

## The purpose of FLUSH TABLES table\_list

FLUSH TABLES

table\_list is useful if you want to copy a table object/files to or from the server. This command puts a lock that stops new users of the table and will wait until everyone has stopped using the table. The table is then removed from the table definition and table cache.

Note that it's up to the user to ensure that no one is accessing the table between

FLUSH TABLES

and the table is copied to or from the server. This can be secured by using [LOCK TABLES](#).

If there are any tables locked by the connection that is using

FLUSH TABLES

all the locked tables will be closed as part of the flush and reopened and relocked before

FLUSH TABLES

returns. This allows one to copy the table after

FLUSH TABLES

returns without having any writes on the table. For now this works works with most tables, except InnoDB as InnoDB may do background purges on the table even while it's write locked.

## The purpose of FLUSH TABLES table\_list WITH READ LOCK

FLUSH TABLES table\_list WITH READ LOCK

should work as

FLUSH TABLES WITH READ LOCK

, but only those tables that are listed will be properly closed. However in practice this works exactly like

FLUSH TABLES WITH READ LOCK

as the

FLUSH

command has anyway to wait for all WRITE operations to end because we are depending on a global read lock for this code. In the future we should consider fixing this to instead use meta data locks.

## Implementation of FLUSH TABLES commands in MariaDB 10.4.8 and above

## Implementation of FLUSH TABLES

- Free memory and file descriptors not in use

## Implementation of FLUSH TABLES WITH READ LOCK

- Lock all tables read only for simple old style backup.
- All background writes are suspended and tables are marked as closed.
- No statement requiring table changes are allowed for any user until  
UNLOCK TABLES

Instead of using

FLUSH TABLE WITH READ LOCK  
one should in most cases instead use [BACKUP STAGE BLOCK\\_COMMIT](#).

## Implementation of FLUSH TABLES table\_list

- Free memory and file descriptors for tables not in use from table list.
- Lock given tables as read only.
- Wait until all translations has ended that uses any of the given tables.
- Wait until all background writes are suspended and tables are marked as closed.

## Implementation of FLUSH TABLES table\_list FOR EXPORT

- Free memory and file descriptors for tables not in use from table list
  - Lock given tables as read.
  - Wait until all background writes are suspended and tables are marked as closed.
  - Check that all tables supports  
FOR EXPORT
- No changes to these tables allowed until  
UNLOCK TABLES

This is basically the same behavior as in old MariaDB version if one first lock the tables, then do

FLUSH TABLES  
. The tables will be copyable until  
UNLOCK TABLES

## FLUSH SSL

MariaDB starting with [10.4](#)

The

FLUSH SSL  
command was first added in [MariaDB 10.4](#).

In [MariaDB 10.4](#) and later, the

FLUSH SSL  
command can be used to dynamically reinitialize the server's [TLS](#) context. This is most useful if you need to replace a certificate that is about to expire without restarting the server.

This operation is performed by reloading the files defined by the following [TLS system variables](#):

-

`ssl_cert`

`ssl_key`

`ssl_ca`

`ssl_capath`

`ssl_crl`

`ssl_crlpath`

These [TLS system variables](#) are not dynamic, so their values can **not** be changed without restarting the server.

If you want to dynamically reinitialize the server's [TLS](#) context, then you need to change the certificate and key files at the relevant paths defined by these [TLS system variables](#), without actually changing the values of the variables. See [MDEV-19341](#) for more information.

## Reducing Memory Usage

To flush some of the global caches that take up memory, you could execute the following command:

```
FLUSH LOCAL HOSTS,  
        QUERY CACHE,  
        TABLE_STATISTICS,  
        INDEX_STATISTICS,  
        USER_STATISTICS;
```

### 1.1.2.4.2 FLUSH QUERY CACHE

#### Description

You can defragment [the query cache](#) to better utilize its memory with the

```
FLUSH QUERY CACHE  
statement. The statement does not remove any queries from the cache.
```

The

```
RESET QUERY CACHE
```

statement removes all query results from the query cache. The

```
FLUSH TABLES
```

statement also does this.

### 1.1.2.4.3 FLUSH TABLES FOR EXPORT

#### Syntax

```
FLUSH TABLES table_name [, table_name] FOR EXPORT
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [See Also](#)

## Description

`FLUSH TABLES ... FOR EXPORT`

flushes changes to the specified tables to disk so that binary copies can be made while the server is still running. This works for [Archive](#) , [Aria](#) , [CSV](#) , [InnoDB](#) , [MyISAM](#) , [MERGE](#) , and [XtraDB](#) tables.

The table is read locked until one has issued [UNLOCK TABLES](#) .

If a storage engine does not support

`FLUSH TABLES FOR EXPORT`

, a 1031 error ( [SQLSTATE](#) 'HY000') is produced.

If

`FLUSH TABLES ... FOR EXPORT`

is in effect in the session, the following statements will produce an error if attempted:

- `FLUSH TABLES WITH READ LOCK`
- `FLUSH TABLES ... WITH READ LOCK`
- `FLUSH TABLES ... FOR EXPORT`
- Any statement trying to update any table

If any of the following statements is in effect in the session, attempting

`FLUSH TABLES ... FOR EXPORT`

will produce an error.

- `FLUSH TABLES ... WITH READ LOCK`
- `FLUSH TABLES ... FOR EXPORT`
- `LOCK TABLES ... READ`
- `LOCK TABLES ... WRITE`

`FLUSH FOR EXPORT`

is not written to the [binary log](#) .

This statement requires the [RELOAD](#) and the [LOCK TABLES](#) privileges.

If one of the specified tables cannot be locked, none of the tables will be locked.

If a table does not exist, an error like the following will be produced:

```
ERROR 1146 (42S02): Table 'test.xxx' doesn't exist
```

If a table is a view, an error like the following will be produced:

```
ERROR 1347 (HY000): 'test.v' is not BASE TABLE
```

## Example

```
FLUSH TABLES test.t1 FOR EXPORT;
# Copy files related to the table (see below)
UNLOCK TABLES;
```

## See Also

- [Copying Tables Between Different MariaDB Databases and MariaDB Servers](#)
- [Copying Transportable InnoDB Tablespaces](#)
- [myisampack](#) - Compressing the MyISAM data file for easier distribution.
- [aria\\_pack](#) - Compressing the Aria data file for easier distribution

## 1.1.2.5 Replication Commands

### 1.1.2.5.1 CHANGE MASTER TO

The terms *master* and *slave* have historically been used in replication, but the terms *primary* and *replica* are now preferred. The old terms are used still used in parts of the documentation, and in MariaDB commands, although MariaDB 10.5 has begun the process of renaming. The documentation process is ongoing. See [MDEV-18777](#) to follow progress on this effort.

## Syntax

```
CHANGE MASTER ['connection_name'] TO master_def [, master_def] ... [FOR CHANNEL 'channel_name']

master_def:
  MASTER_BIND = 'interface_name'
  | MASTER_HOST = 'host_name'
  | MASTER_USER = 'user_name'
  | MASTER_PASSWORD = 'password'
  | MASTER_PORT = port_num
  | MASTER_CONNECT_RETRY = interval
  | MASTER_HEARTBEAT_PERIOD = interval
  | MASTER_LOG_FILE = 'master_log_name'
  | MASTER_LOG_POS = master_log_pos
  | RELAY_LOG_FILE = 'relay_log_name'
  | RELAY_LOG_POS = relay_log_pos
  | MASTER_DELAY = interval
  | MASTER_SSL = {0|1}
  | MASTER_SSL_CA = 'ca_file_name'
  | MASTER_SSL_CAPATH = 'ca_directory_name'
  | MASTER_SSL_CERT = 'cert_file_name'
  | MASTER_SSL_CRL = 'crl_file_name'
  | MASTER_SSL_CRLPATH = 'crl_directory_name'
  | MASTER_SSL_KEY = 'key_file_name'
  | MASTER_SSL_CIPHER = 'cipher_list'
  | MASTER_SSL_VERIFY_SERVER_CERT = {0|1}
  | MASTER_USE_GTID = {current_pos|slave_pos|no}
  | IGNORE_SERVER_IDS = (server_id_list)
  | DO_DOMAIN_IDS = ([N,...])
  | IGNORE_DOMAIN_IDS = ([N,...])
```

## Contents

1. Syntax
2. Description
3. Multi-Source Replication
  1. `default_master_connection`
  2. `connection_name`
4. Options
  1. Connection Options
    1. `MASTER_USER`
    2. `MASTER_PASSWORD`
    3. `MASTER_HOST`
    4. `MASTER_PORT`
    5. `MASTER_CONNECT_RETRY`
    6. `MASTER_BIND`
    7. `MASTER_HEARTBEAT_PERIOD`
  2. TLS Options
    1. `MASTER_SSL`
    2. `MASTER_SSL_CA`
    3. `MASTER_SSL_CAPATH`
    4. `MASTER_SSL_CERT`
    5. `MASTER_SSL_CRL`
    6. `MASTER_SSL_CRLPATH`
    7. `MASTER_SSL_KEY`
    8. `MASTER_SSL_CIPHER`
    9. `MASTER_SSL_VERIFY_SERVER_CERT`
  3. Binary Log Options
    1. `MASTER_LOG_FILE`
    2. `MASTER_LOG_POS`
  4. Relay Log Options
    1. `RELAY_LOG_FILE`
    2. `RELAY_LOG_POS`
  5. GTID Options
    1. `MASTER_USE_GTID`
  6. Replication Filter Options
    1. `IGNORE_SERVER_IDS`
    2. `DO_DOMAIN_IDS`
    3. `IGNORE_DOMAIN_IDS`
  7. Delayed Replication Options
    1. `MASTER_DELAY`
  5. Changing Option Values
  6. Option Persistence
  7. GTID Persistence
  8. Creating a Slave from a Backup
  9. Example
  10. See Also

## Description

The

```
CHANGE MASTER  
statement sets the options that a replica uses to connect to and replicate from a primary.
```

MariaDB starting with 10.7.0

The

```
FOR CHANNEL  
keyword was added for MySQL compatibility. This is identical to using the channel_name directly after  
CHANGE MASTER
```

## Multi-Source Replication

If you are using `multi-source replication`, then you need to specify a connection name when you execute

```
CHANGE MASTER  
. There are two ways to do this:

- Setting the default_master_connection system variable prior to executing  
CHANGE MASTER
- Setting the

```

```
connection_name  
parameter when executing  
CHANGE MASTER
```

## default\_master\_connection

```
SET default_master_connection = 'gandalf';  
STOP SLAVE;  
CHANGE MASTER TO  
    MASTER_PASSWORD='new3cret';  
START SLAVE;
```

## connection\_name

```
STOP SLAVE 'gandalf';  
CHANGE MASTER 'gandalf' TO  
    MASTER_PASSWORD='new3cret';  
START SLAVE 'gandalf';
```

# Options

## Connection Options

### MASTER\_USER

The

```
MASTER_USER  
option for  
CHANGE MASTER  
defines the user account that the replica will use to connect to the primary.
```

This user account will need the [REPLICATION SLAVE](#) privilege (or, from [MariaDB 10.5.1](#), the [REPLICATION REPLICA](#) on the primary).

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    MASTER_USER='rep1',  
    MASTER_PASSWORD='new3cret';  
START SLAVE;
```

The maximum length of the

```
MASTER_USER  
string is 96 characters until MariaDB 10.5, and 128 characters from MariaDB 10.6.
```

### MASTER\_PASSWORD

The

```
MASTER_USER  
option for  
CHANGE MASTER  
defines the password that the replica will use to connect to the primary as the user account defined by the MASTER\_USER option.
```

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    MASTER_PASSWORD='new3cret';  
START SLAVE;
```

The maximum length of the  
MASTER\_PASSWORD  
string is 32 characters.

#### MASTER\_HOST

The  
MASTER\_HOST  
option for  
CHANGE MASTER  
defines the hostname or IP address of the primary .

If you set the value of the

MASTER\_HOST  
option to the empty string, then that is not the same as not setting the option's value at all. If you set the value of the  
MASTER\_HOST  
option to the empty string, then the  
CHANGE MASTER  
command will fail with an error. In MariaDB 5.3 and before, if you set the value of the  
MASTER\_HOST  
option to the empty string, then the  
CHANGE MASTER  
command would succeed, but the subsequent START SLAVE command would fail.

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
MASTER_HOST='dbserver1.example.com',  
MASTER_USER='rep1',  
MASTER_PASSWORD='new3cret',  
MASTER_USE_GTID=slave_pos;  
START SLAVE;
```

If you set the value of the

MASTER\_HOST  
option in a  
CHANGE MASTER  
command, then the replica assumes that the primary is different from before, even if you set the value of this option to the same value it had previously. In this scenario, the replica will consider the old values for the primary's binary log file name and position to be invalid for the new primary. As a side effect, if you do not explicitly set the values of the MASTER\_LOG\_FILE and MASTER\_LOG\_POS options in the statement, then the statement will be implicitly appended with  
MASTER\_LOG\_FILE=''  
and  
MASTER\_LOG\_POS=4  
. However, if you enable GTID mode for replication by setting the MASTER\_USE\_GTID option to some value other than  
no  
in the statement, then these values will effectively be ignored anyway.

Replicas cannot connect to primaries using Unix socket files or Windows named pipes. The replica must connect to the primary using TCP/IP.

The maximum length of the  
MASTER\_HOST  
string is 60 characters until MariaDB 10.5 , and 255 characters from MariaDB 10.6 .

#### MASTER\_PORT

The  
MASTER\_PORT  
option for  
CHANGE MASTER  
defines the TCP/IP port of the primary .

For example:

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_HOST='dbserver1.example.com',
  MASTER_PORT=3307,
  MASTER_USER='rep1',
  MASTER_PASSWORD='new3cret',
  MASTER_USE_GTID=slave_pos;
START SLAVE;
```

If you set the value of the

MASTER\_PORT  
option in a  
CHANGE MASTER

command, then the replica assumes that the primary is different from before, even if you set the value of this option to the same value it had previously. In this scenario, the replica will consider the old values for the primary's [binary log](#) file name and position to be invalid for the new primary. As a side effect, if you do not explicitly set the values of the [MASTER\\_LOG\\_FILE](#) and [MASTER\\_LOG\\_POS](#) options in the statement, then the statement will be implicitly appended with

MASTER\_LOG\_FILE=''  
and  
MASTER\_LOG\_POS=4  
. However, if you enable [GTID](#) mode for replication by setting the [MASTER\\_USE\\_GTID](#) option to some value other than no  
in the statement, then these values will effectively be ignored anyway.

Replicas cannot connect to primaries using Unix socket files or Windows named pipes. The replica must connect to the primary using TCP/IP.

#### MASTER\_CONNECT\_RETRY

The

MASTER\_CONNECT\_RETRY  
option for  
CHANGE MASTER  
defines how many seconds that the replica will wait between connection retries. The default is  
60

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_CONNECT_RETRY=20;
START SLAVE;
```

The number of connection attempts is limited by the [master\\_retry\\_count](#) option. It can be set either on the command-line or in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
master_retry_count=4294967295
```

#### MASTER\_BIND

The

MASTER\_BIND  
option for  
CHANGE MASTER

is only supported by MySQL 5.6.2 and later and by MySQL NDB Cluster 7.3.1 and later. This option is not supported by MariaDB. See [MDEV-19248](#) for more information.

The

MASTER\_BIND  
option for  
CHANGE MASTER

can be used on replicas that have multiple network interfaces to choose which network interface the replica will use to connect to the primary.

#### MASTER\_HEARTBEAT\_PERIOD

The

MASTER\_HEARTBEAT\_PERIOD

option for

CHANGE MASTER

can be used to set the interval in seconds between replication heartbeats. Whenever the primary's [binary log](#) is updated with an event, the waiting period for the next heartbeat is reset.

This option's *interval* argument has the following characteristics:

- It is a decimal value with a range of
  - 0
  - to
  - 4294967
  - seconds.
- It has a resolution of hundredths of a second.
- Its smallest valid non-zero value is
  - 0.001
- Its default value is the value of the [slave\\_net\\_timeout](#) system variable divided by 2.
- If it's set to
  - 0
  - , then heartbeats are disabled.

Heartbeats are sent by the primary only if there are no unsent events in the binary log file for a period longer than the interval.

If the [RESET SLAVE](#) statement is executed, then the heartbeat interval is reset to the default.

If the [slave\\_net\\_timeout](#) system variable is set to a value that is lower than the current heartbeat interval, then a warning will be issued.

## TLS Options

The TLS options are used for providing information about [TLS](#). The options can be set even on replicas that are compiled without TLS support. The TLS options are saved to either the default

master.info

file or the file that is configured by the [master\\_info\\_file](#) option, but these TLS options are ignored unless the replica supports TLS.

See [Replication with Secure Connections](#) for more information.

#### MASTER\_SSL

The

MASTER\_SSL

option for

CHANGE MASTER

tells the replica whether to force [TLS](#) for the connection. The valid values are

0

or

1

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    MASTER_SSL=1;  
START SLAVE;
```

#### MASTER\_SSL\_CA

The

MASTER\_SSL\_CA

option for

CHANGE MASTER

defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for [TLS](#). This option requires that you use the absolute path, not a relative path. This option implies the [MASTER\\_SSL](#) option.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
  MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
  MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
  MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;
```

See [Secure Connections Overview: Certificate Authorities \(CAs\)](#) for more information.

The maximum length of

MASTER\_SSL\_CA  
string is 511 characters.

#### MASTER\_SSL\_CAPATH

The

MASTER\_SSL\_CAPATH  
option for  
CHANGE MASTER

defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for [TLS](#). This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the [openssl rehash](#) command. This option implies the [MASTER\\_SSL](#) option.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
  MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
  MASTER_SSL_CAPATH='/etc/my.cnf.d/certificates/ca/',
  MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;
```

See [Secure Connections Overview: Certificate Authorities \(CAs\)](#) for more information.

The maximum length of

MASTER\_SSL\_CA\_PATH  
string is 511 characters.

#### MASTER\_SSL\_CERT

The

MASTER\_SSL\_CERT  
option for  
CHANGE MASTER

defines a path to the X509 certificate file to use for [TLS](#). This option requires that you use the absolute path, not a relative path. This option implies the [MASTER\\_SSL](#) option.

For example:

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
  MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
  MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
  MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;
```

The maximum length of

MASTER\_SSL\_CERT  
string is 511 characters.

## MASTER\_SSL\_CRL

The

MASTER\_SSL\_CRL  
option for  
CHANGE MASTER

defines a path to a PEM file that should contain one or more revoked X509 certificates to use for [TLS](#). This option requires that you use the absolute path, not a relative path.

This option is only supported if the server was built with OpenSSL. If the server was built with yaSSL, then this option is not supported. See [TLS and Cryptography Libraries Used by MariaDB](#) for more information about which libraries are used on which platforms.

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',  
MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',  
MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',  
MASTER_SSL_VERIFY_SERVER_CERT=1,  
MASTER_SSL_CRL='/etc/my.cnf.d/certificates/crl.pem';  
START SLAVE;
```

See [Secure Connections Overview: Certificate Revocation Lists \(CRLs\)](#) for more information.

The maximum length of

MASTER\_SSL\_CRL  
string is 511 characters.

## MASTER\_SSL\_CRLPATH

The

MASTER\_SSL\_CRLPATH  
option for  
CHANGE MASTER

defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for [TLS](#). This option requires that you use the absolute path, not a relative path. The directory specified by this variable needs to be run through the [openssl rehash](#) command.

This option is only supported if the server was built with OpenSSL. If the server was built with yaSSL, then this option is not supported. See [TLS and Cryptography Libraries Used by MariaDB](#) for more information about which libraries are used on which platforms.

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',  
MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',  
MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',  
MASTER_SSL_VERIFY_SERVER_CERT=1,  
MASTER_SSL_CRLPATH='/etc/my.cnf.d/certificates/crl/';  
START SLAVE;
```

See [Secure Connections Overview: Certificate Revocation Lists \(CRLs\)](#) for more information.

The maximum length of

MASTER\_SSL\_CRL\_PATH  
string is 511 characters.

## MASTER\_SSL\_KEY

The

MASTER\_SSL\_KEY  
option for  
CHANGE MASTER

defines a path to a private key file to use for [TLS](#). This option requires that you use the absolute path, not a relative path. This option implies the [MASTER\\_SSL](#) option.

For example:

```

STOP SLAVE;
CHANGE MASTER TO
  MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
  MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
  MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
  MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;

```

The maximum length of

MASTER\_SSL\_KEY  
string is 511 characters.

#### MASTER\_SSL\_CIPHER

The

MASTER\_SSL\_CIPHER  
option for  
CHANGE MASTER

defines the list of permitted ciphers or cipher suites to use for [TLS](#). Besides cipher names, if MariaDB was compiled with OpenSSL, this option could be set to "SSLv3" or "TLSv1.2" to allow all SSLv3 or all TLSv1.2 ciphers. Note that the TLSv1.3 ciphers cannot be excluded when using OpenSSL, even by using this option. See [Using TLSv1.3](#) for details. This option implies the [MASTER\\_SSL](#) option.

For example:

```

STOP SLAVE;
CHANGE MASTER TO
  MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
  MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
  MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
  MASTER_SSL_VERIFY_SERVER_CERT=1,
  MASTER_SSL_CIPHER='TLSv1.2';
START SLAVE;

```

The maximum length of

MASTER\_SSL\_CIPHER  
string is 511 characters.

#### MASTER\_SSL\_VERIFY\_SERVER\_CERT

The

MASTER\_SSL\_VERIFY\_SERVER\_CERT  
option for  
CHANGE MASTER  
enables [server certificate verification](#). This option is disabled by default.

For example:

```

STOP SLAVE;
CHANGE MASTER TO
  MASTER_SSL_CERT='/etc/my.cnf.d/certificates/server-cert.pem',
  MASTER_SSL_KEY='/etc/my.cnf.d/certificates/server-key.pem',
  MASTER_SSL_CA='/etc/my.cnf.d/certificates/ca.pem',
  MASTER_SSL_VERIFY_SERVER_CERT=1;
START SLAVE;

```

See [Secure Connections Overview: Server Certificate Verification](#) for more information.

## Binary Log Options

These options are related to the [binary log](#) position on the primary.

#### MASTER\_LOG\_FILE

The

MASTER\_LOG\_FILE

option for  
CHANGE MASTER  
can be used along with  
MASTER\_LOG\_POS  
to specify the coordinates at which the [replica's I/O thread](#) should begin reading from the primary's [binary logs](#) the next time the thread starts.

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    MASTER_LOG_FILE='master2-bin.001',  
    MASTER_LOG_POS=4;  
START SLAVE;
```

The [MASTER\\_LOG\\_FILE](#) and [MASTER\\_LOG\\_POS](#) options cannot be specified if the [RELAY\\_LOG\\_FILE](#) and [RELAY\\_LOG\\_POS](#) options were also specified.

The [MASTER\\_LOG\\_FILE](#) and [MASTER\\_LOG\\_POS](#) options are effectively ignored if you enable [GTID](#) mode for replication by setting the [MASTER\\_USE\\_GTID](#) option to some value other than

no  
in the statement.

### MASTER\_LOG\_POS

The

MASTER\_LOG\_POS  
option for  
CHANGE MASTER  
can be used along with  
MASTER\_LOG\_FILE  
to specify the coordinates at which the [replica's I/O thread](#) should begin reading from the primary's [binary logs](#) the next time the thread starts.

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    MASTER_LOG_FILE='master2-bin.001',  
    MASTER_LOG_POS=4;  
START SLAVE;
```

The [MASTER\\_LOG\\_FILE](#) and [MASTER\\_LOG\\_POS](#) options cannot be specified if the [RELAY\\_LOG\\_FILE](#) and [RELAY\\_LOG\\_POS](#) options were also specified.

The [MASTER\\_LOG\\_FILE](#) and [MASTER\\_LOG\\_POS](#) options are effectively ignored if you enable [GTID](#) mode for replication by setting the [MASTER\\_USE\\_GTID](#) option to some value other than

no  
in the statement.

## Relay Log Options

These options are related to the [relay log](#) position on the replica.

### RELAY\_LOG\_FILE

The

RELAY\_LOG\_FILE  
option for  
CHANGE MASTER  
can be used along with the [RELAY\\_LOG\\_POS](#) option to specify the coordinates at which the [replica's SQL thread](#) should begin reading from the [relay log](#) the next time the thread starts.

The

```
CHANGE MASTER  
statement usually deletes all relay log files. However, if the  
RELAY_LOG_FILE  
and/or  
RELAY_LOG_POS  
options are specified, then existing relay log files are kept.
```

When you want to change the [relay log](#) position, you only need to stop the [replica's SQL thread](#). The [replica's I/O thread](#) can continue running. The [STOP SLAVE](#) and [START SLAVE](#) statements support the

```
SQL_THREAD  
option for this scenario. For example:
```

```
STOP SLAVE SQL_THREAD;  
CHANGE MASTER TO  
    RELAY_LOG_FILE='slave-relay-bin.006',  
    RELAY_LOG_POS=4025;  
START SLAVE SQL_THREAD;
```

When the value of this option is changed, the metadata about the [replica's SQL thread's](#) position in the [relay logs](#) will also be changed in the [relay-log.info](#) file or the file that is configured by the [relay\\_log\\_info\\_file](#) system variable.

The [RELAY\\_LOG\\_FILE](#) and [RELAY\\_LOG\\_POS](#) options cannot be specified if the [MASTER\\_LOG\\_FILE](#) and [MASTER\\_LOG\\_POS](#) options were also specified.

### RELAY\_LOG\_POS

The

```
RELAY_LOG_POS  
option for  
CHANGE MASTER  
can be used along with the RELAY\_LOG\_FILE option to specify the coordinates at which the replica's SQL thread should begin reading from the  
relay log the next time the thread starts.
```

The

```
CHANGE MASTER  
statement usually deletes all relay log files. However, if the  
RELAY_LOG_FILE  
and/or  
RELAY_LOG_POS  
options are specified, then existing relay log files are kept.
```

When you want to change the [relay log](#) position, you only need to stop the [replica's SQL thread](#). The [replica's I/O thread](#) can continue running. The [STOP SLAVE](#) and [START SLAVE](#) statements support the

```
SQL_THREAD  
option for this scenario. For example:
```

```
STOP SLAVE SQL_THREAD;  
CHANGE MASTER TO  
    RELAY_LOG_FILE='slave-relay-bin.006',  
    RELAY_LOG_POS=4025;  
START SLAVE SQL_THREAD;
```

When the value of this option is changed, the metadata about the [replica's SQL thread's](#) position in the [relay logs](#) will also be changed in the [relay-log.info](#) file or the file that is configured by the [relay\\_log\\_info\\_file](#) system variable.

The [RELAY\\_LOG\\_FILE](#) and [RELAY\\_LOG\\_POS](#) options cannot be specified if the [MASTER\\_LOG\\_FILE](#) and [MASTER\\_LOG\\_POS](#) options were also specified.

## GTID Options

### MASTER\_USE\_GTID

The

MASTER\_USE\_GTID  
option for  
CHANGE MASTER  
can be used to configure the replica to use the [global transaction ID \(GTID\)](#) when connecting to a primary. The possible values are:

- current\_pos
  - Replicate in **GTID** mode and use [gtid\\_current\\_pos](#) as the position to start downloading transactions from the primary.
- slave\_pos
  - Replicate in **GTID** mode and use [gtid\\_slave\\_pos](#) as the position to start downloading transactions from the primary. From [MariaDB 10.5.1](#),
- replica\_pos
  - is an alias for
  - slave\_pos
- no
  - Don't replicate in **GTID** mode.

For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    MASTER_USE_GTID = current_pos;  
START SLAVE;
```

Or:

```
STOP SLAVE;  
SET GLOBAL gtid_slave_pos='0-1-153';  
CHANGE MASTER TO  
    MASTER_USE_GTID = slave_pos;  
START SLAVE;
```

## Replication Filter Options

Also see [Replication filters](#).

### IGNORE\_SERVER\_IDS

The

IGNORE\_SERVER\_IDS  
option for  
CHANGE MASTER  
can be used to configure a [replica](#) to ignore [binary log](#) events that originated from certain servers. Filtered [binary log](#) events will not get logged to the replica's [relay log](#), and they will not be applied by the replica.

The option's value can be specified by providing a comma-separated list of [server\\_id](#) values. For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    IGNORE_SERVER_IDS = (3,5);  
START SLAVE;
```

If you would like to clear a previously set list, then you can set the value to an empty list. For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    IGNORE_SERVER_IDS = ();  
START SLAVE;
```

### DO\_DOMAIN\_IDS

MariaDB starting with [10.1.2](#)

The

DO\_DOMAIN\_IDS

option for  
CHANGE MASTER  
was first added in [MariaDB 10.1.2](#).

The

DO\_DOMAIN\_IDS  
option for  
CHANGE MASTER  
can be used to configure a [replica](#) to only apply [binary log](#) events if the transaction's [GTID](#) is in a specific [gtid\\_domain\\_id](#) value. Filtered [binary log](#) events will not get logged to the replica's [relay log](#), and they will not be applied by the replica.

The option's value can be specified by providing a comma-separated list of [gtid\\_domain\\_id](#) values. Duplicate values are automatically ignored. For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    DO_DOMAIN_IDS = (1,2);  
START SLAVE;
```

If you would like to clear a previously set list, then you can set the value to an empty list. For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    DO_DOMAIN_IDS = ();  
START SLAVE;
```

The [DO\\_DOMAIN\\_IDS](#) option and the [IGNORE\\_DOMAIN\\_IDS](#) option cannot both be set to non-empty values at the same time. If you want to set the [DO\\_DOMAIN\\_IDS](#) option, and the [IGNORE\\_DOMAIN\\_IDS](#) option was previously set, then you need to clear the value of the [IGNORE\\_DOMAIN\\_IDS](#) option. For example:

```
STOP SLAVE;  
CHANGE MASTER TO  
    IGNORE_DOMAIN_IDS = (),  
    DO_DOMAIN_IDS = (1,2);  
START SLAVE;
```

The

DO\_DOMAIN\_IDS  
option can only be specified if the replica is replicating in [GTID](#) mode. Therefore, the [MASTER\\_USE\\_GTID](#) option must also be set to some value other than  
no  
in order to use this option.

## IGNORE\_DOMAIN\_IDS

MariaDB starting with [10.1.2](#)

The

IGNORE\_DOMAIN\_IDS  
option for  
CHANGE MASTER  
was first added in [MariaDB 10.1.2](#).

The

IGNORE\_DOMAIN\_IDS  
option for  
CHANGE MASTER  
can be used to configure a [replica](#) to ignore [binary log](#) events if the transaction's [GTID](#) is in a specific [gtid\\_domain\\_id](#) value. Filtered [binary log](#) events will not get logged to the replica's [relay log](#), and they will not be applied by the replica.

The option's value can be specified by providing a comma-separated list of [gtid\\_domain\\_id](#) values. Duplicate values are automatically ignored. For example:

```
STOP SLAVE;
CHANGE MASTER TO
  IGNORE_DOMAIN_IDS = (1,2);
START SLAVE;
```

If you would like to clear a previously set list, then you can set the value to an empty list. For example:

```
STOP SLAVE;
CHANGE MASTER TO
  IGNORE_DOMAIN_IDS = ();
START SLAVE;
```

The `DO_DOMAIN_IDS` option and the `IGNORE_DOMAIN_IDS` option cannot both be set to non-empty values at the same time. If you want to set the `IGNORE_DOMAIN_IDS` option, and the `DO_DOMAIN_IDS` option was previously set, then you need to clear the value of the `DO_DOMAIN_IDS` option. For example:

```
STOP SLAVE;
CHANGE MASTER TO
  DO_DOMAIN_IDS = (),
  IGNORE_DOMAIN_IDS = (1,2);
START SLAVE;
```

The

`IGNORE_DOMAIN_IDS` option can only be specified if the replica is replicating in `GTID` mode. Therefore, the `MASTER_USE_GTID` option must also be set to some value other than `no` in order to use this option.

## Delayed Replication Options

### `MASTER_DELAY`

MariaDB starting with 10.2.3

The

`MASTER_DELAY` option for `CHANGE MASTER` was first added in MariaDB 10.2.3 to enable `delayed replication`.

The

`MASTER_DELAY` option for `CHANGE MASTER` can be used to enable `delayed replication`. This option specifies the time in seconds (at least) that a replica should lag behind the primary up to a maximum value of 2147483647, or about 68 years. Before executing an event, the replica will first wait, if necessary, until the given time has passed since the event was created on the primary. The result is that the replica will reflect the state of the primary some time back in the past. The default is zero, no delay.

```
STOP SLAVE;
CHANGE MASTER TO
  MASTER_DELAY=3600;
START SLAVE;
```

## Changing Option Values

If you don't specify a given option when executing the

`CHANGE MASTER`

statement, then the option keeps its old value in most cases. Most of the time, there is no need to specify the options that do not need to change. For example, if the password for the user account that the replica uses to connect to its primary has changed, but no other options need to change, then you can just change the `MASTER_PASSWORD` option by executing the following commands:

```
STOP SLAVE;
CHANGE MASTER TO
    MASTER_PASSWORD='new3cret';
START SLAVE;
```

There are some cases where options are implicitly reset, such as when the `MASTER_HOST` and `MASTER_PORT` options are changed.

## Option Persistence

The values of the `MASTER_LOG_FILE` and `MASTER_LOG_POS` options (i.e. the `binary log` position on the primary) and most other options are written to either the default

`master.info`

file or the file that is configured by the `master_info_file` option. The replica's I/O thread keeps this `binary log` position updated as it downloads events only when `MASTER_USE_GTID` option is set to

`NO`

. Otherwise the file is not updated on a per event basis.

The `master_info_file` option can be set either on the command-line or in a server `option group` in an `option file` prior to starting up the server. For example:

```
[mariadb]
...
master_info_file=/mariadb/myserver1-master.info
```

The values of the `RELAY_LOG_FILE` and `RELAY_LOG_POS` options (i.e. the `relay log` position) are written to either the default

`relay-log.info`

file or the file that is configured by the `relay_log_info_file` system variable. The replica's SQL thread keeps this `relay log` position updated as it applies events.

The `relay_log_info_file` system variable can be set either on the command-line or in a server `option group` in an `option file` prior to starting up the server. For example:

```
[mariadb]
...
relay_log_info_file=/mariadb/myserver1-relay-log.info
```

## GTID Persistence

If the replica is replicating `binary log` events that contain `GTIDs`, then the replica's SQL thread will write every GTID that it applies to the `mysql.gtid_slave_pos` table. This GTID can be inspected and modified through the `gtid_slave_pos` system variable.

If the replica has the `log_slave_updates` system variable enabled and if the replica has the `binary log` enabled, then every write by the replica's SQL thread will also go into the replica's `binary log`. This means that `GTIDs` of replicated transactions would be reflected in the value of the `gtid_binlog_pos` system variable.

## Creating a Slave from a Backup

The

`CHANGE MASTER`

statement is useful for setting up a replica when you have a backup of the primary and you also have the `binary log` position or `GTID` position corresponding to the backup.

After restoring the backup on the replica, you could execute something like this to use the `binary log` position:

```
CHANGE MASTER TO
    MASTER_LOG_FILE='master2-bin.001',
    MASTER_LOG_POS=4;
START SLAVE;
```

Or you could execute something like this to use the `GTID` position:

```
SET GLOBAL gtid_slave_pos='0-1-153';
CHANGE MASTER TO
    MASTER_USE_GTID=slave_pos;
START SLAVE;
```

See [Setting up a Replication Slave with Mariabackup](#) for more information on how to do this with [Mariabackup](#).

## Example

The following example changes the primary and primary's binary log coordinates. This is used when you want to set up the replica to replicate the primary:

```
CHANGE MASTER TO
  MASTER_HOST='master2.mycompany.com',
  MASTER_USER='replication',
  MASTER_PASSWORD='bigs3cret',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='master2-bin.001',
  MASTER_LOG_POS=4,
  MASTER_CONNECT_RETRY=10;
START SLAVE;
```

## See Also

- [Setting up replication](#)
- [START SLAVE](#)
- [Multi-source replication](#)
- [RESET SLAVE](#). Removes a connection created with  
CHANGE MASTER TO

- [Global Transaction ID](#)

## 1.1.2.5.2 START SLAVE

The terms *master* and *slave* have historically been used in replication, but the terms *primary* and *replica* are now preferred. The old terms are used still used in parts of the documentation, and in MariaDB commands, although MariaDB 10.5 has begun the process of renaming. The documentation process is ongoing. See [MDEV-18777](#) to follow progress on this effort.

## Syntax

```
START SLAVE ["connection_name"] [thread_type [, thread_type] ... ] [FOR CHANNEL "connection_name"]
START SLAVE ["connection_name"] [SQL_THREAD] UNTIL
  MASTER_LOG_FILE = 'log_name', MASTER_LOG_POS = log_pos [FOR CHANNEL "connection_name"]
START SLAVE ["connection_name"] [SQL_THREAD] UNTIL
  RELAY_LOG_FILE = 'log_name', RELAY_LOG_POS = log_pos [FOR CHANNEL "connection_name"]
START SLAVE ["connection_name"] [SQL_THREAD] UNTIL
  MASTER_GTID_POS = <GTID position> [FOR CHANNEL "connection_name"]
START ALL SLAVES [thread_type [, thread_type]]

START REPLICA ["connection_name"] [thread_type [, thread_type] ... ] -- from 10.5.1
START REPLICA ["connection_name"] [SQL_THREAD] UNTIL
  MASTER_LOG_FILE = 'log_name', MASTER_LOG_POS = log_pos -- from 10.5.1
START REPLICA ["connection_name"] [SQL_THREAD] UNTIL
  RELAY_LOG_FILE = 'log_name', RELAY_LOG_POS = log_pos -- from 10.5.1
START REPLICA ["connection_name"] [SQL_THREAD] UNTIL
  MASTER_GTID_POS = <GTID position> -- from 10.5.1
START ALL REPLICAS [thread_type [, thread_type]] -- from 10.5.1

thread_type: IO_THREAD | SQL_THREAD
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [START SLAVE UNTIL](#)
  2. [connection\\_name](#)
  3. [START ALL SLAVES](#)
  4. [START REPLICA](#)
3. [See Also](#)

## Description

```
START SLAVE
(
START REPLICA
  from MariaDB 10.5.1 ) with no thread_type options starts both of the replica threads (see replication). The I/O thread reads events from the primary server and stores them in the relay log. The SQL thread reads events from the relay log and executes them.
```

START SLAVE

requires the [SUPER](#) privilege, or, from [MariaDB 10.5.2](#), the [REPLICATION SLAVE ADMIN](#) privilege.

If

START SLAVE

succeeds in starting the replica threads, it returns without any error. However, even in that case, it might be that the replica threads start and then later stop (for example, because they do not manage to connect to the primary or read its [binary log](#), or some other problem).

START SLAVE

does not warn you about this. You must check the replica's [error log](#) for error messages generated by the replica threads, or check that they are running satisfactorily with [SHOW SLAVE STATUS](#) ( [SHOW REPLICAS STATUS](#) from [MariaDB 10.5.1](#) ).

## START SLAVE UNTIL

START SLAVE UNTIL

refers to the

SQL\_THREAD

replica position at which the

SQL\_THREAD

replication will halt. If

SQL\_THREAD

isn't specified both threads are started.

START SLAVE UNTIL master\_gtid\_pos=xxx

is also supported. See [Global Transaction ID/START SLAVE UNTIL master\\_gtid\\_pos=xxx](#) for more details.

## connection\_name

If there is only one nameless primary, or the default primary (as specified by the [default\\_master\\_connection](#) system variable) is intended,

connection\_name

can be omitted. If provided, the

START SLAVE

statement will apply to the specified primary.

connection\_name

is case-insensitive.

MariaDB starting with [10.7.0](#)

The

FOR CHANNEL

keyword was added for MySQL compatibility. This is identical as using the channel\_name directly after

START SLAVE

## START ALL SLAVES

START ALL SLAVES

starts all configured replicas (replicas with master\_host not empty) that were not started before. It will give a

note

for all started connections. You can check the notes with [SHOW WARNINGS](#).

## START REPLICA

MariaDB starting with [10.5.1](#)

START REPLICA

is an alias for

START SLAVE

from [MariaDB 10.5.1](#).

## See Also

- [Setting up replication](#).
- [CHANGE MASTER TO](#) is used to create and change connections.
- [STOP SLAVE](#) is used to stop a running connection.
- [RESET SLAVE](#) is used to reset parameters for a connection and also to permanently delete a primary connection.

## 1.1.2.5.3 STOP SLAVE

The terms *master* and *slave* have historically been used in replication, but the terms *primary* and *replica* are now preferred. The old terms are used still used in parts of the documentation, and in MariaDB commands, although MariaDB 10.5 has begun the process of renaming. The documentation process is ongoing. See [MDEV-18777](#) to follow progress on this effort.

### Syntax

```
STOP SLAVE ["connection_name"] [thread_type [, thread_type] ... ] [FOR CHANNEL "connection_name"]

STOP ALL SLAVES [thread_type [, thread_type]]

STOP REPLICA ["connection_name"] [thread_type [, thread_type] ... ] -- from 10.5.1

STOP ALL REPLICAS [thread_type [, thread_type]] -- from 10.5.1

thread_type: IO_THREAD | SQL_THREAD
```

### Contents

1. [Syntax](#)
2. [Description](#)
  - 1. [STOP ALL SLAVES](#)
  - 2. [connection\\_name](#)
  - 3. [STOP REPLICA](#)
3. [See Also](#)

### Description

Stops the replica threads.

```
STOP SLAVE
requires the SUPER privilege, or, from MariaDB 10.5.2 , the REPLICATION SLAVE ADMIN privilege.
```

Like [START SLAVE](#) , this statement may be used with the

IO\_THREAD  
and  
SQL\_THREAD  
options to name the thread or threads to be stopped. In almost all cases, one never need to use the  
thread\_type  
options.

STOP SLAVE  
waits until any current replication event group affecting one or more non-transactional tables has finished executing (if there is any such replication group), or until the user issues a [KILL QUERY](#) or [KILL CONNECTION](#) statement.

Note that

```
STOP SLAVE
doesn't delete the connection permanently. Next time you execute START SLAVE or the MariaDB server restarts, the replica connection is
restored with it's original arguments . If you want to delete a connection, you should execute RESET SLAVE .
```

### STOP ALL SLAVES

```
STOP ALL SLAVES
stops all your running replicas. It will give you a
note
for every stopped connection. You can check the notes with SHOW WARNINGS .
```

### connection\_name

The

```
connection_name
option is used for multi-source replication .
```

If there is only one nameless master, or the default master (as specified by the [default\\_master\\_connection](#) system variable) is intended,  
connection\_name  
can be omitted. If provided, the  
STOP SLAVE  
statement will apply to the specified master.

`connection_name`  
is case-insensitive.

MariaDB starting with 10.7.0

The

`FOR CHANNEL`  
keyword was added for MySQL compatibility. This is identical as using the `channel_name` directly after  
`STOP SLAVE`

## STOP REPLICA

MariaDB starting with 10.5.1

`STOP REPLICA`  
is an alias for  
`STOP SLAVE`  
from MariaDB 10.5.1 .

## See Also

- [CHANGE MASTER TO](#) is used to create and change connections.
- [START SLAVE](#) is used to start a predefined connection.
- [RESET SLAVE](#) is used to reset parameters for a connection and also to permanently delete a master connection.

## 1.1.2.5.4 RESET SLAVE

The terms *master* and *slave* have historically been used in replication, but the terms *primary* and *replica* are now preferred. The old terms are used still used in parts of the documentation, and in MariaDB commands, although MariaDB 10.5 has begun the process of renaming. The documentation process is ongoing. See [MDEV-18777](#) to follow progress on this effort.

## Syntax

```
RESET SLAVE ["connection_name"] [ALL] [FOR CHANNEL "connection_name"]
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [connection\\_name](#)
  2. [RESET REPLICA](#)
3. [See Also](#)

## Description

`RESET SLAVE` makes the slave forget its [replication](#) position in the master's [binary log](#). This statement is meant to be used for a clean start. It deletes the `master.info` and `relay-log.info` files, all the [relay log](#) files, and starts a new relay log file. To use `RESET SLAVE`, the slave replication threads must be stopped (use `STOP SLAVE` if necessary).

Note: All relay log files are deleted, even if they have not been completely executed by the slave SQL thread. (This is a condition likely to exist on a replication slave if you have issued a `STOP SLAVE` statement or if the slave is highly loaded.)

Note:

`RESET REPLICA`  
does not reset the global  
`gtid_slave_pos`  
variable. This means that a replica server configured with  
`CHANGE MASTER TO MASTER_USE_GTID=slave_pos`  
will not receive events with GTIDs occurring before the state saved in  
`gtid_slave_pos`  
. If the intent is to reprocess these events,  
`gtid_slave_pos`

```
must be manually reset, e.g. by executing  
set global gtid_slave_pos=""
```

Connection information stored in the master.info file is immediately reset using any values specified in the corresponding startup options. This information includes values such as master host, master port, master user, and master password. If the slave SQL thread was in the middle of replicating temporary tables when it was stopped, and RESET SLAVE is issued, these replicated temporary tables are deleted on the slave.

The

```
ALL  
also resets the  
PORT  
,  
HOST  
,  
USER  
and  
PASSWORD
```

parameters for the slave. If you are using a connection name, it will permanently delete it and it will not show up anymore in [SHOW ALL SLAVES STATUS](#).

`connection_name`

The

```
connection_name  
option is used for multi-source replication.
```

If there is only one nameless master, or the default master (as specified by the `default_master_connection` system variable) is intended,

```
connection_name  
can be omitted. If provided, the  
RESET SLAVE  
statement will apply to the specified master.  
connection_name  
is case-insensitive.
```

MariaDB starting with [10.7.0](#)

The

```
FOR CHANNEL  
keyword was added for MySQL compatibility. This is identical as using the channel_name directly after  
RESET SLAVE
```

**RESET REPLICA**

MariaDB starting with [10.5.1](#)

```
RESET REPLICA  
is an alias for  
RESET SLAVE  
from MariaDB 10.5.1.
```

## See Also

- [STOP SLAVE](#) stops the slave, but it can be restarted with [START SLAVE](#) or after next MariaDB server restart.

## 1.1.2.5.5 SET GLOBAL SQL\_SLAVE\_SKIP\_COUNTER

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [Multiple Replication Domains](#)
5. [See Also](#)

## Syntax

```
SET GLOBAL sql_slave_skip_counter = N
```

## Description

This statement skips the next

*N*

events from the master. This is useful for recovering from [replication stops](#) caused by a statement.

If multi-source replication is used, this statement applies to the default connection. It could be necessary to change the value of the

[default\\_master\\_connection](#)

server system variable.

Note that, if the event is a [transaction](#), the whole transaction will be skipped. With non-transactional engines, an event is always a single statement.

This statement is valid only when the slave threads are not running. Otherwise, it produces an error.

The statement does not automatically restart the slave threads.

## Example

```
SHOW SLAVE STATUS \G
...
SET GLOBAL sql_slave_skip_counter = 1;
START SLAVE;
```

Multi-source replication:

```
SET @@default_master_connection = 'master_01';
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;
START SLAVE;
```

## Multiple Replication Domains

`sql_slave_skip_counter`  
can't be used to skip transactions on a slave if [GTID replication](#) is in use and if

[gtid\\_slave\\_pos](#)

contains multiple

[gtid\\_domain\\_id](#)

values. In that case, you'll get an error like the following:

```
ERROR 1966 (HY000): When using parallel replication and GTID with multiple
replication domains, @@sql_slave_skip_counter can not be used. Instead,
setting @@gtid_slave_pos explicitly can be used to skip to after a given GTID
position.
```

In order to skip transactions in cases like this, you will have to manually change

[gtid\\_slave\\_pos](#)

## See Also

- [Selectively Skipping Replication of Binlog Events](#)

## 1.1.2.5.6 SHOW RELAYLOG EVENTS

The terms *master* and *slave* have historically been used in replication, but the terms *primary* and *replica* are now preferred. The old terms are

used still used in parts of the documentation, and in MariaDB commands, although MariaDB 10.5 has begun the process of renaming. The documentation process is ongoing. See [MDEV-18777](#) to follow progress on this effort.

## Syntax

```
SHOW RELAYLOG ['connection_name'] EVENTS  
[IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]  
[ FOR CHANNEL 'channel_name']
```

## Description

On `replicas`, this command shows the events in the `relay log`. If  
`'log_name'`  
is not specified, the first relay log is shown.

Syntax for the

`LIMIT`  
clause is the same as for `SELECT ... LIMIT`.

Using the

`LIMIT`  
clause is highly recommended because the  
`SHOW RELAYLOG EVENTS`  
command returns the complete contents of the relay log, which can be quite large.

This command does not return events related to setting user and system variables. If you need those, use `mariadb-binlog/mysqlbinlog`.

On the primary, this command does nothing.

Requires the `REPLICA MONITOR` privilege (>= MariaDB 10.5.9), the `REPLICATION SLAVE ADMIN` privilege (>= MariaDB 10.5.2) or the `REPLICATION SLAVE` privilege (<= MariaDB 10.5.1).

`connection_name`

If there is only one nameless primary, or the default primary (as specified by the `default_master_connection` system variable) is intended,

`connection_name`  
can be omitted. If provided, the  
`SHOW RELAYLOG`  
statement will apply to the specified primary.  
`connection_name`  
is case-insensitive.

MariaDB starting with 10.7.0

The  
`FOR CHANNEL`  
keyword was added for MySQL compatibility. This is identical as using the `channel_name` directly after  
`SHOW RELAYLOG`

## 1.1.2.5.7 SHOW SLAVE STATUS

## Syntax

```
SHOW SLAVE ["connection_name"] STATUS [FOR CHANNEL "connection_name"]  
SHOW REPLICA ["connection_name"] STATUS -- From MariaDB 10.5.1
```

or

```
SHOW ALL SLAVES STATUS  
SHOW ALL REPLICAS STATUS -- From MariaDB 10.5.1
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [Multi-Source](#)
  2. [Column Descriptions](#)
  3. [SHOW REPLICA STATUS](#)
3. [Examples](#)
4. [See Also](#)

## Description

This statement is to be run on a replica and provides status information on essential parameters of the `replica` threads.

This statement requires the `SUPER` privilege, the `REPLICATION_CLIENT` privilege, or, from MariaDB 10.5.2 , the `REPLICATION SLAVE ADMIN` privilege, or, from MariaDB 10.5.9 , the `REPLICA MONITOR` privilege.

### Multi-Source

The

```
FULL  
and  
"connection_name"  
options allow you to connect to many primaries at the same time .
```

```
ALL SLAVES  
(or  
ALL REPLICAS  
from MariaDB 10.5.1 ) gives you a list of all connections to the primary nodes.
```

The rows will be sorted according to

```
Connection_name
```

If you specify a

```
connection_name  
, you only get the information about that connection. If  
connection_name  
is not used, then the name set by  
default_master_connection  
is used. If the connection name doesn't exist you will get an error:  
There is no master connection for 'xxx'
```

MariaDB starting with 10.7.0

The

```
FOR CHANNEL  
keyword was added for MySQL compatibility. This is identical as using the channel_name directly after  
SHOW SLAVE
```

### Column Descriptions

Name	Description	Added
Connection_name	Name of the primary connection. Returned with <code>SHOW ALL SLAVES STATUS</code> (or <code>SHOW ALL REPLICAS STATUS</code> from MariaDB 10.5.1 ) only.	
Slave_SQL_State	State of SQL thread. Returned with <code>SHOW ALL SLAVES STATUS</code> (or <code>SHOW ALL REPLICAS STATUS</code> from MariaDB 10.5.1 ) only. See <a href="#">Slave SQL Thread States</a> .	
Slave_IO_State	State of I/O thread. See <a href="#">Slave I/O Thread States</a> .	
Master_host	Master host that the replica is connected to.	

Master_user	Account user name being used to connect to the primary.	
Master_port	The port being used to connect to the primary.	
Connect_Retry	Time in seconds between retries to connect. The default is 60. The <a href="#">CHANGE MASTER TO</a> statement can set this. The <a href="#">master-retry-count</a> option determines the maximum number of reconnection attempts.	
Master_Log_File	Name of the primary <a href="#">binary log</a> file that the I/O thread is currently reading from.	
Read_Master_Log_Pos	Position up to which the I/O thread has read in the current primary <a href="#">binary log</a> file.	
Relay_Log_File	Name of the relay log file that the SQL thread is currently processing.	
Relay_Log_Pos	Position up to which the SQL thread has finished processing in the current relay log file.	
Relay_Master_Log_File	Name of the primary <a href="#">binary log</a> file that contains the most recent event executed by the SQL thread.	
Slave_IO_Running	Whether the replica I/O thread is running and connected ( Yes ), running but not connected to a primary ( Connecting ) or not running ( No ).	
Slave_SQL_Running	Whether or not the SQL thread is running.	
Replicate_Do_DB	Databases specified for replicating with the  <a href="#">replicate_do_db</a>  option.	
Replicate_Ignore_DB	Databases specified for ignoring with the  <a href="#">replicate_ignore_db</a>  option.	
Replicate_Do_Table	Tables specified for replicating with the  <a href="#">replicate_do_table</a>  option.	
Replicate_Ignore_Table	Tables specified for ignoring with the  <a href="#">replicate_ignore_table</a>  option.	
Replicate_Wild_Do_Table	Tables specified for replicating with the  <a href="#">replicate_wild_do_table</a>  option.	
Replicate_Wild_Ignore_Table	Tables specified for ignoring with the  <a href="#">replicate_wild_ignore_table</a>  option.	
Last_Error	Alias for  Last_SQL_Error (see below)	
Skip_Counter	Number of events that a replica skips from the master, as recorded in the <a href="#">sql_slave_skip_counter</a> system variable.	
Exec_Master_Log_Pos	Position up to which the SQL thread has processed in the current master <a href="#">binary log</a> file. Can be used to start a new replica from a current replica with the <a href="#">CHANGE MASTER TO ... MASTER_LOG_POS</a> option.	
Relay_Log_Space	Total size of all relay log files combined.	

Until_Condition		
Until_Log_File	The MASTER_LOG_FILE value of the <b>START SLAVE UNTIL</b> condition.	
Until_Log_Pos	The MASTER_LOG_POS value of the <b>START SLAVE UNTIL</b> condition.	
Master_SSL_Allowed	Whether an SSL connection is permitted ( Yes , not permitted ( No ) or permitted but without the replica having SSL support enabled ( Ignored )	
Master_SSL_CA_File	The MASTER_SSL_CA option of the  <b>CHANGE MASTER TO</b>  statement.	
Master_SSL_CA_Path	The MASTER_SSL_CAPATH option of the  <b>CHANGE MASTER TO</b>  statement.	
Master_SSL_Cert	The MASTER_SSL_CERT option of the  <b>CHANGE MASTER TO</b>  statement.	
Master_SSL_Cipher	The MASTER_SSL_CIPHER option of the  <b>CHANGE MASTER TO</b>  statement.	
Master_SSL_Key	The MASTER_SSL_KEY option of the  <b>CHANGE MASTER TO</b>  statement.	
Seconds_Behind_Master	Difference between the timestamp logged on the master for the event that the replica is currently processing, and the current timestamp on the replica. Zero if the replica is not currently processing an event. With <a href="#">parallel replication</a> , seconds_behind_master is updated only after transactions commit.	
Master_SSL_Verify_Server_Cert	The MASTER_SSL_VERIFY_SERVER_CERT option of the  <b>CHANGE MASTER TO</b>  statement.	
Last_IO_Error	<a href="#">Error code</a> of the most recent error that caused the I/O thread to stop (also recorded in the replica's error log). 0 means no error. <b>RESET SLAVE</b> or <b>RESET MASTER</b> will reset this value.	

Last_IO_Error	<a href="#">Error message</a> of the most recent error that caused the I/O thread to stop (also recorded in the replica's error log). An empty string means no error. <a href="#">RESET SLAVE</a> or <a href="#">RESET MASTER</a> will reset this value.	
Last_SQL_Erno	<a href="#">Error code</a> of the most recent error that caused the SQL thread to stop (also recorded in the replica's error log). 0 means no error. <a href="#">RESET SLAVE</a> or <a href="#">RESET MASTER</a> will reset this value.	
Last_SQL_Error	<a href="#">Error message</a> of the most recent error that caused the SQL thread to stop (also recorded in the replica's error log). An empty string means no error. <a href="#">RESET SLAVE</a> or <a href="#">RESET MASTER</a> will reset this value.	
Replicate_Ignore_Server_Ids	List of <a href="#">server_ids</a> that are currently being ignored for replication purposes, or an empty string for none, as specified in the  IGNORE_SERVER_IDS option of the  CHANGE MASTER TO  statement.	
Master_Server_Id	The master's <a href="#">server_id</a> value.	
Master_SSL_Crl	The  MASTER_SSL_CRL option of the  CHANGE MASTER TO  statement.	
Master_SSL_Crlpath	The  MASTER_SSL_CRLPATH option of the  CHANGE MASTER TO  statement.	
Using_Gtid	Whether or not <a href="#">global transaction ID's</a> are being used for replication (can be No ,, Slave_Pos , or Current_Pos ).	
Gtid_IO_Pos	Current <a href="#">global transaction ID</a> value.	
Retried_transactions	Number of retried transactions for this connection. Returned with  SHOW ALL SLAVES STATUS only.	
Max_relay_log_size	Max <a href="#">relay log</a> size for this connection. Returned with  SHOW ALL SLAVES STATUS only.	
Executed_log_entries	How many log entries the replica has executed. Returned with  SHOW ALL SLAVES STATUS only.	
Slave_received_heartbeats	How many heartbeats we have got from the master. Returned with  SHOW ALL SLAVES STATUS only.	
Slave_heartbeat_period	How often to request a heartbeat packet from the master (in seconds). Returned with  SHOW ALL SLAVES STATUS only.	
Gtid_Slave_Pos	GTID of the last event group replicated on a replica server, for each replication domain, as stored in the <a href="#">gtid_slave_pos</a> system variable. Returned with  SHOW ALL SLAVES STATUS only.	

SQL_Delay	<p>Value specified by</p> <pre>MASTER_DELAY in  CHANGE MASTER</pre> <p>(or 0 if none).</p>	MariaDB 10.2.3
SQL_Remaining_Delay	<p>When the replica is delaying the execution of an event due to</p> <pre>MASTER_DELAY</pre> <p>, this is the number of seconds of delay remaining before the event will be applied.</p> <p>Otherwise, the value is</p> <pre>NULL</pre>	MariaDB 10.2.3
Slave_SQL_Running_State	<p>The state of the SQL driver threads, same as in</p> <pre>SHOW PROCESSLIST</pre> <p>. When the replica is delaying the execution of an event due to</p> <pre>MASTER_DELAY</pre> <p>, this field displays: "</p> <p>Waiting until MASTER_DELAY seconds after master executed event</p> <p>".</p>	MariaDB 10.2.3
Slave_DDL_Groups	This status variable counts the occurrence of DDL statements. This is a replica-side counter for optimistic parallel replication.	MariaDB 10.3.7
Slave_Non_Transactional_Groups	This status variable counts the occurrence of non-transactional event groups. This is a replica-side counter for optimistic parallel replication.	MariaDB 10.3.7
Slave_Transactional_Groups	This status variable counts the occurrence of transactional event groups. This is a replica-side counter for optimistic parallel replication.	MariaDB 10.3.7

## SHOW REPLICAS STATUS

MariaDB starting with 10.5.1

```
SHOW REPLICAS STATUS
is an alias for
SHOW SLAVE STATUS
from MariaDB 10.5.1 .
```

## Examples

If you issue this statement using the

`mysql`

client, you can use a

`\G`

statement terminator rather than a semicolon to obtain a more readable vertical layout.

```
SHOW SLAVE STATUS\G
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: db01.example.com
Master_User: replicant
Master_Port: 3306
Connect_Retry: 60
Master_Log_File: mariadb-bin.000010
Read_Master_Log_Pos: 548
Relay_Log_File: relay-bin.000004
Relay_Log_Pos: 837
Relay_Master_Log_File: mariadb-bin.000010
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Error:
Skip_Counter: 0
Exec_Master_Log_Pos: 548
Relay_Log_Space: 1497
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Error:
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 101
Master_SSL_Crl:
Master_SSL_Crlpath:
Using_Gtid: No
Gtid_IO_Pos:
```

```

SHOW ALL SLAVES STATUS\G
***** 1. row *****
Connection_name:
Slave_SQL_State: Slave has read all relay log; waiting for the slave I/O thread to update it
Slave_IO_State: Waiting for master to send event
  Master_Host: db01.example.com
  Master_User: replicant
  Master_Port: 3306
  Connect_Retry: 60
  Master_Log_File: mariadb-bin.000010
Read_Master_Log_Pos: 3608
  Relay_Log_File: relay-bin.000004
  Relay_Log_Pos: 3897
Relay_Master_Log_File: mariadb-bin.000010
  Slave_IO_Running: Yes
  Slave_SQL_Running: Yes
  Replicate_Do_DB:
  Replicate_Ignore_DB:
  Replicate_Do_Table:
  Replicate_Ignore_Table:
  Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
  Last_Error:
  Skip_Counter: 0
  Exec_Master_Log_Pos: 3608
  Relay_Log_Space: 4557
  Until_Condition: None
  Until_Log_File:
  Until_Log_Pos: 0
  Master_SSL_Allowed: No
  Master_SSL_CA_File:
  Master_SSL_CA_Path:
    Master_SSL_Cert:
    Master_SSL_Cipher:
    Master_SSL_Key:
  Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
  Last_IO_Error:
  Last_SQL_Error:
Replicate_Ignore_Server_Ids:
  Master_Server_Id: 101
  Master_SSL_Crl:
  Master_SSL_Crlpath:
    Using_Gtid: No
    Gtid_IO_Pos:
  Retried_transactions: 0
  Max_relay_log_size: 104857600
  Executed_log_entries: 40
Slave_received_heartbeats: 11
  Slave_heartbeat_period: 1800.000
  Gtid_Slave_Pos: 0-101-2320

```

You can also access some of the variables directly from status variables:

```

SET @@default_master_connection="test" ;
show status like "%slave%"

Variable_name      Value
Com_show_slave_hosts      0
Com_show_slave_status      0
Com_start_all_slaves      0
Com_start_slave 0
Com_stop_all_slaves      0
Com_stop_slave 0
Rpl_semi_sync_slave_status      OFF
Slave_connections      0
Slave_heartbeat_period 1800.000
Slave_open_temp_tables 0
Slave_received_heartbeats      0
Slave_retried_transactions      0
Slave_running      OFF
Slaves_connected      0
Slaves_running 1

```

## See Also

- [MariaDB replication](#)

### 1.1.2.5.8 SHOW MASTER STATUS

#### Syntax

```

SHOW MASTER STATUS
SHOW BINLOG STATUS -- From MariaDB 10.5.2

```

#### Description

Provides status information about the [binary log](#) files of the primary.

This statement requires the [SUPER](#) privilege, the [REPLICATION\\_CLIENT](#) privilege, or, from [MariaDB 10.5.2](#), the [BINLOG MONITOR](#) privilege.

To see information about the current GTIDs in the binary log, use the [gtid\\_binlog\\_pos](#) variable.

```

SHOW MASTER STATUS
was renamed to
SHOW BINLOG STATUS
in MariaDB 10.5.2 , but the old name remains an alias for compatibility purposes.

```

#### Example

```

SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File          | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mariadb-bin.000016 |      475 |           |           |
+-----+-----+-----+-----+
SELECT @@global.gtid_binlog_pos;
+-----+
| @@global.gtid_binlog_pos |
+-----+
| 0-1-2           |
+-----+

```

## See Also

- [MariaDB replication](#)
- [Using and Maintaining the Binary Log](#)
- [The gtid\\_binlog\\_pos variable](#)

### 1.1.2.5.9 SHOW SLAVE HOSTS

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [SHOW REPLICHOSTS](#)
3. [See Also](#)

## Syntax

```
SHOW SLAVE HOSTS  
SHOW REPLICHOSTS -- from MariaDB 10.5.1
```

## Description

This command is run on the primary and displays a list of replicas that are currently registered with it. Only replicas started with the `--report-host=host_name` option are visible in this list.

The list is displayed on any server (not just the primary server). The output looks like this:

```
SHOW SLAVE HOSTS;  
+-----+-----+-----+  
| Server_id | Host      | Port | Master_id |  
+-----+-----+-----+  
| 192168010 | iconnect2 | 3306 | 192168011 |  
| 1921680101 | athena     | 3306 | 192168011 |  
+-----+-----+-----+
```

- **Server\_id**  
: The unique server ID of the replica server, as configured in the server's option file, or on the command line with `--server-id=value`.
- **Host**  
: The host name of the replica server, as configured in the server's option file, or on the command line with `--report-host=host_name`. Note that this can differ from the machine name as configured in the operating system.
- **Port**  
: The port the replica server is listening on.
- **Master\_id**  
: The unique server ID of the primary server that the replica server is replicating from.

Some MariaDB and MySQL versions report another variable, `rpl_recovery_rank`. This variable was never used, and was eventually removed in [MariaDB 10.1.2](#).

Requires the [REPLICATION MASTER ADMIN](#) privilege (>= [MariaDB 10.5.2](#)) or the [REPLICATION SLAVE](#) privilege (<= [MariaDB 10.5.1](#)).

## SHOW REPLICHOSTS

MariaDB starting with [10.5.1](#)

```
SHOW REPLICHOSTS  
is an alias for  
SHOW SLAVE HOSTS  
from MariaDB 10.5.1.
```

## See Also

- [MariaDB replication](#)
- [Replication threads](#)
- [SHOW PROCESSLIST](#). In

[SHOW PROCESSLIST](#)

output, replica threads are identified by  
Binlog Dump

## 1.1.2.5.10 RESET MASTER

```
RESET MASTER [TO #]
```

Deletes all [binary log](#) files listed in the index file, resets the binary log index file to be empty, and creates a new binary log file with a suffix of .000001.

If

TO #

is given, then the first new binary log file will start from number #.

This statement is for use only when the master is started for the first time, and should never be used if any slaves are actively [replicating](#) from the binary log.

### See Also

- The [PURGE BINARY LOGS](#) statement is intended for use in active replication.

## 1.1.2.6 Plugin SQL Statements

### 1.1.2.6.1 SHOW PLUGINS

#### Syntax

```
SHOW PLUGINS;
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

#### Description

`SHOW PLUGINS`  
displays information about installed [plugins](#). The  
Library  
column indicates the plugin library - if it is  
NULL  
, the plugin is built-in and cannot be uninstalled.

The

[PLUGINS](#)  
table in the  
[information\\_schema](#)  
database contains more detailed information.

For specific information about storage engines (a particular type of plugin), see the

[information\\_schema.ENGINES](#)  
table and the  
[SHOW ENGINES](#)  
statement.

#### Examples

```
SHOW PLUGINS;
+-----+-----+-----+-----+-----+
| Name      | Status   | Type     | Library   | License  |
+-----+-----+-----+-----+-----+
| binlog    | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| mysql_native_password | ACTIVE   | AUTHENTICATION | NULL      | GPL      |
| mysql_old_password | ACTIVE   | AUTHENTICATION | NULL      | GPL      |
| MRG_MyISAM | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| MyISAM    | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| CSV       | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| MEMORY    | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| FEDERATED | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| PERFORMANCE_SCHEMA | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| Aria      | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| InnoDB    | ACTIVE   | STORAGE ENGINE | NULL      | GPL      |
| INNODB_TRX | ACTIVE   | INFORMATION SCHEMA | NULL      | GPL      |
...
| INNODB_SYS_FOREIGN | ACTIVE   | INFORMATION SCHEMA | NULL      | GPL      |
| INNODB_SYS_FOREIGN_COLS | ACTIVE   | INFORMATION SCHEMA | NULL      | GPL      |
| SPHINX    | ACTIVE   | STORAGE ENGINE  | NULL      | GPL      |
| ARCHIVE   | ACTIVE   | STORAGE ENGINE  | NULL      | GPL      |
| BLACKHOLE  | ACTIVE   | STORAGE ENGINE  | NULL      | GPL      |
| FEEDBACK  | DISABLED | INFORMATION SCHEMA | NULL      | GPL      |
| partition | ACTIVE   | STORAGE ENGINE  | NULL      | GPL      |
| pam       | ACTIVE   | AUTHENTICATION  | auth_pam.so | GPL      |
+-----+-----+-----+-----+-----+
```

## See Also

- [List of Plugins](#)
- [Plugin Overview](#)
- [INFORMATION\\_SCHEMA.PLUGINS Table](#)
- [INSTALL PLUGIN](#)
- [INFORMATION\\_SCHEMA.ALL\\_PLUGINS Table](#) (all plugins, installed or not)
- [INSTALL SONAME](#)
- [UNINSTALL PLUGIN](#)
- [UNINSTALL SONAME](#)

### 1.1.2.6.2 SHOW PLUGINS SONAME

#### Syntax

```
SHOW PLUGINS SONAME { library | LIKE 'pattern' | WHERE expr };
```

#### Description

`SHOW PLUGINS SONAME`  
displays information about compiled-in and all server plugins in the

`plugin_dir`

directory, including plugins that haven't been installed.

#### Examples

```
SHOW PLUGINS SONAME 'ha_example.so';
+-----+-----+-----+-----+-----+
| Name      | Status   | Type     | Library   | License  |
+-----+-----+-----+-----+-----+
| EXAMPLE   | NOT INSTALLED | STORAGE ENGINE | ha_example.so | GPL      |
| UNUSABLE  | NOT INSTALLED | DAEMON      | ha_example.so | GPL      |
+-----+-----+-----+-----+-----+
```

There is also a corresponding

[information\\_schema](#)

table, called

#### [ALL\\_PLUGINS](#)

, which contains more complete information.

## 1.1.2.6.3 INSTALL PLUGIN

### Syntax

```
INSTALL PLUGIN [IF NOT EXISTS] plugin_name SONAME 'plugin_library'
```

### Contents

- 1. [Syntax](#)
- 2. [Description](#)
- 3. [IF NOT EXISTS](#)
- 4. [Examples](#)
- 5. [See Also](#)

### Description

This statement installs an individual [plugin](#) from the specified library. To install the whole library (which could be required), use [INSTALL SONAME](#). See also [Installing a Plugin](#).

`plugin_name`

is the name of the plugin as defined in the plugin declaration structure contained in the library file. Plugin names are not case sensitive. For maximal compatibility, plugin names should be limited to ASCII letters, digits, and underscore, because they are used in C source files, shell command lines, M4 and Bourne shell scripts, and SQL environments.

`plugin_library`

is the name of the shared library that contains the plugin code. The file name extension can be omitted (which makes the statement look the same on all architectures).

The shared library must be located in the plugin directory (that is, the directory named by the [plugin\\_dir](#) system variable). The library must be in the plugin directory itself, not in a subdirectory. By default,

```
plugin_dir  
is plugin directory under the directory named by the  
pkglibdir  
configuration variable, but it can be changed by setting the value of  
plugin_dir  
at server startup. For example, set its value in a  
my.cnf  
file:
```

```
[mysqld]  
plugin_dir=/path/to/plugin/directory
```

If the value of [plugin\\_dir](#) is a relative path name, it is taken to be relative to the MySQL base directory (the value of the [basedir](#) system variable).

```
INSTALL PLUGIN  
adds a line to the  
mysql.plugin  
table that describes the plugin. This table contains the plugin name and library file name.
```

```
INSTALL PLUGIN  
causes the server to read option (br/>  
my.cnf  
) files just as during server startup. This enables the plugin to pick up any relevant options from those files. It is possible to add plugin options to  
an option file even before loading a plugin (if the loose prefix is used). It is also possible to uninstall a plugin, edit  
my.cnf  
, and install the plugin again. Restarting the plugin this way enables it to the new option values without a server restart.
```

```
INSTALL PLUGIN  
also loads and initializes the plugin code to make the plugin available for use. A plugin is initialized by executing its initialization function, which
```

handles any setup that the plugin must perform before it can be used.

To use

```
INSTALL PLUGIN  
, you must have the INSERT privilege for the  
mysql.plugin  
table.
```

At server startup, the server loads and initializes any plugin that is listed in the

```
mysql.plugin  
table. This means that a plugin is installed with  
INSTALL PLUGIN  
only once, not every time the server starts. Plugin loading at startup does not occur if the server is started with the  
--skip-grant-tables  
option.
```

When the server shuts down, it executes the de-initialization function for each plugin that is loaded so that the plugin has a chance to perform any final cleanup.

If you need to load plugins for a single server startup when the

```
--skip-grant-tables  
option is given (which tells the server not to read system tables), use the  
--plugin-load  
mysqld option .
```

MariaDB starting with 10.4.0

## IF NOT EXISTS

When the

```
IF NOT EXISTS  
clause is used, MariaDB will return a note instead of an error if the specified plugin already exists. See SHOW WARNINGS .
```

## Examples

```
INSTALL PLUGIN sphinx SONAME 'ha_sphinx.so';
```

The extension can also be omitted:

```
INSTALL PLUGIN innodb SONAME 'ha_xtradb';
```

From MariaDB 10.4.0 :

```
INSTALL PLUGIN IF NOT EXISTS example SONAME 'ha_example';  
Query OK, 0 rows affected (0.104 sec)  
  
INSTALL PLUGIN IF NOT EXISTS example SONAME 'ha_example';  
Query OK, 0 rows affected, 1 warning (0.000 sec)  
  
SHOW WARNINGS;  
+-----+-----+  
| Level | Code | Message |  
+-----+-----+  
| Note  | 1968 | Plugin 'example' already installed |  
+-----+-----+
```

## See Also

- [List of Plugins](#)
- [Plugin Overview](#)
- [INFORMATION\\_SCHEMA.PLUGINS Table](#)
- [mysql\\_plugin](#)
- [SHOW PLUGINS](#)
- [INSTALL SONAME](#)
- [UNINSTALL PLUGIN](#)
- [UNINSTALL SONAME](#)

## 1.1.2.6.4 UNINSTALL PLUGIN

# Syntax

```
UNINSTALL PLUGIN [IF EXISTS] plugin_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
  - 1. [IF EXISTS](#)
3. [Examples](#)
4. [See Also](#)

## Description

This statement removes a single installed plugin . To uninstall the whole library which contains the plugin, use [UNINSTALL SONAME](#) . You cannot uninstall a plugin if any table that uses it is open.

plugin\_name

must be the name of some plugin that is listed in the [mysql.plugin](#) table. The server executes the plugin's deinitialization function and removes the row for the plugin from the

[mysql.plugin](#)

table, so that subsequent server restarts will not load and initialize the plugin.

[UNINSTALL PLUGIN](#)

does not remove the plugin's shared library file.

To use

[UNINSTALL PLUGIN](#)

, you must have the [DELETE](#) privilege for the [mysql.plugin](#) table.

MariaDB starting with 10.4.0

### IF EXISTS

If the

[IF EXISTS](#)

clause is used, MariaDB will return a note instead of an error if the plugin does not exist. See [SHOW WARNINGS](#) .

## Examples

```
UNINSTALL PLUGIN example;
```

From [MariaDB 10.4.0](#) :

```
UNINSTALL PLUGIN IF EXISTS example;
Query OK, 0 rows affected (0.099 sec)

UNINSTALL PLUGIN IF EXISTS example;
Query OK, 0 rows affected, 1 warning (0.000 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1305 | PLUGIN example does not exist |
+-----+-----+
```

## See Also

- [Plugin Overview](#)
- [mysql\\_plugin](#)
- [INSTALL PLUGIN](#)
- [List of Plugins](#)

## 1.1.2.6.5 INSTALL SONAME

### Syntax

```
INSTALL SONAME 'plugin_library'
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

This statement is a variant of [INSTALL PLUGIN](#). It installs **all** [plugins](#) from a given

`plugin_library`  
. See [INSTALL PLUGIN](#) for details.

`plugin_library`  
is the name of the shared library that contains the plugin code. The file name extension (for example,  
`libmyplugin.so`  
or  
`libmyplugin.dll`  
) can be omitted (which makes the statement look the same on all architectures).

The shared library must be located in the plugin directory (that is, the directory named by the

`plugin_dir`

system variable). The library must be in the plugin directory itself, not in a subdirectory. By default,  
`plugin_dir`  
is plugin directory under the directory named by the  
`pkglibdir`  
configuration variable, but it can be changed by setting the value of  
`plugin_dir`  
at server startup. For example, set its value in a  
`my.cnf`  
file:

```
[mysqld]
plugin_dir=/path/to/plugin/directory
```

If the value of `plugin_dir` is a relative path name, it is taken to be relative to the MySQL base directory (the value of the  
`basedir`  
system variable).

`INSTALL SONAME`  
adds one or more lines to the  
`mysql.plugin`  
table that describes the plugin. This table contains the plugin name and library file name.

`INSTALL SONAME`  
causes the server to read option (  
`my.cnf`  
) files just as during server startup. This enables the plugin to pick up any relevant options from those files. It is possible to add plugin options to  
an option file even before loading a plugin (if the loose prefix is used). It is also possible to uninstall a plugin, edit  
`my.cnf`  
, and install the plugin again. Restarting the plugin this way enables it to the new option values without a server restart.

`INSTALL SONAME`  
also loads and initializes the plugin code to make the plugin available for use. A plugin is initialized by executing its initialization function, which  
handles any setup that the plugin must perform before it can be used.

To use

`INSTALL SONAME`  
, you must have the [INSERT privilege](#) for the  
`mysql.plugin`  
table.

At server startup, the server loads and initializes any plugin that is listed in the  
`mysql.plugin`

table. This means that a plugin is installed with  
INSTALL SONAME  
only once, not every time the server starts. Plugin loading at startup does not occur if the server is started with the  
--skip-grant-tables  
option.

When the server shuts down, it executes the de-initialization function for each plugin that is loaded so that the plugin has a chance to perform any final cleanup.

If you need to load plugins for a single server startup when the  
--skip-grant-tables  
option is given (which tells the server not to read system tables), use the  
--plugin-load  
`mysql option`.

If you need to install only one plugin from a library, use the [INSTALL PLUGIN](#) statement.

## Examples

To load the XtraDB storage engine and all of its

information\_schema  
tables with one statement, use

```
INSTALL SONAME 'ha_xtradb';
```

This statement can be used instead of

INSTALL PLUGIN  
even when the library contains only one plugin:

```
INSTALL SONAME 'ha_sequence';
```

## See Also

- [List of Plugins](#)
- [Plugin Overview](#)
- [SHOW PLUGINS](#)
- [INSTALL PLUGIN](#)
- [UNINSTALL PLUGIN](#)
- [UNINSTALL SONAME](#)
- [SHOW PLUGINS](#)
- [INFORMATION\\_SCHEMA.PLUGINS Table](#)
- [mysql\\_plugin](#)

## 1.1.2.6.6 UNINSTALL SONAME

### Syntax

```
UNINSTALL SONAME [IF EXISTS] 'plugin_library'
```

### Contents

1. [Syntax](#)
2. [Description](#)
  - 1. [IF EXISTS](#)
3. [Examples](#)
4. [See Also](#)

### Description

This statement is a variant of [UNINSTALL PLUGIN](#) statement, that removes all [plugins](#) belonging to a specified `plugin_library`. See [UNINSTALL PLUGIN](#) for details.

`plugin_library` is the name of the shared library that contains the plugin code. The file name extension (for example, `libmyplugin.so` or `libmyplugin.dll`)

) can be omitted (which makes the statement look the same on all architectures).

To use

```
UNINSTALL SONAME  
, you must have the DELETE privilege for the  
mysql.plugin  
table.
```

MariaDB starting with 10.4.0

## IF EXISTS

If the

```
IF EXISTS  
clause is used, MariaDB will return a note instead of an error if the plugin library does not exist. See SHOW WARNINGS.
```

## Examples

To uninstall the XtraDB plugin and all of its

```
information_schema  
tables with one statement, use
```

```
UNINSTALL SONAME 'ha_xtradb';
```

From MariaDB 10.4.0 :

```
UNINSTALL SONAME IF EXISTS 'ha_example';  
Query OK, 0 rows affected (0.099 sec)  
  
UNINSTALL SONAME IF EXISTS 'ha_example';  
Query OK, 0 rows affected, 1 warning (0.000 sec)  
  
SHOW WARNINGS;  
+-----+-----+  
| Level | Code | Message |  
+-----+-----+  
| Note | 1305 | SONAME ha_example.so does not exist |  
+-----+-----+
```

## See Also

- [INSTALL SONAME](#)
- [SHOW PLUGINS](#)
- [INSTALL PLUGIN](#)
- [UNINSTALL PLUGIN](#)
- [SHOW PLUGINS](#)
- [INFORMATION\\_SCHEMA.PLUGINS Table](#)
- [mysql\\_plugin](#)
- [List of Plugins](#)

## 1.1.2.6.7 mysql\_plugin

MariaDB starting with 10.4.6

From MariaDB 10.4.6 ,

```
mariadb-plugin  
is a symlink to  
mysql_plugin
```

MariaDB starting with 10.5.2

From MariaDB 10.5.2 ,

```
mysql_plugin  
is the symlink, and  
mariadb-plugin  
the binary name.
```

## Contents

1. [Usage](#)
2. [Options](#)
3. [See Also](#)

`mysql_plugin`  
is a tool for enabling or disabling [plugins](#).

It is a commandline alternative to the [INSTALL PLUGIN](#) and [UNINSTALL PLUGIN](#) statements, and the  
`--plugin-load` option  
to [mysqld](#).

`mysql_plugin`  
must be run while the server is offline, and works by adding or removing rows from the [mysql.plugin](#) table.

## Usage

```
mysql_plugin [options] <plugin> ENABLE|DISABLE
```

`mysql_plugin`  
expects to find a configuration file that indicates how to configure the plugins. The configuration file is by default the same name as the plugin,  
with a  
.ini  
extension. For example:

```
mysql_plugin crazyplugins ENABLE
```

Here,

`mysql_plugin`  
will look for a file called  
crazyplugins.ini

```
crazyplugins
crazyplugin1
crazyplugin2
crazyplugin3
```

The first line should contain the name of the library object file, with no extension. The other lines list the names of the components. Each value should be  
on a separate line, and the

#  
character at the start of the line indicates a comment.

## Options

The following options can be specified on the command line, while some can be specified in the

[mysqld]  
group of any option file. For options specified in a  
[mysqld]  
group, only the  
`--basedir`  
,  
`--datadir`  
, and  
`--plugin-dir`  
options can be used - the rest are ignored.

Option	Description
<code>-b</code> , <code>--basedir=name</code>	The base directory for the server.

<pre>-d , --datadir=name</pre>	The data directory for the server.
<pre>-? , --help</pre>	Display help and exit.
<pre>-f , --my-print-defaults=name</pre>	Path to my_print_defaults executable. Example: /source/temp11/extra
<pre>-m , --mysqld=name</pre>	Path to mysqld executable. Example: /sbin/temp1/mysql/bin
<pre>-n , --no-defaults</pre>	Do not read values from configuration file.
<pre>-p , --plugin-dir=name</pre>	The plugin directory for the server.
<pre>-i , --plugin-ini=name</pre>	Read plugin information from configuration file specified instead of from <plugin-dir>/<plugin_name>.ini
<pre>-P , --print-defaults</pre>	Show default values from configuration file.
<pre>-v , --verbose</pre>	More verbose output; you can use this multiple times to get even more verbose output.
<pre>-V , --version</pre>	Output version information and exit.

## See Also

- [List of Plugins](#)
- [Plugin Overview](#)
- [INFORMATION\\_SCHEMA.PLUGINS Table](#)
- [INSTALL PLUGIN](#)
- [INSTALL SONAME](#)
- [UNINSTALL PLUGIN](#)
- [UNINSTALL SONAME](#)

## 1.1.2.7 SET Commands

### 1.1.2.7.1 SET

# Syntax

```
SET variable_assignment [, variable_assignment] ...
variable_assignment:
    user_var_name = expr
    | [GLOBAL | SESSION] system_var_name = expr
    | @@global. | @@session. | @@system_var_name = expr
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [GLOBAL / SESSION](#)
  2. [DEFAULT](#)
3. [Examples](#)
4. [See Also](#)

One can also set a user variable in any expression with this syntax:

```
user_var_name := expr
```

## Description

The

```
SET
```

statement assigns values to different types of variables that affect the operation of the server or your client. Older versions of MySQL employed `SET OPTION`, but this syntax was deprecated in favor of `SET` without `OPTION`, and was removed in [MariaDB 10.0](#).

Changing a system variable by using the `SET` statement does not make the change permanently. To do so, the change must be made in a [configuration file](#).

For setting variables on a per-query basis (from [MariaDB 10.1.2](#)), see

[SET STATEMENT](#)

See

[SHOW VARIABLES](#)

for documentation on viewing server system variables.

See [Server System Variables](#) for a list of all the system variables.

## GLOBAL / SESSION

When setting a system variable, the scope can be specified as either GLOBAL or SESSION.

A global variable change affects all new sessions. It does not affect any currently open sessions, including the one that made the change.

A session variable change affects the current session only.

If the variable has a session value, not specifying either GLOBAL or SESSION will be the same as specifying SESSION. If the variable only has a global value, not specifying GLOBAL or SESSION will apply to the change to the global value.

## DEFAULT

Setting a global variable to DEFAULT will restore it to the server default, and setting a session variable to DEFAULT will restore it to the current global value.

## Examples

- [innodb\\_sync\\_spin\\_loops](#) is a global variable.
- [skip\\_parallel\\_replication](#) is a session variable.
- [max\\_error\\_count](#) is both global and session.

```

SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME IN ('max_error_count', 'skip_parallel_replication', 'innodb_sync_spin_loops');
+-----+-----+-----+
| VARIABLE_NAME | SESSION_VALUE | GLOBAL_VALUE |
+-----+-----+-----+
| MAX_ERROR_COUNT | 64 | 64 |
| SKIP_PARALLEL_REPLICATION | OFF | NULL |
| INNODB_SYNC_SPIN_LOOPS | NULL | 30 |
+-----+-----+-----+

```

Setting the session values:

```

SET max_error_count=128;Query OK, 0 rows affected (0.000 sec)

SET skip_parallel_replication=ON;Query OK, 0 rows affected (0.000 sec)

SET innodb_sync_spin_loops=60;
ERROR 1229 (HY000): Variable 'innodb_sync_spin_loops' is a GLOBAL variable
and should be set with SET GLOBAL

```

```

SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME IN ('max_error_count', 'skip_parallel_replication', 'innodb_sync_spin_loops');
+-----+-----+-----+
| VARIABLE_NAME | SESSION_VALUE | GLOBAL_VALUE |
+-----+-----+-----+
| MAX_ERROR_COUNT | 128 | 64 |
| SKIP_PARALLEL_REPLICATION | ON | NULL |
| INNODB_SYNC_SPIN_LOOPS | NULL | 30 |
+-----+-----+-----+

```

Setting the global values:

```

SET GLOBAL max_error_count=256;

SET GLOBAL skip_parallel_replication=ON;
ERROR 1228 (HY000): Variable 'skip_parallel_replication' is a SESSION variable
and can't be used with SET GLOBAL

SET GLOBAL innodb_sync_spin_loops=120;

SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME IN ('max_error_count', 'skip_parallel_replication', 'innodb_sync_spin_loops');
+-----+-----+-----+
| VARIABLE_NAME | SESSION_VALUE | GLOBAL_VALUE |
+-----+-----+-----+
| MAX_ERROR_COUNT | 128 | 256 |
| SKIP_PARALLEL_REPLICATION | ON | NULL |
| INNODB_SYNC_SPIN_LOOPS | NULL | 120 |
+-----+-----+-----+

```

**SHOW VARIABLES** will by default return the session value unless the variable is global only.

```

SHOW VARIABLES LIKE 'max_error_count';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_error_count | 128 |
+-----+-----+

SHOW VARIABLES LIKE 'skip_parallel_replication';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| skip_parallel_replication | ON |
+-----+-----+

SHOW VARIABLES LIKE 'innodb_sync_spin_loops';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| innodb_sync_spin_loops | 120 |
+-----+-----+

```

Using the inplace syntax:

```

SELECT (@a:=1);
+-----+
| (@a:=1) |
+-----+
|   1   |
+-----+

SELECT @a;
+-----+
| @a   |
+-----+
|   1   |
+-----+

```

## See Also

- [Using last\\_value\(\)](#) to return data of used rows
- [SET STATEMENT](#)
- [SET Variable](#)
- [SET Data Type](#)
- [DECLARE Variable](#)

## 1.1.2.7.2 SET CHARACTER SET

### Syntax

```

SET {CHARACTER SET | CHARSET}
{charset_name | DEFAULT}

```

### Description

Sets the `character_set_client` and `character_set_results` session system variables to the specified character set and `collation_connection` to the value of `collation_database`, which implicitly sets `character_set_connection` to the value of `character_set_database`.

This maps all strings sent between the current client and the server with the given mapping.

### Example

```

SHOW VARIABLES LIKE 'character_set\%';
+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| character_set_client | utf8    |
| character_set_connection | utf8    |
| character_set_database | latin1  |
| character_set_filesystem | binary  |
| character_set_results | utf8    |
| character_set_server | latin1  |
| character_set_system | utf8    |
+-----+-----+

SHOW VARIABLES LIKE 'collation%';
+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| collation_connection | utf8_general_ci |
| collation_database | latin1_swedish_ci |
| collation_server | latin1_swedish_ci |
+-----+-----+

SET CHARACTER SET utf8mb4;

SHOW VARIABLES LIKE 'character_set\%';
+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| character_set_client | utf8mb4 |
| character_set_connection | latin1  |
| character_set_database | latin1  |
| character_set_filesystem | binary  |
| character_set_results | utf8mb4 |
| character_set_server | latin1  |
| character_set_system | utf8    |
+-----+-----+

SHOW VARIABLES LIKE 'collation%';
+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| collation_connection | latin1_swedish_ci |
| collation_database | latin1_swedish_ci |
| collation_server | latin1_swedish_ci |
+-----+-----+

```

## See Also

- [SET NAMES](#)

### 1.1.2.7.3 SET GLOBAL SQL\_SLAVE\_SKIP\_COUNTER

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [Multiple Replication Domains](#)
5. [See Also](#)

## Syntax

```
SET GLOBAL sql_slave_skip_counter = N
```

## Description

This statement skips the next

*N*

events from the master. This is useful for recovering from [replication](#) stops caused by a statement.

If multi-source replication is used, this statement applies to the default connection. It could be necessary to change the value of the

```
default_master_connection
```

server system variable.

Note that, if the event is a [transaction](#), the whole transaction will be skipped. With non-transactional engines, an event is always a single statement.

This statement is valid only when the slave threads are not running. Otherwise, it produces an error.

The statement does not automatically restart the slave threads.

## Example

```
SHOW SLAVE STATUS \G  
...  
SET GLOBAL sql_slave_skip_counter = 1;  
START SLAVE;
```

Multi-source replication:

```
SET @@default_master_connection = 'master_01';  
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;  
START SLAVE;
```

## Multiple Replication Domains

`sql_slave_skip_counter`  
can't be used to skip transactions on a slave if [GTID replication](#) is in use and if

```
gtid_slave_pos
```

contains multiple

```
gtid_domain_id
```

values. In that case, you'll get an error like the following:

```
ERROR 1966 (HY000): When using parallel replication and GTID with multiple  
replication domains, @@sql_slave_skip_counter can not be used. Instead,  
setting @@gtid_slave_pos explicitly can be used to skip to after a given GTID  
position.
```

In order to skip transactions in cases like this, you will have to manually change

```
gtid_slave_pos
```

## See Also

- [Selectively Skipping Replication of Binlog Events](#)

## 1.1.2.7.4 SET NAMES

### Syntax

```
SET NAMES {'charset_name'  
[COLLATE 'collation_name'] | DEFAULT}
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

# Description

Sets the `character_set_client`, `character_set_connection`, `character_set_results` and, implicitly, the `collation_connection` session system variables to the specified character set and collation.

This determines which `character set` the client will use to send statements to the server, and the server will use for sending results back to the client.

```
ucs2
,
utf16
, and
utf32
are not valid character sets for
SET NAMES
, as they cannot be used as client character sets.
```

The collation clause is optional. If not defined (or if

```
DEFAULT
is specified), the default collation for the character set will be used.
```

Quotes are optional for the character set or collation clauses.

## Examples

```

SELECT VARIABLE_NAME, SESSION_VALUE
  FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME LIKE 'character_set_c%' OR
VARIABLE_NAME LIKE 'character_set_re%' OR
VARIABLE_NAME LIKE 'collation_c%';
+-----+-----+
| VARIABLE_NAME | SESSION_VALUE |
+-----+-----+
| CHARACTER_SET_RESULTS | utf8 |
| CHARACTER_SET_CONNECTION | utf8 |
| CHARACTER_SET_CLIENT | utf8 |
| COLLATION_CONNECTION | utf8_general_ci |
+-----+-----+

SET NAMES big5;

SELECT VARIABLE_NAME, SESSION_VALUE
  FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME LIKE 'character_set_c%' OR
VARIABLE_NAME LIKE 'character_set_re%' OR
VARIABLE_NAME LIKE 'collation_c%';
+-----+-----+
| VARIABLE_NAME | SESSION_VALUE |
+-----+-----+
| CHARACTER_SET_RESULTS | big5 |
| CHARACTER_SET_CONNECTION | big5 |
| CHARACTER_SET_CLIENT | big5 |
| COLLATION_CONNECTION | big5_chinese_ci |
+-----+-----+

SET NAMES 'latin1' COLLATE 'latin1_bin';

SELECT VARIABLE_NAME, SESSION_VALUE
  FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME LIKE 'character_set_c%' OR
VARIABLE_NAME LIKE 'character_set_re%' OR
VARIABLE_NAME LIKE 'collation_c%';
+-----+-----+
| VARIABLE_NAME | SESSION_VALUE |
+-----+-----+
| CHARACTER_SET_RESULTS | latin1 |
| CHARACTER_SET_CONNECTION | latin1 |
| CHARACTER_SET_CLIENT | latin1 |
| COLLATION_CONNECTION | latin1_bin |
+-----+-----+

SET NAMES DEFAULT;

SELECT VARIABLE_NAME, SESSION_VALUE
  FROM INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME LIKE 'character_set_c%' OR
VARIABLE_NAME LIKE 'character_set_re%' OR
VARIABLE_NAME LIKE 'collation_c%';
+-----+-----+
| VARIABLE_NAME | SESSION_VALUE |
+-----+-----+
| CHARACTER_SET_RESULTS | latin1 |
| CHARACTER_SET_CONNECTION | latin1 |
| CHARACTER_SET_CLIENT | latin1 |
| COLLATION_CONNECTION | latin1_swedish_ci |
+-----+-----+

```

## See Also

- [SET CHARACTER SET](#)
- [Character Sets and Collations](#)

## 1.1.2.7.5 SET PASSWORD

### Syntax

```
SET PASSWORD [FOR user] =
{
    PASSWORD('some password')
    | OLD_PASSWORD('some password')
    | 'encrypted password'
}
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Authentication Plugin Support](#)
4. [Passwordless User Accounts](#)
5. [Example](#)
6. [See Also](#)

## Description

The

```
SET PASSWORD
```

statement assigns a password to an existing MariaDB user account.

If the password is specified using the

```
PASSWORD()
```

or

```
OLD_PASSWORD()
```

function, the literal text of the password should be given. If the password is specified without using either function, the password should be the already-encrypted password value as returned by

```
PASSWORD()
```

```
OLD_PASSWORD()
```

should only be used if your MariaDB/MySQL clients are very old (< 4.0.0).

With no

```
FOR
```

clause, this statement sets the password for the current user. Any client that has connected to the server using a non-anonymous account can change the password for that account.

With a

```
FOR
```

clause, this statement sets the password for a specific account on the current server host. Only clients that have the

```
UPDATE
```

privilege for the

```
mysql
```

database can do this. The user value should be given in

```
user_name@host_name
```

format, where

```
user_name
```

and

```
host_name
```

are exactly as they are listed in the User and Host columns of the

```
mysql.user
```

table entry.

The argument to

```
PASSWORD()
```

and the password given to MariaDB clients can be of arbitrary length.

# Authentication Plugin Support

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later,

```
SET PASSWORD  
(with or without  
PASSWORD()  
) works for accounts authenticated via any authentication plugin that supports passwords stored in the
```

`mysql.global_priv`

table.

The

`ed25519`

,

`mysql_native_password`

, and

`mysql_old_password`

authentication plugins store passwords in the

`mysql.global_priv`

table.

If you run

```
SET PASSWORD  
on an account that authenticates with one of these authentication plugins that stores passwords in the
```

`mysql.global_priv`

table, then the

`PASSWORD()`

function is evaluated by the specific authentication plugin used by the account. The authentication plugin hashes the password with a method that is compatible with that specific authentication plugin.

The

`unix_socket`

,

`named_pipe`

,

`gssapi`

, and

`pam`

authentication plugins do **not** store passwords in the

`mysql.global_priv`

table. These authentication plugins rely on other methods to authenticate the user.

If you attempt to run

```
SET PASSWORD  
on an account that authenticates with one of these authentication plugins that doesn't store a password in the
```

`mysql.global_priv`

table, then MariaDB Server will raise a warning like the following:

SET PASSWORD is ignored for users authenticating via unix\_socket plugin

See [Authentication from MariaDB 10.4](#) for an overview of authentication changes in [MariaDB 10.4](#).

#### MariaDB until 10.3

In [MariaDB 10.3](#) and before,

```
SET PASSWORD  
(with or without  
PASSWORD()  
) only works for accounts authenticated via
```

`mysql_native_password`

or

`mysql_old_password`

authentication plugins

## Passwordless User Accounts

User accounts do not always require passwords to login.

The

`unix_socket`

,

`named_pipe`

and

`gssapi`

authentication plugins do not require a password to authenticate the user.

The

`pam`

authentication plugin may or may not require a password to authenticate the user, depending on the specific configuration.

The

`mysql_native_password`

and

`mysql_old_password`

authentication plugins require passwords for authentication, but the password can be blank. In that case, no password is required.

If you provide a password while attempting to log into the server as an account that doesn't require a password, then MariaDB server will simply ignore the password.

#### MariaDB starting with 10.4

In [MariaDB 10.4](#) and later, a user account can be defined to use multiple authentication plugins in a specific order of preference. This specific scenario may be more noticeable in these versions, since an account could be associated with some authentication plugins that require a password, and some that do not.

## Example

For example, if you had an entry with User and Host column values of '

```
bob  
' and '  
.loc.gov
```

', you would write the statement like this:

```
SET PASSWORD FOR 'bob'@'%loc.gov' = PASSWORD('newpass');
```

If you want to delete a password for a user, you would do:

```
SET PASSWORD FOR 'bob'@localhost = PASSWORD("");
```

## See Also

- [Password Validation Plugins](#) - permits the setting of basic criteria for passwords
- [ALTER USER](#)

## 1.1.2.7.6 SET ROLE

### Syntax

```
SET ROLE { role | NONE }
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)

### Description

The

```
SET ROLE
```

statement enables a [role](#), along with all of its associated permissions, for the current session. To unset a role, use [NONE](#).

If a role that doesn't exist, or to which the user has not been assigned, is specified, an

```
ERROR 1959 (OP000): Invalid role specification
```

error occurs.

An automatic SET ROLE is implicitly performed when a user connects if that user has been assigned a default role. See [SET DEFAULT ROLE](#).

### Example

```
SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |
+-----+
| NULL         |
+-----+

SET ROLE staff;

SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |
+-----+
| staff        |
+-----+

SET ROLE NONE;

SELECT CURRENT_ROLE();
+-----+
| CURRENT_ROLE() |
+-----+
| NULL          |
+-----+
```

## 1.1.2.7.7 SET SQL\_LOG\_BIN

### Syntax

```
SET [SESSION] sql_log_bin = {0|1}
```

### Description

Sets the [sql\\_log\\_bin](#) system variable, which disables or enables [binary logging](#) for the current connection, if the client has the [SUPER](#) [privilege](#). The statement is refused with an error if the client does not have that privilege.

Before MariaDB 5.5 and before MySQL 5.6 one could also set  
sql\_log\_bin  
as a global variable. This has now been disabled as this was too dangerous as it could damage replication.

## 1.1.2.7.8 SET STATEMENT

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [Limitations](#)
5. [Source](#)

MariaDB starting with [10.1.2](#)

Per-query variables were introduced in [MariaDB 10.1.2](#)

SET STATEMENT

can be used to set the value of a system variable for the duration of the statement. It is also possible to set multiple variables.

### Syntax

```
SET STATEMENT var1=value1 [, var2=value2, ...]  
FOR <statement>
```

where

varN

is a system variable (list of allowed variables is provided below), and

valueN

is a constant literal.

### Description

```
SET STATEMENT var1=value1 FOR stmt
```

is roughly equivalent to

```
SET @save_value=@var1;  
SET SESSION var1=value1;  
stmt;  
SET SESSION var1=@save_value;
```

The server parses the whole statement before executing it, so any variables set in this fashion that affect the parser may not have the expected effect. Examples include the charset variables, sql\_mode=ansi\_quotes, etc.

### Examples

One can limit statement execution time

```
max_statement_time  
:  
SET STATEMENT max_statement_time=1000 FOR SELECT ... ;
```

One can switch on/off individual optimizations:

```
SET STATEMENT optimizer_switch='materialization=off' FOR SELECT ....;
```

It is possible to enable MRR/BKA for a query:

```
SET STATEMENT join_cache_level=6, optimizer_switch='mrr=on' FOR SELECT ...
```

Note that it makes no sense to try to set a session variable inside a

```
SET STATEMENT  
:  
#USELESS STATEMENT  
SET STATEMENT sort_buffer_size = 100000 for SET SESSION sort_buffer_size = 200000;
```

For the above, after setting sort\_buffer\_size to 200000 it will be reset to its original state (the state before the

```
SET STATEMENT  
started) after the statement execution.
```

## Limitations

There are a number of variables that cannot be set on per-query basis. These include:

- autocommit
- character\_set\_client
- character\_set\_connection
- character\_set\_filesystem
- collation\_connection
- default\_master\_connection
- debug\_sync
- interactive\_timeout
- gtid\_domain\_id
- last\_insert\_id
- log\_slow\_filter
- log\_slow\_rate\_limit
- log\_slow\_verbosity

- long\_query\_time
- min\_examined\_row\_limit
- profiling
- profiling\_history\_size
- query\_cache\_type
- rand\_seed1
- rand\_seed2
- skip\_replication
- slow\_query\_log
- sql\_log\_off
- tx\_isolation
- wait\_timeout

## Source

- The feature was originally implemented as a Google Summer of Code 2009 project by Joseph Lukas.
- Percona Server 5.6 included it as [Per-query variable statement](#)
- MariaDB ported the patch and fixed *many* bugs. The task in MariaDB Jira is [MDEV-5231](#) .

## 1.1.2.7.9 SET TRANSACTION

### Syntax

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_property [, transaction_property] ...

transaction_property:
    ISOLATION LEVEL level
    | READ WRITE
    | READ ONLY

level:
    REPEATABLE READ
    | READ COMMITTED
    | READ UNCOMMITTED
    | SERIALIZABLE
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [Isolation Level](#)
  2. [Isolation Levels](#)
    1. [READ UNCOMMITTED](#)
    2. [READ COMMITTED](#)
    3. [REPEATABLE READ](#)
    4. [SERIALIZABLE](#)
  3. [Access Mode](#)
  3. [Examples](#)

## Description

This statement sets the transaction isolation level or the transaction access mode globally, for the current session, or for the next transaction:

- With the

GLOBAL keyword, the statement sets the default transaction level globally for all subsequent sessions. Existing sessions are unaffected.

- With the

SESSION keyword, the statement sets the default transaction level for all subsequent transactions performed within the current session.

- Without any

SESSION  
or  
GLOBAL keyword, the statement sets the isolation level for the next (not started) transaction performed within the current session.

A change to the global default isolation level requires the

SUPER

privilege. Any session is free to change its session isolation level (even in the middle of a transaction), or the isolation level for its next transaction.

## Isolation Level

To set the global default isolation level at server startup, use the

--transaction-isolation=level

option on the command line or in an option file. Values of level for this option use dashes rather than spaces, so the allowable values are READ-UNCOMMITTED

,  
READ-COMMITTED

,  
REPEATABLE-READ  
, OR  
SERIALIZABLE

. For example, to set the default isolation level to REPEATABLE READ, use these lines in the [mysqld] section of an option file:

```
[mysqld]
transaction-isolation = REPEATABLE-READ
```

To determine the global and session transaction isolation levels at runtime, check the value of the

tx\_isolation

system variable:

```
SELECT @@GLOBAL.tx_isolation, @@tx_isolation;
```

InnoDB supports each of the transaction isolation levels described here using different locking strategies. The default level is

REPEATABLE READ

. For additional information about InnoDB record-level locks and how it uses them to execute various types of statements, see [InnoDB Lock Modes](#), and <http://dev.mysql.com/doc/refman/en/innodb-locks-set.html>.

## Isolation Levels

The following sections describe how MariaDB supports the different transaction levels.

### READ UNCOMMITTED

SELECT

statements are performed in a non-locking fashion, but a possible earlier version of a row might be used. Thus, using this isolation level, such reads are not consistent. This is also called a "dirty read." Otherwise, this isolation level works like

READ COMMITTED

### READ COMMITTED

A somewhat Oracle-like isolation level with respect to consistent (non-locking) reads: Each consistent read, even within the same transaction, sets and reads its own fresh snapshot. See <http://dev.mysql.com/doc/refman/en/innodb-consistent-read.html>.

For locking reads (

SELECT

with

FOR UPDATE

or

LOCK IN SHARE MODE

), InnoDB locks only index records, not the gaps before them, and thus allows the free insertion of new records next to locked records. For

UPDATE

and

DELETE

statements, locking depends on whether the statement uses a unique index with a unique search condition (such as

WHERE id = 100

), or a range-type search condition (such as

WHERE id > 100

). For a unique index with a unique search condition, InnoDB locks only the index record found, not the gap before it. For range-type searches, InnoDB locks the index range scanned, using gap locks or next-key (gap plus index-record) locks to block insertions by other sessions into the gaps covered by the range. This is necessary because "phantom rows" must be blocked for MySQL replication and recovery to work.

**Note:** If the

READ COMMITTED

isolation level is used or the `innodb_locks_unsafe_for_binlog` system variable is enabled, there is no InnoDB gap locking except for `foreign-key` constraint checking and duplicate-key checking. Also, record locks for non-matching rows are released after MariaDB has evaluated the

WHERE

condition. If you use

READ COMMITTED

or enable `innodb_locks_unsafe_for_binlog`, you must use row-based binary logging.

### REPEATABLE READ

This is the default isolation level for InnoDB. For consistent reads, there is an important difference from the

READ COMMITTED

isolation level: All consistent reads within the same transaction read the snapshot established by the first read. This convention means that if you issue several plain (non-locking)

SELECT

statements within the same transaction, these

SELECT

statements are consistent also with respect to each other. See <http://dev.mysql.com/doc/refman/en/innodb-consistent-read.html>.

For locking reads (SELECT with FOR UPDATE or LOCK IN SHARE MODE), UPDATE, and DELETE statements, locking depends on whether the statement uses a unique index with a unique search condition, or a range-type search condition. For a unique index with a unique search condition, InnoDB locks only the index record found, not the gap before it. For other search conditions, InnoDB locks the index range scanned, using gap locks or next-key (gap plus index-record) locks to block insertions by other sessions into the gaps covered by the range.

This is the minimum isolation level for non-distributed [XA transactions](#).

### SERIALIZABLE

This level is like REPEATABLE READ, but InnoDB implicitly converts all plain SELECT statements to

SELECT ... LOCK IN SHARE MODE

if

autocommit

is disabled. If autocommit is enabled, the SELECT is its own transaction. It therefore is known to be read only and can be serialized if performed as a consistent (non-locking) read and need not block for other transactions. (This means that to force a plain SELECT to block if other transactions have modified the selected rows, you should disable autocommit.)

Distributed [XA transactions](#) should always use this isolation level.

## Access Mode

The access mode specifies whether the transaction is allowed to write data or not. By default, transactions are in

READ WRITE  
mode (see the [tx\\_read\\_only](#) system variable).

READ ONLY

mode allows the storage engine to apply optimizations that cannot be used for transactions which write data. The only exception to this rule is that read only transactions can perform DDL statements on temporary tables.

It is not permitted to specify both

READ WRITE  
and  
READ ONLY  
in the same statement.

READ WRITE  
and  
READ ONLY  
can also be specified in the

[START TRANSACTION](#)

statement, in which case the specified mode is only valid for one transaction.

## Examples

```
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Attempting to set the isolation level within an existing transaction without specifying

GLOBAL  
or  
SESSION

```
START TRANSACTION;
```

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
ERROR 1568 (25001): Transaction characteristics can't be changed while a transaction is in progress
```

## 1.1.2.7.10 SET Variable

### Syntax

```
SET var_name = expr [, var_name = expr] ...
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

### Description

The

SET  
statement in [stored programs](#) is an extended version of the general

[SET](#)

statement. Referenced variables may be ones declared inside a stored program, global system variables, or user-defined variables.

The

```
SET  
statement in stored programs is implemented as part of the pre-existing SET syntax. This allows an extended syntax of  
SET a=x,  
b=y, ...
```

where different variable types (locally declared variables, global and session server variables, user-defined variables) can be mixed. This also allows combinations of local variables and some options that make sense only for system variables; in that case, the options are recognized but ignored.

```
SET  
can be used with both local variables and user-defined variables .
```

When setting several variables using the columns returned by a query,

```
SELECT INTO
```

should be preferred.

To set many variables to the same value, the

```
LAST_VALUE( )
```

function can be used.

Below is an example of how a user-defined variable may be set:

```
SET @x = 1;
```

## See Also

- [SET](#)
- [SET STATEMENT](#)
- [DECLARE Variable](#)

## 1.1.2.8 SHOW

### 1.1.2.8.1 About SHOW

```
SHOW
```

has many forms that provide information about databases, tables, columns, or status information about the server. These include:

- [SHOW AUTHORS](#)
- [SHOW CHARACTER SET \[like\\_or\\_where\]](#)
- [SHOW COLLATION \[like\\_or\\_where\]](#)
- [SHOW \[FULL\] COLUMNS FROM tbl\\_name \[FROM db\\_name\] \[like\\_or\\_where\]](#)
- [SHOW CONTRIBUTORS](#)
- [SHOW CREATE DATABASE db\\_name](#)
- [SHOW CREATE EVENT event\\_name](#)
- [SHOW CREATE PACKAGE package\\_name](#)
- [SHOW CREATE PACKAGE BODY package\\_name](#)
- [SHOW CREATE PROCEDURE proc\\_name](#)
- [SHOW CREATE TABLE tbl\\_name](#)
- [SHOW CREATE TRIGGER trigger\\_name](#)
- [SHOW CREATE VIEW view\\_name](#)
- [SHOW DATABASES \[like\\_or\\_where\]](#)
- [SHOW ENGINE engine\\_name {STATUS | MUTEX}](#)
- [SHOW \[STORAGE\] ENGINES](#)
- [SHOW ERRORS \[LIMIT \[offset,\] row\\_count\]](#)
- [SHOW \[FULL\] EVENTS](#)
- [SHOW FUNCTION CODE func\\_name](#)
- [SHOW FUNCTION STATUS \[like\\_or\\_where\]](#)
- [SHOW GRANTS FOR user](#)
- [SHOW INDEX FROM tbl\\_name \[FROM db\\_name\]](#)
- [SHOW INNODB STATUS](#)
- [SHOW OPEN TABLES \[FROM db\\_name\] \[like\\_or\\_where\]](#)
- [SHOW PLUGINS](#)
- [SHOW PROCEDURE CODE proc\\_name](#)
- [SHOW PROCEDURE STATUS \[like\\_or\\_where\]](#)
- [SHOW PRIVILEGES](#)
- [SHOW \[FULL\] PROCESSLIST](#)

- SHOW PROFILE [types] [FOR QUERY n] [OFFSET n] [LIMIT n]
- SHOW PROFILES
- SHOW [GLOBAL | SESSION] STATUS [like\_or\_where]
- SHOW TABLE STATUS [FROM db\_name] [like\_or\_where]
- SHOW TABLES [FROM db\_name] [like\_or\_where]
- SHOW TRIGGERS [FROM db\_name] [like\_or\_where]
- SHOW [GLOBAL | SESSION] VARIABLES [like\_or\_where]
- SHOW WARNINGS [LIMIT [offset,] row\_count]

```
like_or_where:
  LIKE 'pattern'
  | WHERE expr
```

If the syntax for a given

```
SHOW
statement includes a
LIKE 'pattern'
part,
'pattern'
is a string that can contain the SQL "
%
" and "
-
" wildcard characters. The pattern is useful for restricting statement output to matching values.
```

Several

```
SHOW
statements also accept a
WHERE
clause that provides more flexibility in specifying which rows to display. See Extended Show.
```

## 1.1.2.8.2 Extended Show

### Contents

1. [Examples](#)

The following

```
SHOW
```

```
statements can be extended by using a
WHERE
clause and a
LIKE
clause to refine the results:
```

- [SHOW CHARACTER SET](#)
- [SHOW COLLATION](#)
- [SHOW COLUMNS](#)
- [SHOW DATABASES](#)
- [SHOW FUNCTION STATUS](#)
- [SHOW INDEX](#)
- [SHOW OPEN TABLES](#)
- [SHOW PACKAGE STATUS](#)
- [SHOW PACKAGE BODY STATUS](#)
- [SHOW INDEX](#)
- [SHOW PROCEDURE STATUS](#)
- [SHOW STATUS](#)
- [SHOW TABLE STATUS](#)
- [SHOW TABLES](#)
- [SHOW TRIGGERS](#)
- [SHOW VARIABLES](#)

As with a regular

```
SELECT
```

```
, the
WHERE
clause can be used for the specific columns returned, and the
```

clause with the regular wildcards.

## Examples

```
SHOW TABLES;
+-----+
| Tables_in_test |
+-----+
| animal_count   |
| animals        |
| are_the_mooses_loose |
| aria_test2     |
| t1             |
| view1          |
+-----+
```

Showing the tables beginning with a only.

```
SHOW TABLES WHERE Tables_in_test LIKE 'a%';
+-----+
| Tables_in_test |
+-----+
| animal_count   |
| animals        |
| are_the_mooses_loose |
| aria_test2     |
+-----+
```

Variables whose name starts with *aria* and with a value of greater than 8192:

```
SHOW VARIABLES WHERE Variable_name LIKE 'aria%' AND Value >8192;
+-----+
| Variable_name | Value |
+-----+
| aria_checkpoint_log_activity | 1048576 |
| aria_log_file_size | 1073741824 |
| aria_max_sort_file_size | 9223372036853727232 |
| aria_pagecache_buffer_size | 134217728 |
| aria_sort_buffer_size | 134217728 |
+-----+
```

Shortcut, just returning variables whose name begins with *aria* .

```
SHOW VARIABLES LIKE 'aria%';
+-----+
| Variable_name | Value |
+-----+
| aria_block_size | 8192 |
| aria_checkpoint_interval | 30 |
| aria_checkpoint_log_activity | 1048576 |
| aria_force_start_after_recovery_failures | 0 |
| aria_group_commit | none |
| aria_group_commit_interval | 0 |
| aria_log_file_size | 1073741824 |
| aria_log_purge_type | immediate |
| aria_max_sort_file_size | 9223372036853727232 |
| aria_page_checksum | ON |
| aria_pagecache_age_threshold | 300 |
| aria_pagecache_buffer_size | 134217728 |
| aria_pagecache_division_limit | 100 |
| aria_recover | NORMAL |
| aria_repair_threads | 1 |
| aria_sort_buffer_size | 134217728 |
| aria_stats_method | nulls_unequal |
| aria_sync_log_dir | NEWFILE |
| aria_used_for_temp_tables | ON |
+-----+
```

## 1.1.2.8.3 SHOW AUTHORS

# Syntax

```
SHOW AUTHORS
```

## Description

The

```
SHOW AUTHORS
```

statement displays information about the people who work on MariaDB. For each author, it displays Name, Location, and Comment values. All columns are encoded as latin1.

These include:

- First the active people in MariaDB are listed.
- Then the active people in MySQL.
- Last the people that have contributed to MariaDB/MySQL in the past.

The order is somewhat related to importance of the contribution given to the MariaDB project, but this is not 100% accurate. There is still room for improvement and debate...

## Example

```
SHOW AUTHORS;
```

Name	Location	Comment
Michael (Monty) Widenius	Tusby, Finland	Lead developer and main author
Sergei Golubchik	Kerpen, Germany	Architect, Full-text search, precision math, plugin f
Igor Babaev	Bellevue, USA	Optimizer, keycache, core work
Sergey Petrunia	St. Petersburg, Russia	Optimizer
Oleksandr Byelkin	Lugansk, Ukraine	Query Cache (4.0), Subqueries (4.1), Views (5.0)
Timour Katchaounov	Sofia , Bulgaria	Optimizer
Kristian Nielsen	Copenhagen, Denmark	Replication, Async client protocol, General buildbo
Alexander (Bar) Barkov	Izhevsk, Russia	Unicode and character sets
Alexey Botchkov (Holyfoot)	Izhevsk, Russia	GIS extensions, embedded server, precision math
Daniel Bartholomew	Raleigh, USA	MariaDB documentation
Colin Charles	Selangor, Malesia	MariaDB documentation, talks at a LOT of conferences
Sergey Vojtovich	Izhevsk, Russia	initial implementation of plugin architecture, mainta
Vladislav Vaintroub	Mannheim, Germany	MariaDB Java connector, new thread pool, Windows opti
Elena Stepanova	Sankt Petersburg, Russia	QA, test cases
Georg Richter	Heidelberg, Germany	New LGPL C connector, PHP connector
Jan Lindström	Ylämylly, Finland	Working on InnoDB
Lixun Peng	Hangzhou, China	Multi Source replication
Percona	CA, USA	XtraDB, microslow patches, extensions to slow log
...		

## See Also

- [SHOW CONTRIBUTORS](#) . This list all members and sponsors of the MariaDB Foundation and other sponsors.

### 1.1.2.8.4 SHOW BINARY LOGS

## Syntax

```
SHOW BINARY LOGS  
SHOW MASTER LOGS
```

## Description

Lists the [binary log](#) files on the server. This statement is used as part of the procedure described in

```
PURGE BINARY LOGS
```

, that shows how to determine which logs can be purged.

This statement requires the [SUPER](#) privilege, the [REPLICATION\\_CLIENT](#) privilege, or, from [MariaDB 10.5.2](#) , the [BINLOG MONITOR](#) privilege.

## Examples

```
SHOW BINARY LOGS;
+-----+-----+
| Log_name | File_size |
+-----+-----+
| mariadb-bin.000001 | 19039 |
| mariadb-bin.000002 | 717389 |
| mariadb-bin.000003 | 300 |
| mariadb-bin.000004 | 333 |
| mariadb-bin.000005 | 899 |
| mariadb-bin.000006 | 125 |
| mariadb-bin.000007 | 18907 |
| mariadb-bin.000008 | 19530 |
| mariadb-bin.000009 | 151 |
| mariadb-bin.000010 | 151 |
| mariadb-bin.000011 | 125 |
| mariadb-bin.000012 | 151 |
| mariadb-bin.000013 | 151 |
| mariadb-bin.000014 | 125 |
| mariadb-bin.000015 | 151 |
| mariadb-bin.000016 | 314 |
+-----+-----+
```

### 1.1.2.8.5 SHOW BINLOG EVENTS

#### Syntax

```
SHOW BINLOG EVENTS
[IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]
```

#### Description

Shows the events in the [binary log](#). If you do not specify '

log\_name  
' , the first binary log is displayed.

Requires the [BINLOG MONITOR](#) privilege (>= MariaDB 10.5.2) or the [REPLICATION SLAVE](#) privilege (<= MariaDB 10.5.1).

#### Example

```
SHOW BINLOG EVENTS IN 'mysql_sandbox10019-bin.000002';
+-----+-----+-----+-----+-----+
| Log_name | Pos | Event_type | Server_id | End_log_pos | Info
+-----+-----+-----+-----+-----+
| mysql_sandbox10019-bin.000002 | 4 | Format_desc | 1 | 248 | Server ver: 10.0.19-MariaDB-log, Binlog ve
| mysql_sandbox10019-bin.000002 | 248 | Gtid_list | 1 | 273 | []
| mysql_sandbox10019-bin.000002 | 273 | Binlog_checkpoint | 1 | 325 | mysql_sandbox10019-bin.000002
| mysql_sandbox10019-bin.000002 | 325 | Gtid | 1 | 363 | GTID 0-1-1
| mysql_sandbox10019-bin.000002 | 363 | Query | 1 | 446 | CREATE DATABASE blog
| mysql_sandbox10019-bin.000002 | 446 | Gtid | 1 | 484 | GTID 0-1-2
| mysql_sandbox10019-bin.000002 | 484 | Query | 1 | 571 | use `blog`; CREATE TABLE bb (id INT)
```

### 1.1.2.8.6 SHOW CHARACTER SET

#### Syntax

```
SHOW CHARACTER SET
[LIKE 'pattern' | WHERE expr]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

```
SHOW CHARACTER SET
statement shows all available character sets . The
LIKE
clause, if present on its own, indicates which character set names to match. The
WHERE
and
LIKE
clauses can be given to select rows using more general conditions, as discussed in Extended SHOW .
```

The same information can be queried from the [Information Schema CHARACTER\\_SETS](#) table.

See [Setting Character Sets and Collations](#) for details on specifying the character set at the server, database, table and column levels.

## Examples

```
SHOW CHARACTER SET LIKE 'latin%';
+-----+-----+-----+
| Charset | Description          | Default collation | Maxlen |
+-----+-----+-----+
| latin1  | cp1252 West European   | latin1_swedish_ci |      1 |
| latin2  | ISO 8859-2 Central European | latin2_general_ci |      1 |
| latin5  | ISO 8859-9 Turkish       | latin5_turkish_ci |      1 |
| latin7  | ISO 8859-13 Baltic       | latin7_general_ci |      1 |
+-----+-----+-----+
```

```
SHOW CHARACTER SET WHERE Maxlen LIKE '2';
+-----+-----+-----+
| Charset | Description          | Default collation | Maxlen |
+-----+-----+-----+
| big5   | Big5 Traditional Chinese | big5_chinese_ci   |      2 |
| sjis   | Shift-JIS Japanese     | sjis_japanese_ci |      2 |
| euckr  | EUC-KR Korean          | euckr_korean_ci  |      2 |
| gb2312 | GB2312 Simplified Chinese | gb2312_chinese_ci |      2 |
| gbk    | GBK Simplified Chinese | gbk_chinese_ci   |      2 |
| ucs2   | UCS-2 Unicode           | ucs2_general_ci  |      2 |
| cp932  | SJIS for Windows Japanese | cp932_japanese_ci |      2 |
+-----+-----+-----+
```

## See Also

- [Supported Character Sets and Collations](#)
- [Setting Character Sets and Collations](#)
- [Information Schema CHARACTER\\_SETS](#)

## 1.1.2.8.7 SHOW CLIENT\_STATISTICS

### Syntax

```
SHOW CLIENT_STATISTICS
```

## Description

The

```
SHOW CLIENT_STATISTICS
statement is part of the User Statistics feature. It was removed as a separate statement in MariaDB 10.1.1 , but effectively replaced by the generic SHOW information\_schema\_table statement. The information\_schema.CLIENT\_STATISTICS table holds statistics about client connections.
```

The [userstat](#) system variable must be set to 1 to activate this feature. See the [User Statistics](#) and [information\\_schema.CLIENT\\_STATISTICS](#) articles for

more information.

## Example

```
SHOW CLIENT_STATISTICS
*****
1. row ****
Client: localhost
Total_connections: 35
Concurrent_connections: 0
Connected_time: 708
    Busy_time: 2.555797999999985
    Cpu_time: 0.0412374000000002
Bytes_received: 3883
Bytes_sent: 21595
Binlog_bytes_written: 0
    Rows_read: 18
    Rows_sent: 115
Rows_deleted: 0
Rows_inserted: 0
Rows_updated: 0
Select_commands: 70
Update_commands: 0
Other_commands: 0
Commit_transactions: 1
Rollback_transactions: 0
Denied_connections: 0
Lost_connections: 0
Access_denied: 0
Empty_queries: 35
```

### 1.1.2.8.8 SHOW COLUMNS

### 1.1.2.8.9 SHOW CONTRIBUTORS

#### Syntax

```
SHOW CONTRIBUTORS
```

#### Description

The

```
SHOW CONTRIBUTORS
```

statement displays information about the companies and people who financially contribute to MariaDB. For each contributor, it displays Name  
,  
Location  
, and  
Comment  
values. All columns are encoded as latin1

It displays all [members and sponsors of the MariaDB Foundation](#) as well as other financial contributors.

## Example

Name	Location	Comment
Booking.com	<a href="https://www.booking.com">https://www.booking.com</a>	Founding member, Platinum Sponsor of the MariaDB Foundation
Alibaba Cloud	<a href="https://www.alibabacloud.com/">https://www.alibabacloud.com/</a>	Platinum Sponsor of the MariaDB Foundation
Tencent Cloud	<a href="https://cloud.tencent.com">https://cloud.tencent.com</a>	Platinum Sponsor of the MariaDB Foundation
Microsoft	<a href="https://microsoft.com/">https://microsoft.com/</a>	Platinum Sponsor of the MariaDB Foundation
MariaDB Corporation	<a href="https://mariadb.com">https://mariadb.com</a>	Founding member, Platinum Sponsor of the MariaDB Foundation
Visma	<a href="https://visma.com">https://visma.com</a>	Gold Sponsor of the MariaDB Foundation
DBS	<a href="https://dbs.com">https://dbs.com</a>	Gold Sponsor of the MariaDB Foundation
IBM	<a href="https://www.ibm.com">https://www.ibm.com</a>	Gold Sponsor of the MariaDB Foundation
Tencent Games	<a href="http://game.qq.com/">http://game.qq.com/</a>	Gold Sponsor of the MariaDB Foundation
Nexedi	<a href="https://www.nexedi.com">https://www.nexedi.com</a>	Silver Sponsor of the MariaDB Foundation
Acronis	<a href="https://www.acronis.com">https://www.acronis.com</a>	Silver Sponsor of the MariaDB Foundation
Verkkokauppa.com	<a href="https://www.verkkokauppa.com">https://www.verkkokauppa.com</a>	Bronze Sponsor of the MariaDB Foundation
Virtuozzo	<a href="https://virtuozzo.com">https://virtuozzo.com</a>	Bronze Sponsor of the MariaDB Foundation
Tencent Game DBA	<a href="http://tencentdba.com/about">http://tencentdba.com/about</a>	Bronze Sponsor of the MariaDB Foundation
Tencent TDSQL	<a href="http://tdsql.org">http://tdsql.org</a>	Bronze Sponsor of the MariaDB Foundation
Percona	<a href="https://www.percona.com/">https://www.percona.com/</a>	Bronze Sponsor of the MariaDB Foundation
Google	USA	Sponsoring encryption, parallel replication and GTID
Facebook	USA	Sponsoring non-blocking API, LIMIT ROWS EXAMINED etc
Ronald Bradford	Brisbane, Australia	EFF contribution for UC2006 Auction
Sheeri Kritzer	Boston, Mass. USA	EFF contribution for UC2006 Auction
Mark Shuttleworth	London, UK.	EFF contribution for UC2006 Auction

## See Also

- [Log of MariaDB contributors](#) .
- [SHOW AUTHORS](#) list the authors of MariaDB (including documentation, QA etc).
- [MariaDB Foundation page on contributing financially](#)

## 1.1.2.8.10 SHOW CREATE DATABASE

### Syntax

```
SHOW CREATE {DATABASE | SCHEMA} db_name
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Shows the [CREATE DATABASE](#) statement that creates the given database.

```
SHOW CREATE SCHEMA
is a synonym for
SHOW CREATE DATABASE
.
SHOW CREATE DATABASE
quotes database names according to the value of the sql\_quote\_show\_create server system variable.
```

### Examples

```
SHOW CREATE DATABASE test;
+-----+
| Database | Create Database |
+-----+
| test      | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+-----+  
  
SHOW CREATE SCHEMA test;
+-----+
| Database | Create Database |
+-----+
| test      | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+-----+
```

With

## sql\_quote\_show\_create

off:

```
SHOW CREATE DATABASE test;
+-----+-----+
| Database | Create Database
+-----+-----+
| test      | CREATE DATABASE test /*!40100 DEFAULT CHARACTER SET latin1 */ |
+-----+-----+
```

With a comment, from MariaDB 10.5 :

```
SHOW CREATE DATABASE p;
+-----+-----+
| Database | Create Database |
+-----+-----+
| p       | CREATE DATABASE `p` /*!40100 DEFAULT CHARACTER SET latin1 */ COMMENT 'presentations' |
+-----+-----+
```

#### See Also

- CREATE DATABASE
  - ALTER DATABASE
  - Character Sets and Collations

## 1.1.2.8.11 SHOW CREATE EVENT

## Syntax

```
SHOW CREATE EVENT event_name
```

## Description

This statement displays the

CREATE EVENT

statement needed to re-create a given `event` , as well as the

SOL MODE

that was used when the trigger has been created and the character set used by the connection. To find out which events are present, use

#### [SHOW EVENTS](#)

The output of this statement is unreliablely affected by the

sql quote show create

server system variable - see <http://bugs.mysql.com/bug.php?id=12719>

The

`information_schema.EVENTS`

table provides similar, but more complete, information.

## Examples

```
SHOW CREATE EVENT test.e_daily\G
*****
1. row ****
Event: e_daily
sql_mode:
time_zone: SYSTEM
Create Event: CREATE EVENT `e_daily`
    ON SCHEDULE EVERY 1 DAY
    STARTS CURRENT_TIMESTAMP + INTERVAL 6 HOUR
    ON COMPLETION NOT PRESERVE
    ENABLE
    COMMENT 'Saves total number of sessions then
              clears the table each day'
    DO BEGIN
        INSERT INTO site_activity.totals (time, total)
        SELECT CURRENT_TIMESTAMP, COUNT(*)
        FROM site_activity.sessions;
        DELETE FROM site_activity.sessions;
    END
character_set_client: latin1
collation_connection: latin1_swedish_ci
Database Collation: latin1_swedish_ci
```

## See also

- [Events Overview](#)
- 

[CREATE EVENT](#)

•

[ALTER EVENT](#)

•

[DROP EVENT](#)

## 1.1.2.8.12 SHOW CREATE FUNCTION

### Syntax

```
SHOW CREATE FUNCTION func_name
```

### Description

This statement is similar to

`SHOW CREATE PROCEDURE`

but for [stored functions](#).

The output of this statement is unreliable affected by the

`sql_quote_show_create`

## Example

```
MariaDB [test]> SHOW CREATE FUNCTION VatCents\G
***** 1. row *****
    Function: VatCents
    sql_mode:
  Create Function: CREATE DEFINER='root'@`localhost` FUNCTION `VatCents`(price DECIMAL(10,2)) RETURNS int(11)
DETERMINISTIC
BEGIN
DECLARE x INT;
SET x = price * 114;
RETURN x;
END
character_set_client: utf8
collation_connection: utf8_general_ci
Database Collation: latin1_swedish_ci
```

## See also:

- [Stored Functions](#)
- 

[CREATE FUNCTION](#)

## 1.1.2.8.13 SHOW CREATE PACKAGE

MariaDB starting with [10.3.5](#)

Oracle-style packages were introduced in [MariaDB 10.3.5](#).

## Syntax

```
SHOW CREATE PACKAGE [ db_name . ] package_name
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

```
SHOW CREATE PACKAGE
```

statement can be used when [Oracle SQL\\_MODE](#) is set.

Shows the

```
CREATE
```

statement that creates the given package specification.

## Examples

```
SHOW CREATE PACKAGE employee_tools\G
*****
***** 1. row *****
  Package: employee_tools
  sql_mode: PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ORACLE,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPTIONS,NO_AUTO_C
  Create Package: CREATE DEFINER="root"@"localhost" PACKAGE "employee_tools" AS
    FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2);
    PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2));
    PROCEDURE raiseSalaryStd(eid INT);
    PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2));
  END
  character_set_client: utf8
  collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

## See Also

- [CREATE PACKAGE](#)
- [DROP PACKAGE](#)
- [CREATE PACKAGE BODY](#)
- [SHOW CREATE PACKAGE BODY](#)
- [DROP PACKAGE BODY](#)
- [Oracle SQL\\_MODE](#)

## 1.1.2.8.14 SHOW CREATE PACKAGE BODY

MariaDB starting with [10.3.5](#)

Oracle-style packages were introduced in [MariaDB 10.3.5](#).

## Syntax

```
SHOW CREATE PACKAGE BODY [ db_name . ] package_name
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See also](#)

## Description

The

```
SHOW CREATE PACKAGE BODY
```

statement can be used when [Oracle SQL\\_MODE](#) is set.

Shows the

```
CREATE
```

statement that creates the given package body (i.e. the implementation).

## Examples

```

SHOW CREATE PACKAGE BODY employee_tools\G
***** 1. row *****
  Package body: employee_tools
    sql_mode: PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ORACLE,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPTIONS,NO_AUTO_C
Create Package Body: CREATE DEFINER="root"@"localhost" PACKAGE BODY "employee_tools" AS

stdRaiseAmount DECIMAL(10,2):=500;

PROCEDURE log (eid INT, ecmnt TEXT) AS
BEGIN
  INSERT INTO employee_log (id, cmnt) VALUES (eid, ecmnt);
END;

PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2)) AS
  eid INT;
BEGIN
  INSERT INTO employee (name, salary) VALUES (ename, esalary);
  eid:= last_insert_id();
  log(eid, 'hire ' || ename);
END;

FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2) AS
  nSalary DECIMAL(10,2);
BEGIN
  SELECT salary INTO nSalary FROM employee WHERE id=eid;
  log(eid, 'getSalary id=' || eid || ' salary=' || nSalary);
  RETURN nSalary;
END;

PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2)) AS
BEGIN
  UPDATE employee SET salary=salary+amount WHERE id=eid;
  log(eid, 'raiseSalary id=' || eid || ' amount=' || amount);
END;

PROCEDURE raiseSalaryStd(eid INT) AS
BEGIN
  raiseSalary(eid, stdRaiseAmount);
  log(eid, 'raiseSalaryStd id=' || eid);
END;

BEGIN
  log(0, 'Session ' || connection_id() || ' ' || current_user || ' started');
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci

```

## See also

- [CREATE PACKAGE](#)
- [SHOW CREATE PACKAGE](#)
- [DROP PACKAGE](#)
- [CREATE PACKAGE BODY](#)
- [DROP PACKAGE BODY](#)
- [Oracle SQL\\_MODE](#)

## 1.1.2.8.15 SHOW CREATE PROCEDURE

### Syntax

```
SHOW CREATE PROCEDURE proc_name
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

This statement is a MariaDB extension. It returns the exact string that can be used to re-create the named `stored procedure`, as well as the

`SQL_MODE`

that was used when the trigger has been created and the character set used by the connection.. A similar statement,

`SHOW CREATE FUNCTION`

, displays information about `stored functions`.

Both statements require that you are the owner of the routine or have the

`SELECT`

privilege on the

`mysql.proc`

table. When neither is true, the statements display

NULL  
for the  
Create Procedure  
or  
Create Function  
field.

#### Warning Users with

`SELECT`  
privileges on

`mysql.proc`

or  
`USAGE`  
privileges on  
`*.*`

can view the text of routines, even when they do not have privileges for the function or procedure itself.

The output of these statements is unreliablely affected by the

`sql_quote_show_create`

server system variable - see <http://bugs.mysql.com/bug.php?id=12719>

## Examples

Here's a comparison of the

`SHOW CREATE PROCEDURE`  
and

`SHOW CREATE FUNCTION`

statements.

```
SHOW CREATE PROCEDURE test.simpleproc\G
*****
***** 1. row *****
Procedure: simpleproc
sql_mode:
Create Procedure: CREATE PROCEDURE `simpleproc`(OUT param1 INT)
    BEGIN
        SELECT COUNT(*) INTO param1 FROM t;
    END
character_set_client: latin1
collation_connection: latin1_swedish_ci
Database Collation: latin1_swedish_ci

SHOW CREATE FUNCTION test.hello\G
*****
***** 1. row *****
Function: hello
sql_mode:
Create Function: CREATE FUNCTION `hello`(s CHAR(20))
    RETURNS CHAR(50)
    RETURN CONCAT('Hello, ',s,'!')
character_set_client: latin1
collation_connection: latin1_swedish_ci
Database Collation: latin1_swedish_ci
```

When the user issuing the statement does not have privileges on the routine, attempting to

`CALL`

the procedure raises Error 1370.

```
CALL test.prc1();
Error 1370 (42000): execute command denied to user 'test_user'@'localhost' for routine 'test'.'prc1'
```

If the user neither has privilege to the routine nor the

`SELECT`

privilege on

`mysql.proc`

table, it raises Error 1305, informing them that the procedure does not exist.

```
SHOW CREATE TABLES test.prc1\G
Error 1305 (42000): PROCEDURE prc1 does not exist
```

## See Also

- [Stored Procedure Overview](#)
- [CREATE PROCEDURE](#)
- [ALTER PROCEDURE](#)
- [DROP PROCEDURE](#)
- [SHOW PROCEDURE STATUS](#)
- [Stored Routine Privileges](#)
- [Information Schema ROUTINES Table](#)

## 1.1.2.8.16 SHOW CREATE SEQUENCE

MariaDB starting with [10.3.1](#)

Sequences were introduced in [MariaDB 10.3](#).

## Syntax

```
SHOW CREATE SEQUENCE sequence_name;
```

## Contents

1. Syntax
2. Description
3. Example
4. Notes
5. See Also

## Description

Shows the `CREATE SEQUENCE` statement that created the given sequence. The statement requires the

`SELECT`

privilege for the table.

## Example

```
CREATE SEQUENCE s1 START WITH 50;
SHOW CREATE SEQUENCE s1\G;
***** 1. row *****
Table: s1
Create Table: CREATE SEQUENCE `s1` start with 50 minvalue 1 maxvalue 9223372036854775806
increment by 1 cache 1000 nocycle ENGINE=InnoDB
```

## Notes

If you want to see the underlying table structure used for the

`SEQUENCE`

you can use `SHOW CREATE TABLE` on the

`SEQUENCE`

. You can also use

`SELECT`

to read the current recorded state of the

`SEQUENCE`

:

```
SHOW CREATE TABLE s1\G
***** 1. row *****
Table: s1
Create Table: CREATE TABLE `s1` (
`next_not_cached_value` bigint(21) NOT NULL,
`minimum_value` bigint(21) NOT NULL,
`maximum_value` bigint(21) NOT NULL,
`start_value` bigint(21) NOT NULL COMMENT 'start value when sequences is created
or value if RESTART is used',
`increment` bigint(21) NOT NULL COMMENT 'increment value',
`cache_size` bigint(21) unsigned NOT NULL,
`cycle_option` tinyint(1) unsigned NOT NULL COMMENT '0 if no cycles are allowed,
1 if the sequence should begin a new cycle when maximum_value is passed',
`cycle_count` bigint(21) NOT NULL COMMENT 'How many cycles have been done'
) ENGINE=InnoDB SEQUENCE=1

SELECT * FROM s1\G
***** 1. row *****
next_not_cached_value: 50
minimum_value: 1
maximum_value: 9223372036854775806
start_value: 50
increment: 1
cache_size: 1000
cycle_option: 0
cycle_count: 0
```

## See Also

- [CREATE SEQUENCE](#)
- [ALTER SEQUENCE](#)

## 1.1.2.8.17 SHOW CREATE TABLE

## 1.1.2.8.18 SHOW CREATE TRIGGER

### Syntax

```
SHOW CREATE TRIGGER trigger_name
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See also](#)

### Description

This statement shows a

`CREATE TRIGGER`

statement that creates the given trigger, as well as the

`SQL_MODE`

that was used when the trigger has been created and the character set used by the connection.

The output of this statement is unreliablely affected by the

`sql_quote_show_create`

server system variable - see <http://bugs.mysql.com/bug.php?id=12719>

### Examples

```
SHOW CREATE TRIGGER example\G
***** 1. row *****
Trigger: example
sql_mode: ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,STRICT_ALL_TABLES
,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_
ENGINE_SUBSTITUTION
SQL Original Statement: CREATE DEFINER=`root`@`localhost` TRIGGER example BEFORE
INSERT ON t FOR EACH ROW
BEGIN
    SET NEW.c = NEW.c * 2;
END
character_set_client: cp850
collation_connection: cp850_general_ci
Database Collation: utf8_general_ci
Created: 2016-09-29 13:53:34.35
```

MariaDB starting with 10.2.3

The

Created

column was added in MySQL 5.7 and MariaDB 10.2.3 as part of introducing multiple trigger events per action.

### See also

- [Trigger Overview](#)
- 

`CREATE TRIGGER`

•

`DROP TRIGGER`

information\_schema.TRIGGERS Table

SHOW TRIGGERS

- Trigger Limitations

## 1.1.2.8.19 SHOW CREATE USER

## 1.1.2.8.20 SHOW CREATE VIEW

### Syntax

```
SHOW CREATE VIEW view_name
```

### Description

This statement shows a

CREATE VIEW

statement that creates the given [view](#) , as well as the character set used by the connection when the view was created. This statement also works with views.

```
SHOW CREATE VIEW
quotes table, column and stored function names according to the value of the
sql_quote_show_create
server system variable.
```

### Examples

```
SHOW CREATE VIEW example\G
***** 1. row *****
View: example
Create View: CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL
SECURITY DEFINER VIEW `example` AS (select `t`.`id` AS `id`,`t`.`s` AS `s` from
`t`)
character_set_client: cp850
collation_connection: cp850_general_ci
```

With

sql\_quote\_show\_create

off:

```
SHOW CREATE VIEW example\G
***** 1. row *****
View: example
Create View: CREATE ALGORITHM=UNDEFINED DEFINER=root@localhost SQL SECU
RITY DEFINER VIEW example AS (select t.id AS id,t.s AS s from t)
character_set_client: cp850
collation_connection: cp850_general_ci
```

## 1.1.2.8.21 SHOW DATABASES

# Syntax

```
SHOW {DATABASES | SCHEMAS}
      [LIKE 'pattern' | WHERE expr]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

`SHOW DATABASES`

lists the databases on the MariaDB server host.

`SHOW SCHEMAS`

is a synonym for

`SHOW DATABASES`

. The

`LIKE`

clause, if present on its own, indicates which database names to match. The

`WHERE`

and

`LIKE`

clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

You see only those databases for which you have some kind of privilege, unless you have the global `SHOW DATABASES` privilege. You can also get this list using the `mysqlshow` command.

If the server was started with the

`--skip-show-database`

option, you cannot use this statement at all unless you have the `SHOW DATABASES` privilege.

The list of results returned by

`SHOW DATABASES`

is based on directories in the data directory, which is how MariaDB implements databases. It's possible that output includes directories that do not correspond to actual databases.

The [Information Schema SCHEMATA table](#) also contains database information.

## Examples

```
SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test |
+-----+
```

```
SHOW DATABASES LIKE 'm%';
+-----+
| Database (m%) |
+-----+
| mysql |
+-----+
```

## See Also

- [CREATE DATABASE](#)
- [ALTER DATABASE](#)
- [DROP DATABASE](#)
- [SHOW CREATE DATABASE](#)
- [Character Sets and Collations](#)
- [Information Schema SCHEMATA Table](#)

## 1.1.2.8.22 SHOW ENGINE

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [SHOW ENGINE INNODB STATUS](#)
  2. [SHOW ENGINE INNODB MUTEX](#)
  3. [SHOW ENGINE PERFORMANCE\\_SCHEMA STATUS](#)
  4. [SHOW ENGINE ROCKSDB STATUS](#)

### Syntax

```
SHOW ENGINE engine_name {STATUS | MUTEX}
```

### Description

SHOW ENGINE

displays operational information about a storage engine. The following statements currently are supported:

```
SHOW ENGINE INNODB STATUS  
SHOW ENGINE INNODB MUTEX  
SHOW ENGINE PERFORMANCE_SCHEMA STATUS  
SHOW ENGINE ROCKSDB STATUS
```

If the [Sphinx Storage Engine](#) is installed, the following is also supported:

```
SHOW ENGINE SPHINX STATUS
```

See

[SHOW ENGINE SPHINX STATUS](#)

Older (and now removed) synonyms were

```
SHOW INNODB STATUS  
for  
SHOW ENGINE INNODB STATUS  
and  
SHOW MUTEX STATUS  
for  
SHOW ENGINE INNODB MUTEX
```

### SHOW ENGINE INNODB STATUS

[SHOW ENGINE INNODB STATUS](#)

displays extensive information from the standard InnoDB Monitor about the state of the InnoDB storage engine. See

[SHOW ENGINE INNODB STATUS](#)

for more.

### SHOW ENGINE INNODB MUTEX

[SHOW ENGINE INNODB MUTEX](#)

displays InnoDB mutex statistics.

The statement displays the following output fields:

- **Type:** Always InnoDB.
- **Name:** The source file where the mutex is implemented, and the line number in the file where the mutex is created. The line number is dependent on the MariaDB version.
- **Status:** This field displays the following values if

```

UNIV_DEBUG
was defined at compilation time (for example, in include/univ.h in the InnoDB part of the source tree). Only the
os_waits
value is displayed if
UNIV_DEBUG
was not defined. Without
UNIV_DEBUG
, the information on which the output is based is insufficient to distinguish regular mutexes and mutexes that protect rw-locks (which allow
multiple readers or a single writer). Consequently, the output may appear to contain multiple rows for the same mutex.


- count indicates how many times the mutex was requested.
- spin_waits indicates how many times the spinlock had to run.
- spin_rounds indicates the number of spinlock rounds. (spin_rounds divided by spin_waits provides the average round count.)
- os_waits indicates the number of operating system waits. This occurs when the spinlock did not work (the mutex was not locked during the
spinlock and it was necessary to yield to the operating system and wait).
- os_yields indicates the number of times a the thread trying to lock a mutex gave up its timeslice and yielded to the operating system (on the
presumption that allowing other threads to run will free the mutex so that it can be locked).
- os_wait_times indicates the amount of time (in ms) spent in operating system waits, if the timed_mutexes system variable is 1 (ON). If
timed_mutexes is 0 (OFF), timing is disabled, so os_wait_times is 0. timed_mutexes is off by default.

```

Information from this statement can be used to diagnose system problems. For example, large values of spin\_waits and spin\_rounds may indicate scalability problems.

The

[information\\_schema](#)

[INNODB\\_MUTEXES](#)

table provides similar information.

## SHOW ENGINE PERFORMANCE\_SCHEMA STATUS

This statement shows how much memory is used for

[performance\\_schema](#)

tables and internal buffers.

The output contains the following fields:

- **Type:** Always  
[performance\\_schema](#)
- **Name:** The name of a table, the name of an internal buffer, or the  
[performance\\_schema](#)  
word, followed by a dot and an attribute. Internal buffers names are enclosed by parenthesis.  
[performance\\_schema](#)  
means that the attribute refers to the whole database (it is a total).
- **Status:** The value for the attribute.

The following attributes are shown, in this order, for all tables:

- **row\_size:** The memory used for an individual record. This value will never change.
- **row\_count:** The number of rows in the table or buffer. For some tables, this value depends on a server system variable.
- **memory:** For tables and  
[performance\\_schema](#)  
, this is the result of  
[row\\_size](#)  
\*  
[row\\_count](#)

For internal buffers, the attributes are:

- count
- size

## SHOW ENGINE ROCKSDB STATUS

See also [MyRocks Performance Troubleshooting](#)

## 1.1.2.8.23 SHOW ENGINE INNODB STATUS

```
SHOW ENGINE INNODB STATUS
```

is a specific form of the [SHOW ENGINE](#) statement that displays the [InnoDB Monitor](#) output, which is extensive InnoDB information which can be useful in diagnosing problems.

The following sections are displayed

- **Status:** Shows the timestamp, monitor name and the number of seconds, or the elapsed time between the current time and the time the InnoDB Monitor output was last displayed. The per-second averages are based upon this time.
- **BACKGROUND THREAD:** `srv_master_thread` lines show work performed by the main background thread.
- **SEMAPHORES:** Threads waiting for a semaphore and stats on how the number of times threads have needed a spin or a wait on a mutex or rw-lock semaphore. If this number of threads is large, there may be I/O or contention issues. Reducing the size of the [innodb\\_thread\\_concurrency](#) system variable may help if contention is related to thread scheduling.
  - Spin rounds per wait
    - shows the number of spinlock rounds per OS wait for a mutex.
- **LATEST FOREIGN KEY ERROR:** Only shown if there has been a foreign key constraint error, it displays the failed statement and information about the constraint and the related tables.
- **LATEST DETECTED DEADLOCK:** Only shown if there has been a deadlock, it displays the transactions involved in the deadlock and the statements being executed, held and required locked and the transaction rolled back to.
- **TRANSACTIONS:** The output of this section can help identify lock contention, as well as reasons for the deadlocks.
- **FILE I/O:** InnoDB thread information as well as pending I/O operations and I/O performance statistics.
- **INSERT BUFFER AND ADAPTIVE HASH INDEX:** InnoDB insert buffer (old name for the [change buffer](#)) and adaptive hash index status information, including the number of each type of operation performed, and adaptive hash index performance.
- **LOG:** InnoDB log information, including current log sequence number, how far the log has been flushed to disk, the position at which InnoDB last took a checkpoint, pending writes and write performance statistics.
- **BUFFER POOL AND MEMORY:** Information on buffer pool pages read and written, which allows you to see the number of data file I/O operations performed by your queries. See [InnoDB Buffer Pool](#) for more. Similar information is also available from the [INFORMATION\\_SCHEMA.INNODB\\_BUFFER\\_POOL\\_STATS](#) table.
- **ROW OPERATIONS:** Information about the main thread, including the number and performance rate for each type of row operation.

If the [innodb\\_status\\_output\\_locks](#) system variable is set to

```
1
```

, extended lock information will be displayed.

Example output:

```
=====
2019-09-06 12:44:13 0x7f93cc236700 INNODB MONITOR OUTPUT
=====
Per second averages calculated from the last 4 seconds
-----
BACKGROUND THREAD
-----
srv_master_thread loops: 2 srv_active, 0 srv_shutdown, 83698 srv_idle
srv_master_thread log flush and writes: 83682
-----
SEMAPHORES
-----
OS WAIT ARRAY INFO: reservation count 15
OS WAIT ARRAY INFO: signal count 8
RW-shared spins 0, rounds 20, OS waits 7
RW-excl spins 0, rounds 0, OS waits 0
RW-sx spins 0, rounds 0, OS waits 0
Spin rounds per wait: 20.00 RW-shared, 0.00 RW-excl, 0.00 RW-sx
-----
TRANSACTIONS
-----
Trx id counter 236
Purge done for trx's n:o < 236 undo n:o < 0 state: running
History list length 22
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 421747401994584, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
---TRANSACTION 421747401990328, not started
0 lock struct(s), heap size 1136, 0 row lock(s)
-----
FILE I/O
-----
I/O thread 0 state: waiting for completed aio requests (insert buffer thread)
I/O thread 1 state: waiting for completed aio requests (log thread)
I/O thread 2 state: waiting for completed aio requests (read thread)
I/O thread 3 state: waiting for completed aio requests (read thread)
I/O thread 4 state: waiting for completed aio requests (read thread)
I/O thread 5 state: waiting for completed aio requests (read thread)
I/O thread 6 state: waiting for completed aio requests (write thread)
I/O thread 7 state: waiting for completed aio requests (write thread)
I/O thread 8 state: waiting for completed aio requests (write thread)
```

```

I/O thread 9 state: waiting for completed aio requests (write thread)
Pending normal aio reads: [0, 0, 0, 0] , aio writes: [0, 0, 0, 0] ,
  ibuf aio reads:, log i/o's:, sync i/o's:
Pending flushes (fsync) log: 0; buffer pool: 0
286 OS file reads, 171 OS file writes, 22 OS fsyncs
0.00 reads/s, 0 avg bytes/read, 0.00 writes/s, 0.00 fsyncs/s
-----
INSERT BUFFER AND ADAPTIVE HASH INDEX
-----
Ibuf: size 1, free list len 0, seg size 2, 0 merges
merged operations:
  insert 0, delete mark 0, delete 0
discarded operations:
  insert 0, delete mark 0, delete 0
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
Hash table size 34679, node heap has 0 buffer(s)
0.00 hash searches/s, 0.00 non-hash searches/s
---
LOG
---
Log sequence number 445926
Log flushed up to 445926
Pages flushed up to 445926
Last checkpoint at 445917
0 pending log flushes, 0 pending chkp writes
18 log i/o's done, 0.00 log i/o's/second
-----
BUFFER POOL AND MEMORY
-----
Total large memory allocated 167772160
Dictionary memory allocated 50768
Buffer pool size 8012
Free buffers 7611
Database pages 401
Old database pages 0
Modified db pages 0
Percent of dirty pages(LRU & free pages): 0.000
Max dirty pages percent: 75.000
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 264, created 137, written 156
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
No buffer pool page gets since the last printout
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 401, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
-----
ROW OPERATIONS
-----
0 queries inside InnoDB, 0 queries in queue
0 read views open inside InnoDB
Process ID=4267, Main thread ID=140272021272320, state: sleeping
Number of rows inserted 1, updated 0, deleted 0, read 1
0.00 inserts/s, 0.00 updates/s, 0.00 deletes/s, 0.00 reads/s
Number of system rows inserted 0, updated 0, deleted 0, read 0
0.00 inserts/s, 0.00 updates/s, 0.00 deletes/s, 0.00 reads/s
-----
END OF INNODB MONITOR OUTPUT
=====

```

## 1.1.2.8.24 SHOW ENGINES

### Syntax

```
SHOW [STORAGE] ENGINES
```

# Description

`SHOW ENGINES`

displays status information about the server's storage engines. This is particularly useful for checking whether a storage engine is supported, or to see what the default engine is.

`SHOW TABLE TYPES`

is a deprecated synonym.

The

`information_schema.ENGINES`

table provides the same information.

Since storage engines are plugins, different information about them is also shown in the

`information_schema.PLUGINS`

table and by the

`SHOW PLUGINS`

statement.

Note that both MySQL's InnoDB and Percona's XtraDB replacement are labeled as

InnoDB

. However, if XtraDB is in use, it will be specified in the

COMMENT

field. See [XtraDB and InnoDB](#) . The same applies to [FederatedX](#) .

The output consists of the following columns:

- Engine  
indicates the engine's name.
- Support  
indicates whether the engine is installed, and whether it is the default engine for the current session.
- Comment  
is a brief description.
- Transactions  
,  
XA  
and  
Savepoints  
indicate whether `transactions` , `XA transactions` and `transaction savepoints` are supported by the engine.

## Examples

```

SHOW ENGINES\G
***** 1. row *****
  Engine: InnoDB
  Support: DEFAULT
  Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
    XA: YES
  Savepoints: YES
***** 2. row *****
  Engine: CSV
  Support: YES
  Comment: CSV storage engine
Transactions: NO
    XA: NO
  Savepoints: NO
***** 3. row *****
  Engine: MyISAM
  Support: YES
  Comment: MyISAM storage engine
Transactions: NO
    XA: NO
  Savepoints: NO
***** 4. row *****
  Engine: BLACKHOLE
  Support: YES
  Comment: /dev/null storage engine (anything you write to it disappears)
Transactions: NO
    XA: NO
  Savepoints: NO
***** 5. row *****
  Engine: FEDERATED
  Support: YES
  Comment: FederatedX pluggable storage engine
Transactions: YES
    XA: NO
  Savepoints: YES
***** 6. row *****
  Engine: MRG_MyISAM
  Support: YES
  Comment: Collection of identical MyISAM tables
Transactions: NO
    XA: NO
  Savepoints: NO
***** 7. row *****
  Engine: ARCHIVE
  Support: YES
  Comment: Archive storage engine
Transactions: NO
    XA: NO
  Savepoints: NO
***** 8. row *****
  Engine: MEMORY
  Support: YES
  Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
    XA: NO
  Savepoints: NO
***** 9. row *****
  Engine: PERFORMANCE_SCHEMA
  Support: YES
  Comment: Performance Schema
Transactions: NO
    XA: NO
  Savepoints: NO
***** 10. row *****
  Engine: Aria
  Support: YES
  Comment: Crash-safe tables with MyISAM heritage
Transactions: NO
    XA: NO
  Savepoints: NO
10 rows in set (0.00 sec)

```

## 1.1.2.8.25 SHOW ERRORS

# Syntax

```
SHOW ERRORS [LIMIT [offset,] row_count]
SHOW ERRORS [LIMIT row_count OFFSET offset]
SHOW COUNT(*) ERRORS
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

## Description

This statement is similar to [SHOW WARNINGS](#), except that instead of displaying errors, warnings, and notes, it displays only errors.

The

```
LIMIT
clause has the same syntax as for the SELECT statement.
```

The

```
SHOW COUNT(*) ERRORS
statement displays the number of errors. You can also retrieve this number from the error\_count variable.
```

```
SHOW COUNT(*) ERRORS;
SELECT @@error_count;
```

The value of [error\\_count](#) might be greater than the number of messages displayed by [SHOW WARNINGS](#) if the [max\\_error\\_count](#) system variable is set so low that not all messages are stored.

For a list of MariaDB error codes, see [MariaDB Error Codes](#).

## Examples

```
SELECT f();
ERROR 1305 (42000): FUNCTION f does not exist
```

```
SHOW COUNT(*) ERRORS;
+-----+
| @@session.error_count |
+-----+
| 1 |
+-----+

SHOW ERRORS;
+-----+
| Level | Code | Message           |
+-----+
| Error | 1305 | FUNCTION f does not exist |
+-----+
```

## 1.1.2.8.26 SHOW EVENTS

### Syntax

```
SHOW EVENTS [{FROM | IN} schema_name]
[LIKE 'pattern' | WHERE expr]
```

## Description

Shows information about Event Manager [events](#) (created with

[CREATE EVENT](#)

). Requires the

[EVENT](#)

privilege. Without any arguments,  
SHOW EVENTS  
lists all of the events in the current schema:

```
SELECT CURRENT_USER(), SCHEMA();
+-----+-----+
| CURRENT_USER() | SCHEMA() |
+-----+-----+
| jon@ghidora   | myschema  |
+-----+-----+

SHOW EVENTS\G
***** 1. row *****
          Db: myschema
          Name: e_daily
      Definer: jon@ghidora
    Time zone: SYSTEM
        Type: RECURRING
   Execute at: NULL
Interval value: 10
 Interval field: SECOND
      Starts: 2006-02-09 10:41:23
        Ends: NULL
      Status: ENABLED
Originator: 0
character_set_client: latin1
collation_connection: latin1_swedish_ci
 Database Collation: latin1_swedish_ci
```

To see the event action, use

```
SHOW CREATE EVENT
```

instead, or look at the

```
information_schema.EVENTS
```

table.

To see events for a specific schema, use the

```
FROM
```

clause. For example, to see events for the test schema, use the following statement:

```
SHOW EVENTS FROM test;
```

The

```
LIKE
```

clause, if present, indicates which event names to match. The

```
WHERE
```

clause can be given to select rows using more general conditions, as discussed in [Extended Show](#).

## 1.1.2.8.27 SHOW FUNCTION STATUS

### Syntax

```
SHOW FUNCTION STATUS
[LIKE 'pattern' | WHERE expr]
```

### Description

This statement is similar to

```
SHOW PROCEDURE STATUS
```

but for [stored functions](#).

The LIKE clause, if present on its own, indicates which function names to match.

The

WHERE  
and  
LIKE  
clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#) .

The

```
information_schema.ROUTINES
```

table contains more detailed information.

## Examples

Showing all stored functions:

```
SHOW FUNCTION STATUS\G
***** 1. row *****
    Db: test
    Name: VatCents
    Type: FUNCTION
    Definer: root@localhost
    Modified: 2013-06-01 12:40:31
    Created: 2013-06-01 12:40:31
    Security_type: DEFINER
    Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
    Database Collation: latin1_swedish_ci
```

Stored functions whose name starts with 'V':

```
SHOW FUNCTION STATUS LIKE 'V%'\G
***** 1. row *****
    Db: test
    Name: VatCents
    Type: FUNCTION
    Definer: root@localhost
    Modified: 2013-06-01 12:40:31
    Created: 2013-06-01 12:40:31
    Security_type: DEFINER
    Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
    Database Collation: latin1_swedish_ci
```

Stored functions with a security type of 'DEFINER':

```
SHOW FUNCTION STATUS WHERE Security_type LIKE 'DEFINER'\G
***** 1. row *****
    Db: test
    Name: VatCents
    Type: FUNCTION
    Definer: root@localhost
    Modified: 2013-06-01 12:40:31
    Created: 2013-06-01 12:40:31
    Security_type: DEFINER
    Comment:
character_set_client: utf8
collation_connection: utf8_general_ci
    Database Collation: latin1_swedish_ci
```

## 1.1.2.8.28 SHOW GRANTS

## 1.1.2.8.29 SHOW INDEX

## 1.1.2.8.30 SHOW INDEX\_STATISTICS

### Syntax

```
SHOW INDEX_STATISTICS
```

## Description

The

```
SHOW INDEX_STATISTICS
```

statement was introduced in MariaDB 5.2 as part of the [User Statistics](#) feature. It was removed as a separate statement in MariaDB 10.1.1 , but effectively replaced by the generic [SHOW information\\_schema\\_table](#) statement. The [information\\_schmea.INDEX\\_STATISTICS](#) table shows statistics on index usage and makes it possible to do such things as locating unused indexes and generating the commands to remove them.

The [userstat](#) system variable must be set to 1 to activate this feature. See the [User Statistics](#) and [information\\_schema.INDEX\\_STATISTICS](#) table for more information.

## Example

```
SHOW INDEX_STATISTICS;
+-----+-----+-----+
| Table_schema | Table_name      | Index_name | Rows_read |
+-----+-----+-----+
| test        | employees_example | PRIMARY    |      1     |
+-----+-----+-----+
```

### 1.1.2.8.32 SHOW LOCALES

```
SHOW LOCALES
```

was introduced as part of the [Information Schema plugin extension](#) .

```
SHOW LOCALES
```

is used to return

```
locales
```

information as part of the [Locales](#) plugin. While the

```
information_schema.LOCALES
```

table has 8 columns, the

```
SHOW LOCALES
```

statement will only display 4 of them:

## Example

```
SHOW LOCALES;
+-----+-----+-----+
| Id   | Name      | Description          | Error_Message_Language |
+-----+-----+-----+
| 0    | en_US     | English - United States | english                |
| 1    | en_GB     | English - United Kingdom | english                |
| 2    | ja_JP     | Japanese - Japan     | japanese               |
| 3    | sv_SE     | Swedish - Sweden     | swedish               |
...
```

### 1.1.2.8.33 SHOW BINLOG STATUS

### 1.1.2.8.34 SHOW OPEN TABLES

## Syntax

```
SHOW OPEN TABLES [FROM db_name]
[LIKE 'pattern' | WHERE expr]
```

## Description

SHOW OPEN TABLES  
lists the non-  
TEMPORARY  
tables that are currently open in the table cache. See <http://dev.mysql.com/doc/refman/5.1/en/table-cache.html>.

The  
FROM  
and  
LIKE  
clauses may be used.

The  
FROM  
clause, if present, restricts the tables shown to those present in the  
db\_name  
database.

The  
LIKE  
clause, if present on its own, indicates which table names to match. The  
WHERE  
and  
LIKE  
clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

The following information is returned:

Column	Description
Database	Database name.
Name	Table name.
In_use	Number of table instances being used.
Name_locked	1 if the table is name-locked, e.g. if it is being dropped or renamed, otherwise 0

Before [MariaDB 5.5](#), each use of, for example, `LOCK TABLE ... WRITE` would increment  
`In_use`  
for that table. With the implementation of the metadata locking improvements in [MariaDB 5.5](#),  
`LOCK TABLE... WRITE`  
acquires a strong MDL lock, and concurrent connections will wait on this MDL lock, so any subsequent  
`LOCK TABLE... WRITE`  
will not increment  
`In_use`

## Example

```
SHOW OPEN TABLES;
+-----+-----+-----+
| Database | Table | In_use | Name_locked |
+-----+-----+-----+
...
| test    | xjson   |     0 |      0 |
| test    | jauthor |     0 |      0 |
| test    | locks   |     1 |      0 |
...
+-----+-----+-----+
```

### 1.1.2.8.35 SHOW PACKAGE BODY STATUS

MariaDB starting with [10.3.5](#)

Oracle-style packages were introduced in [MariaDB 10.3.5](#).

# Syntax

```
SHOW PACKAGE BODY STATUS  
[LIKE 'pattern' | WHERE expr]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

```
SHOW PACKAGE BODY STATUS
```

statement returns characteristics of stored package bodies (implementations), such as the database, name, type, creator, creation and modification dates, and character set information. A similar statement,

```
SHOW PACKAGE STATUS
```

, displays information about stored package specifications.

The

LIKE

clause, if present, indicates which package names to match. The

WHERE

and

LIKE

clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

The [ROUTINES table](#) in the INFORMATION\_SCHEMA database contains more detailed information.

## Examples

```
SHOW PACKAGE BODY STATUS LIKE 'pkg1'\G  
*****  
1. row *****  
      Db: test  
      Name: pkg1  
      Type: PACKAGE BODY  
      Definer: root@localhost  
      Modified: 2018-02-27 14:44:14  
      Created: 2018-02-27 14:44:14  
      Security_type: DEFINER  
      Comment: This is my first package body  
character_set_client: utf8  
collation_connection: utf8_general_ci  
Database Collation: latin1_swedish_ci
```

## See Also

- [SHOW PACKAGE STATUS](#)
- [SHOW CREATE PACKAGE BODY](#)
- [CREATE PACKAGE BODY](#)
- [DROP PACKAGE BODY](#)
- [Oracle SQL\\_MODE](#)

## 1.1.2.8.36 SHOW PACKAGE STATUS

MariaDB starting with [10.3.5](#)

Oracle-style packages were introduced in [MariaDB 10.3.5](#).

## Syntax

```
SHOW PACKAGE STATUS  
[LIKE 'pattern' | WHERE expr]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

```
SHOW PACKAGE STATUS
```

statement returns characteristics of stored package specifications, such as the database, name, type, creator, creation and modification dates, and character set information. A similar statement,

```
SHOW PACKAGE BODY STATUS
```

, displays information about stored package bodies (i.e. implementations).

The

LIKE

clause, if present, indicates which package names to match. The

WHERE

and

LIKE

clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

The [ROUTINES table](#) in the INFORMATION\_SCHEMA database contains more detailed information.

## Examples

```
SHOW PACKAGE STATUS LIKE 'pkg1'\G
*****
Db: test
Name: pkg1
Type: PACKAGE
Definer: root@localhost
Modified: 2018-02-27 14:38:15
Created: 2018-02-27 14:38:15
Security_type: DEFINER
Comment: This is my first package
character_set_client: utf8
collation_connection: utf8_general_ci
Database Collation: latin1_swedish_ci
```

## See Also

- [SHOW PACKAGE BODY](#)
- [SHOW CREATE PACKAGE](#)
- [CREATE PACKAGE](#)
- [DROP PACKAGE](#)
- [Oracle SQL\\_MODE](#)

### 1.1.2.8.37 SHOW PLUGINS

### 1.1.2.8.38 SHOW PLUGINS SONAME

### 1.1.2.8.39 SHOW PRIVILEGES

## Syntax

```
SHOW PRIVILEGES
```

## Description

SHOW PRIVILEGES

shows the list of [system privileges](#) that the MariaDB server supports. The exact list of privileges depends on the version of your server.

Note that before [MariaDB 10.3.23](#), [MariaDB 10.4.13](#) and [MariaDB 10.5.2](#), the [Delete history](#) privilege displays as

Delete versioning rows  
( [MDEV-20382](#) ).

## Example

From [MariaDB 10.5.9](#)

Privilege	Context	Comment
Alter	Tables	To alter the table
Alter routine	Functions,Procedures	To alter or drop stored functions/procedures
Create	Databases,Tables,Indexes	To create new databases and tables
Create routine	Databases	To use CREATE FUNCTION/PROCEDURE
Create temporary tables	Databases	To use CREATE TEMPORARY TABLE
Create view	Tables	To create new views
Create user	Server Admin	To create new users
Delete	Tables	To delete existing rows
Delete history	Tables	To delete versioning table historical rows
Drop	Databases,Tables	To drop databases, tables, and views
Event	Server Admin	To create, alter, drop and execute events
Execute	Functions,Procedures	To execute stored routines
File	File access on server	To read and write files on the server
Grant option	Databases,Tables,Functions,Procedures	To give to other users those privileges you possess
Index	Tables	To create or drop indexes
Insert	Tables	To insert data into tables
Lock tables	Databases	To use LOCK TABLES (together with SELECT privilege)
Process	Server Admin	To view the plain text of currently executing queries
Proxy	Server Admin	To make proxy user possible
References	Databases,Tables	To have references on tables
Reload	Server Admin	To reload or refresh tables, logs and privileges
Binlog admin	Server	To purge binary logs
Binlog monitor	Server	To use SHOW BINLOG STATUS and SHOW BINARY LOG
Binlog replay	Server	To use BINLOG (generated by mariadb-binlog)
Replication master admin	Server	To monitor connected slaves
Replication slave admin	Server	To start/stop slave and apply binlog events
Slave monitor	Server	To use SHOW SLAVE STATUS and SHOW RELAYLOG EVENTS
Replication slave	Server Admin	To read binary log events from the master
Select	Tables	To retrieve rows from table
Show databases	Server Admin	To see all databases with SHOW DATABASES
Show view	Tables	To see views with SHOW CREATE VIEW
Shutdown	Server Admin	To shut down the server
Super	Server Admin	To use KILL thread, SET GLOBAL, CHANGE MASTER, etc.
Trigger	Tables	To use triggers
Create tablespace	Server Admin	To create/alter/drop tablespaces
Update	Tables	To update existing rows
Set user	Server	To create views and stored routines with a different define
Federated admin	Server	To execute the CREATE SERVER, ALTER SERVER, DROP SERVER sta
Connection admin	Server	To bypass connection limits and kill other users' connectio
Read_only admin	Server	To perform write operations even if @@read_only=ON
Usage	Server Admin	No privileges - allow connect only

41 rows in set (0.000 sec)

## See Also

- [SHOW CREATE USER](#) shows how the user was created.
- [SHOW GRANTS](#) shows the

GRANTS/PRIVILEGES  
for a user.

## 1.1.2.8.40 SHOW PROCEDURE CODE

### Syntax

```
SHOW PROCEDURE CODE proc_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

This statement is a MariaDB extension that is available only for servers that have been built with debugging support. It displays a representation of the internal implementation of the named [stored procedure](#). A similar statement,

```
SHOW FUNCTION CODE
```

, displays information about [stored functions](#).

Both statements require that you be the owner of the routine or have

```
SELECT
```

access to the

```
mysql.proc
```

table.

If the named routine is available, each statement produces a result set. Each row in the result set corresponds to one "instruction" in the routine. The first column is Pos, which is an ordinal number beginning with 0. The second column is Instruction, which contains an SQL statement (usually changed from the original source), or a directive which has meaning only to the stored-routine handler.

## Examples

```
DELIMITER //  
  
CREATE PROCEDURE p1 ()  
BEGIN  
    DECLARE fanta INT DEFAULT 55;  
    DROP TABLE t2;  
    LOOP  
        INSERT INTO t3 VALUES (fanta);  
    END LOOP;  
END//  
Query OK, 0 rows affected (0.00 sec)  
  
SHOW PROCEDURE CODE p1//  
+-----+  
| Pos | Instruction          |  
+-----+  
| 0  | set fanta@0 55      |  
| 1  | stmt 9 "DROP TABLE t2"  
| 2  | stmt 5 "INSERT INTO t3 VALUES (fanta)" |  
| 3  | jump 2                |  
+-----+
```

## See Also

- [Stored Procedure Overview](#)
- [CREATE PROCEDURE](#)
- [ALTER PROCEDURE](#)
- [DROP PROCEDURE](#)
- [SHOW CREATE PROCEDURE](#)
- [SHOW PROCEDURE STATUS](#)
- [Stored Routine Privileges](#)
- [Information Schema ROUTINES Table](#)

## 1.1.2.8.41 SHOW PROCEDURE STATUS

### Syntax

```
SHOW PROCEDURE STATUS  
[LIKE 'pattern' | WHERE expr]
```

## Description

This statement is a MariaDB extension. It returns characteristics of a stored procedure, such as the database, name, type, creator, creation and modification dates, and character set information. A similar statement,

```
SHOW FUNCTION STATUS
```

, displays information about stored functions.

The

LIKE clause, if present, indicates which procedure or function names to match. The WHERE and LIKE clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

The [ROUTINES table](#) in the INFORMATION\_SCHEMA database contains more detailed information.

## Examples

```
SHOW PROCEDURE STATUS LIKE 'p1'\G  
***** 1. row *****  
      Db: test  
      Name: p1  
      Type: PROCEDURE  
      Definer: root@localhost  
      Modified: 2010-08-23 13:23:03  
      Created: 2010-08-23 13:23:03  
      Security_type: DEFINER  
      Comment:  
character_set_client: latin1  
collation_connection: latin1_swedish_ci  
      Database Collation: latin1_swedish_ci
```

## See Also

- [Stored Procedure Overview](#)
- [CREATE PROCEDURE](#)
- [ALTER PROCEDURE](#)
- [DROP PROCEDURE](#)
- [SHOW CREATE PROCEDURE](#)
- [Stored Routine Privileges](#)
- [Information Schema ROUTINES Table](#)

## 1.1.2.8.42 SHOW PROCESSLIST

### Syntax

```
SHOW [FULL] PROCESSLIST
```

## Description

```
SHOW PROCESSLIST
```

shows you which threads are running. You can also get this information from the [information\\_schema.PROCESSLIST](#) table or the [mysqladmin processlist](#) command. If you have the

```
PROCESS privilege
```

, you can see all threads. Otherwise, you can see only your own threads (that is, threads associated with the MariaDB account that you are using). If you do not use the

```
FULL
```

keyword, only the first 100 characters of each statement are shown in the Info field.

The columns shown in

SHOW PROCESSLIST

are:

Name	Description
ID	The client's process ID.
USER	The username associated with the process.
HOST	The host the client is connected to.
DB	The default database of the process (NULL if no default).
COMMAND	The command type. See <a href="#">Thread Command Values</a> .
TIME	The amount of time, in seconds, the process has been in its current state. For a replica SQL thread before MariaDB 10.1, this is the time in seconds between the last replicated event's timestamp and the replica machine's real time.
STATE	See <a href="#">Thread States</a> .
INFO	The statement being executed.
PROGRESS	The total progress of the process (0-100%) (see <a href="#">Progress Reporting</a> ).

See

TIME\_MS  
column in [information\\_schema.PROCESSLIST](#) for differences in the  
TIME  
column between MariaDB and MySQL.

The [information\\_schema.PROCESSLIST](#) table contains the following additional columns:

Name	Description
TIME_MS	The amount of time, in milliseconds, the process has been in its current state.
STAGE	The stage the process is currently in.
MAX_STAGE	The maximum number of stages.
PROGRESS	The progress of the process within the current stage (0-100%).

MEMORY_USED	The amount of memory used by the process.
EXAMINED_ROWS	The number of rows the process has examined.
QUERY_ID	Query ID.

Note that the

```
PROGRESS
field from the information schema, and the
PROGRESS
field from
SHOW PROCESSLIST
display different results.
SHOW PROCESSLIST
shows the total progress, while the information schema shows the progress for the current stage only.
```

Threads can be killed using their `thread_id` or their `query_id`, with the `KILL` statement.

Since queries on this table are locking, if the `performance_schema` is enabled, you may want to query the `THREADS` table instead.

## Examples

```
SHOW PROCESSLIST;
+-----+-----+-----+-----+-----+-----+
| Id | User      | Host     | db    | Command | Time       | Info          | Progress |
+-----+-----+-----+-----+-----+-----+
| 2 | event_scheduler | localhost | NULL | Daemon | 2693 | Waiting on empty queue | NULL | 0.000 |
| 4 | root        | localhost | NULL | Query   | 0 | Table lock | SHOW PROCESSLIST | 0.000 |
+-----+-----+-----+-----+-----+-----+
```

## See also

[CONNECTION\\_ID\(\)](#)

### 1.1.2.8.43 SHOW PROFILES

#### Syntax

```
SHOW PROFILES
```

#### Description

The

```
SHOW PROFILES
```

statement displays profiling information that indicates resource usage for statements executed during the course of the current session. It is used together with

```
SHOW PROFILE
```

### 1.1.2.8.44 SHOW QUERY\_RESPONSE\_TIME

It is possible to use

```
SHOW QUERY_RESPONSE_TIME
```

as an alternative for retrieving information from the `QUERY_RESPONSE_TIME` plugin.

This was introduced as part of the [Information Schema plugin extension](#).

### 1.1.2.8.45 SHOW RELAYLOG EVENTS

## 1.1.2.8.46 SHOW REPLICHOSTS

## 1.1.2.8.47 SHOW REPLICA STATUS

## 1.1.2.8.48 SHOW STATUS

### Syntax

```
SHOW [GLOBAL | SESSION] STATUS  
[LIKE 'pattern' | WHERE expr]
```

### Description

SHOW STATUS

provides server status information. This information also can be obtained using the

`mysqladmin extended-status`

command, or by querying the [Information Schema GLOBAL\\_STATUS and SESSION\\_STATUS tables](#). The

LIKE

clause, if present, indicates which variable names to match. The

WHERE

clause can be given to select rows using more general conditions.

With the

GLOBAL

modifier,

SHOW STATUS

displays the status values for all connections to MariaDB. With

SESSION

, it displays the status values for the current connection. If no modifier is present, the default is

SESSION

.

LOCAL

is a synonym for

SESSION

. If you see a lot of 0 values, the reason is probably that you have used

SHOW STATUS

with a new connection instead of

SHOW GLOBAL STATUS

.

Some status variables have only a global value. For these, you get the same value for both

GLOBAL

and

SESSION

.

See [Server Status Variables](#) for a full list, scope and description of the variables that can be viewed with

SHOW STATUS

.

The

LIKE

clause, if present on its own, indicates which variable name to match.

The

WHERE

and

LIKE

clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

### Examples

Full output from [MariaDB 10.1.17](#):

```
SHOW GLOBAL STATUS;
```

Variable_name	Value
Aborted_clients	0
Aborted_connects	0
Access_denied_errors	0
Acl_column_grants	0
Acl_database_grants	2
Acl_function_grants	0
Acl_procedure_grants	0
Acl_proxy_users	2
Acl_role_grants	0
Acl_roles	0
Acl_table_grants	0
Acl_users	6
Aria_pagecache_blocks_not_flushed	0
Aria_pagecache_blocks_unused	15706
Aria_pagecache_blocks_used	0
Aria_pagecache_read_requests	0
Aria_pagecache_reads	0
Aria_pagecache_write_requests	0
Aria_pagecache_writes	0
Aria_transaction_log_syncs	0
Binlog_commits	0
Binlog_group_commits	0
Binlog_group_commit_trigger_count	0
Binlog_group_commit_trigger_lock_wait	0
Binlog_group_commit_trigger_timeout	0
Binlog_snapshot_file	
Binlog_snapshot_position	0
Binlog_bytes_written	0
Binlog_cache_disk_use	0
Binlog_cache_use	0
Binlog_stmt_cache_disk_use	0
Binlog_stmt_cache_use	0
Busy_time	0.000000
Bytes_received	432
Bytes_sent	15183
Com_admin_commands	1
Com_alter_db	0
Com_alter_db_upgrade	0
Com_alter_event	0
Com_alter_function	0
Com_alter_procedure	0
Com_alter_server	0
Com_alter_table	0
Com_alter_tablespace	0
Com_analyze	0
Com_assign_to_keycache	0
Com_begin	0
Com_binlog	0
Com_call_procedure	0
Com_change_db	0
Com_change_master	0
Com_check	0
Com_checksum	0
Com_commit	0
Com_compound_sql	0
Com_create_db	0
Com_create_event	0
Com_create_function	0
Com_create_index	0
Com_create_procedure	0
Com_create_role	0
Com_create_server	0
Com_create_table	0
Com_create_temporary_table	0
Com_create_trigger	0
Com_create_udf	0
Com_create_user	0
Com_create_view	0
Com_dealloc_sql	0
Com_delete	0
Com_delete_multi	0
Com_do	0
Com_drop_db	0
Com_drop_event	0
Com_drop_function	0

Com_analyze	0
Com_drop_index	0
Com_drop_procedure	0
Com_drop_role	0
Com_drop_server	0
Com_drop_table	0
Com_drop_temporary_table	0
Com_drop_trigger	0
Com_drop_user	0
Com_drop_view	0
Com_empty_query	0
Com_execute_sql	0
Com_flush	0
Com_get_diagnostics	0
Com_grant	0
Com_grant_role	0
Com_ha_close	0
Com_ha_open	0
Com_ha_read	0
Com_help	0
Com_insert	0
Com_insert_select	0
Com_install_plugin	0
Com_kill	0
Com_load	0
Com_lock_tables	0
Com_optimize	0
Com_preload_keys	0
Com_prepare_sql	0
Com_purge	0
Com_purge_before_date	0
Com_release_savepoint	0
Com_rename_table	0
Com_rename_user	0
Com_repair	0
Com_replace	0
Com_replace_select	0
Com_reset	0
Com_resignal	0
Com_revoke	0
Com_revoke_all	0
Com_revoke_role	0
Com_rollback	0
Com_rollback_to_savepoint	0
Com_savepoint	0
Com_select	1
Com_set_option	0
Com_show_authors	0
Com_show_binlog_events	0
Com_show_binlogs	0
Com_showCharsets	0
Com_show_collations	0
Com_show_contributors	0
Com_show_create_db	0
Com_show_create_event	0
Com_show_create_func	0
Com_show_create_proc	0
Com_show_create_table	0
Com_show_create_trigger	0
Com_show_databases	0
Com_show_engine_logs	0
Com_show_engine_mutex	0
Com_show_engine_status	0
Com_show_errors	0
Com_show_events	0
Com_show_explain	0
Com_show_fields	0
Com_show_function_status	0
Com_show_generic	0
Com_show_grants	0
Com_show_keys	0
Com_show_master_status	0
Com_show_open_tables	0
Com_show_plugins	0
Com_show_privileges	0
Com_show_procedure_status	0
Com_show_processlist	0
Com_show_profile	0

Com_show_profiles	0
Com_show_relaylog_events	0
Com_show_slave_hosts	0
Com_show_slave_status	0
Com_show_status	2
Com_show_storage_engines	0
Com_show_table_status	0
Com_show_tables	0
Com_show_triggers	0
Com_show_variables	0
Com_show_warnings	0
Com_shutdown	0
Com_signal	0
Com_start_all_slaves	0
Com_start_slave	0
Com_stmt_close	0
Com_stmt_execute	0
Com_stmt_fetch	0
Com_stmt_prepare	0
Com_stmt_reprepare	0
Com_stmt_reset	0
Com_stmt_send_long_data	0
Com_stop_all_slaves	0
Com_stop_slave	0
Com_truncate	0
Com_uninstall_plugin	0
Com_unlock_tables	0
Com_update	0
Com_update_multi	0
Com_xa_commit	0
Com_xa_end	0
Com_xa_prepare	0
Com_xa_recover	0
Com_xa_rollback	0
Com_xa_start	0
Compression	OFF
Connection_errors_accept	0
Connection_errors_internal	0
Connection_errors_max_connections	0
Connection_errors_peer_address	0
Connection_errors_select	0
Connection_errors_tcpwrap	0
Connections	4
Cpu_time	0.000000
Created_tmp_disk_tables	0
Created_tmp_files	6
Created_tmp_tables	2
Delayed_errors	0
Delayed_insert_threads	0
Delayed_writes	0
Delete_scan	0
Empty_queries	0
Executed_events	0
Executed_triggers	0
Feature_delay_key_write	0
Feature_dynamic_columns	0
Feature_fulltext	0
Feature_gis	0
Feature_locale	0
Feature_subquery	0
Feature_timezone	0
Feature_trigger	0
Feature_xml	0
Flush_commands	1
Handler_commit	1
Handler_delete	0
Handler_discover	0
Handler_external_lock	0
Handler_icp_attempts	0
Handler_icp_match	0
Handler_mrr_init	0
Handler_mrr_key_refills	0
Handler_mrr_rowid_refills	0
Handler_prepare	0
Handler_read_first	3
Handler_read_key	0
Handler_read_last	0
Handler_read_next	0

Handler_read_prev	0
Handler_read_retry	0
Handler_read_rnd	0
Handler_read_rnd_deleted	0
Handler_read_rnd_next	537
Handler_rollback	0
Handler_savepoint	0
Handler_savepoint_rollback	0
Handler_tmp_update	0
Handler_tmp_write	516
Handler_update	0
Handler_write	0
Innodb_available_undo_logs	128
Innodb_background_log_sync	222
Innodb_buffer_pool_bytes_data	2523136
Innodb_buffer_pool_bytes_dirty	0
Innodb_buffer_pool_dump_status	Dumping buffer pool(s) <b>not</b> yet started
Innodb_buffer_pool_load_status	Loading buffer pool(s) <b>not</b> yet started
Innodb_buffer_pool_pages_data	154
Innodb_buffer_pool_pages_dirty	0
Innodb_buffer_pool_pages_flushed	1
Innodb_buffer_pool_pages_free	8037
Innodb_buffer_pool_pages_lru_flushed	0
Innodb_buffer_pool_pages_made_not_young	0
Innodb_buffer_pool_pages_made_young	0
Innodb_buffer_pool_pages_misc	0
Innodb_buffer_pool_pages_old	0
Innodb_buffer_pool_pages_total	8191
Innodb_buffer_pool_read_ahead	0
Innodb_buffer_pool_read_ahead_evicted	0
Innodb_buffer_pool_read_ahead_rnd	0
Innodb_buffer_pool_read_requests	558
Innodb_buffer_pool_reads	155
Innodb_buffer_pool_wait_free	0
Innodb_buffer_pool_write_requests	1
Innodb_checkpoint_age	0
Innodb_checkpoint_max_age	80826164
Innodb_data_fsyncs	5
Innodb_data_pending_fsyncs	0
Innodb_data_pending_reads	0
Innodb_data_pending_writes	0
Innodb_data_read	2609664
Innodb_data_reads	172
Innodb_data_writes	5
Innodb_data_written	34304
Innodb dblwr_pages_written	1
Innodb dblwr_writes	1
Innodb deadlocks	0
Innodb have_atomic_builtin	ON
Innodb history_list_length	0
Innodb ibuf_discarded_delete_marks	0
Innodb ibuf_discarded_deletes	0
Innodb ibuf_discarded_inserts	0
Innodb ibuf_free_list	0
Innodb ibuf_merged_delete_marks	0
Innodb ibuf_merged_deletes	0
Innodb ibuf_merged_inserts	0
Innodb ibuf_merges	0
Innodb ibuf_segment_size	2
Innodb ibuf_size	1
Innodb log_waits	0
Innodb log_write_requests	0
Innodb log_writes	1
Innodb lsn_current	1616829
Innodb lsn_flushed	1616829
Innodb lsn_last_checkpoint	1616829
Innodb master_thread_active_loops	0
Innodb master_thread_idle_loops	222
Innodb max_trx_id	2308
Innodb mem_adaptive_hash	2217568
Innodb mem_dictionary	630703
Innodb mem_total	140771328
Innodb mutex_os_waits	1
Innodb mutex_spin_rounds	30
Innodb mutex_spin_waits	1
Innodb oldest_view_low_limit_trx_id	0
Innodb os_log_fsyncs	3

Innodb_os_log_pending_fsyncs	0
Innodb_os_log_pending_writes	0
Innodb_os_log_written	512
Innodb_page_size	16384
Innodb_pages_created	0
Innodb_pages_read	154
Innodb_pages_written	1
Innodb_purge_trx_id	0
Innodb_purge_undo_no	0
Innodb_read_views_memory	88
Innodb_row_lock_current_waits	0
Innodb_row_lock_time	0
Innodb_row_lock_time_avg	0
Innodb_row_lock_time_max	0
Innodb_row_lock_waits	0
Innodb_rows_deleted	0
Innodb_rows_inserted	0
Innodb_rows_read	0
Innodb_rows_updated	0
Innodb_system_rows_deleted	0
Innodb_system_rows_inserted	0
Innodb_system_rows_read	0
Innodb_system_rows_updated	0
Innodb_s_lock_os_waits	2
Innodb_s_lock_spin_rounds	60
Innodb_s_lock_spin_waits	2
Innodb_truncated_status_writes	0
Innodb_x_lock_os_waits	0
Innodb_x_lock_spin_rounds	0
Innodb_x_lock_spin_waits	0
Innodb_page_compression_saved	0
Innodb_page_compression_trim_512	0
Innodb_page_compression_trim_1024	0
Innodb_page_compression_trim_2048	0
Innodb_page_compression_trim_4096	0
Innodb_page_compression_trim_8192	0
Innodb_page_compression_trim_16384	0
Innodb_page_compression_trim_32768	0
Innodb_num_index_pages_written	0
Innodb_num_non_index_pages_written	5
Innodb_num_pages_page_compressed	0
Innodb_num_page_compressed_trim_op	0
Innodb_num_page_compressed_trim_op_saved	0
Innodb_num_pages_page_decompressed	0
Innodb_num_pages_page_compression_error	0
Innodb_num_pages_encrypted	0
Innodb_num_pages_decrypted	0
Innodb_have_lz4	OFF
Innodb_have_lzo	OFF
Innodb_have_lzma	OFF
Innodb_have_bzip2	OFF
Innodb_have_snappy	OFF
Innodb_defragment_compression_failures	0
Innodb_defragment_failures	0
Innodb_defragment_count	0
Innodb_onlineddl_rowlog_rows	0
Innodb_onlineddl_rowlog_pct_used	0
Innodb_onlineddl_pct_progress	0
Innodb_secondary_index_triggered_cluster_reads	0
Innodb_secondary_index_triggered_cluster_reads_avoided	0
Innodb_encryption_rotation_pages_read_from_cache	0
Innodb_encryption_rotation_pages_read_from_disk	0
Innodb_encryption_rotation_pages_modified	0
Innodb_encryption_rotation_pages_flushed	0
Innodb_encryption_rotation_estimated_iops	0
Innodb_scrub_background_page_reorganizations	0
Innodb_scrub_background_page_splits	0
Innodb_scrub_background_page_split_failures_underflow	0
Innodb_scrub_background_page_split_failures_out_of_filespace	0
Innodb_scrub_background_page_split_failures_missing_index	0
Innodb_scrub_background_page_split_failures_unknown	0
Key_blocks_not_flushed	0
Key_blocks_unused	107163
Key_blocks_used	0
Key_blocks_warm	0
Key_read_requests	0
Key_reads	0
Key_write_requests	0

Key_writes	0
Last_query_cost	0.000000
Master_gtid_wait_count	0
Master_gtid_wait_time	0
Master_gtid_wait_timeouts	0
Max_statement_time_exceeded	0
Max_used_connections	1
Memory_used	273614696
Not_flushed_delayed_rows	0
Open_files	25
Open_streams	0
Open_table_definitions	18
Open_tables	11
Opened_files	77
Opened_plugin_libraries	0
Opened_table_definitions	18
Opened_tables	18
Opened_views	0
Performance_schema_accounts_lost	0
Performance_schema_cond_classes_lost	0
Performance_schema_cond_instances_lost	0
Performance_schema_digest_lost	0
Performance_schema_file_classes_lost	0
Performance_schema_file_handles_lost	0
Performance_schema_file_instances_lost	0
Performance_schema_hosts_lost	0
Performance_schema_locker_lost	0
Performance_schema_mutex_classes_lost	0
Performance_schema_mutex_instances_lost	0
Performance_schema_rwlock_classes_lost	0
Performance_schema_rwlock_instances_lost	0
Performance_schema_session_connect_attrs_lost	0
Performance_schema_socket_classes_lost	0
Performance_schema_socket_instances_lost	0
Performance_schema_stage_classes_lost	0
Performance_schema_statement_classes_lost	0
Performance_schema_table_handles_lost	0
Performance_schema_table_instances_lost	0
Performance_schema_thread_classes_lost	0
Performance_schema_thread_instances_lost	0
Performance_schema_users_lost	0
Prepared_stmt_count	0
Qcache_free_blocks	1
Qcache_free_memory	1031336
Qcache_hits	0
Qcache_inserts	0
Qcache_lowmem_prunes	0
Qcache_not_cached	0
Qcache_queries_in_cache	0
Qcache_total_blocks	1
Queries	4
Questions	4
Rows_read	10
Rows_sent	517
Rows_tmp_read	516
Rpl_status	AUTH_MASTER
Select_full_join	0
Select_full_range_join	0
Select_range	0
Select_range_check	0
Select_scan	2
Slave_connections	0
Slave_heartbeat_period	0.000
Slave_open_temp_tables	0
Slave_received_heartbeats	0
Slave_retried_transactions	0
Slave_running	OFF
Slave_skipped_errors	0
Slaves_connected	0
Slaves_running	0
Slow_launch_threads	0
Slow_queries	0
Sort_merge_passes	0
Sort_priority_queue_sorts	0
Sort_range	0
Sort_rows	0
Sort_scan	0
Ssl_accept_renegotiates	0

Ssl_accepts_renegotiates	0
Ssl_accepts	0
Ssl_callback_cache_hits	0
Ssl_cipher	0
Ssl_cipher_list	0
Ssl_client_connects	0
Ssl_connect_renegotiates	0
Ssl_ctx_verify_depth	0
Ssl_ctx_verify_mode	0
Ssl_default_timeout	0
Ssl_finished_accepts	0
Ssl_finished_connects	0
Ssl_server_not_after	0
Ssl_server_not_before	0
Ssl_session_cache_hits	0
Ssl_session_cache_misses	0
Ssl_session_cache_mode	NONE
Ssl_session_cache_overflows	0
Ssl_session_cache_size	0
Ssl_session_cache_timeouts	0
Ssl_sessions_reused	0
Ssl_used_session_cache_entries	0
Ssl_verify_depth	0
Ssl_verify_mode	0
Ssl_version	0
Subquery_cache_hit	0
Subquery_cache_miss	0
Syncs	2
Table_locks_immediate	21
Table_locks_waited	0
Tc_log_max_pages_used	0
Tc_log_page_size	4096
Tc_log_page_waits	0
Threadpool_idle_threads	0
Threadpool_threads	0
Threads_cached	0
Threads_connected	1
Threads_created	2
Threads_running	1
Update_scan	0
Uptime	223
Uptime_since_flush_status	223
wsrep_cluster_conf_id	18446744073709551615
wsrep_cluster_size	0
wsrep_cluster_state_uuid	0
wsrep_cluster_status	Disconnected
wsrep_connected	OFF
wsrep_local_bf_aborts	0
wsrep_local_index	18446744073709551615
wsrep_provider_name	
wsrep_provider_vendor	
wsrep_provider_version	
wsrep_ready	OFF
wsrep_thread_count	0

516 rows in set (0.00 sec)

Example of filtered output:

SHOW STATUS LIKE 'Key%';		
Variable_name	Value	
Key_blocks_not_flushed	0	
Key_blocks_unused	107163	
Key_blocks_used	0	
Key_blocks_warm	0	
Key_read_requests	0	
Key_reads	0	
Key_write_requests	0	
Key_writes	0	

8 rows in set (0.00 sec)

## 1.1.2.8.49 SHOW TABLE STATUS

# Syntax

```
SHOW TABLE STATUS [{FROM | IN} db_name]
[LIKE 'pattern' | WHERE expr]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)

## Description

SHOW TABLE STATUS  
works like

[SHOW TABLES](#)

, but provides more extensive information about each non-  
TEMPORARY  
table.

The

LIKE  
clause, if present on its own, indicates which table names to match. The  
WHERE  
and  
LIKE  
clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

The following information is returned:

Column	Description
Name	Table name.
Engine	Table <a href="#">storage engine</a> .
Version	Version number from the table's .frm file.
Row_format	Row format (see <a href="#">InnoDB</a> , <a href="#">Aria</a> and <a href="#">MyISAM</a> row formats).
Rows	Number of rows in the table. Some engines, such as <a href="#">XtraDB</a> and <a href="#">InnoDB</a> may store an estimate.
Avg_row_length	Average row length in the table.
Data_length	For <a href="#">InnoDB/XtraDB</a> , the index size, in pages, multiplied by the page size. For <a href="#">Aria</a> and <a href="#">MyISAM</a> , length of the data file, in bytes. For <a href="#">MEMORY</a> , the approximate allocated memory.
Max_data_length	Maximum length of the data file, ie the total number of bytes that could be stored in the table. Not used in <a href="#">XtraDB</a> and <a href="#">InnoDB</a> .
Index_length	Length of the index file.
Data_free	Bytes allocated but unused. For InnoDB tables in a shared tablespace, the free space of the shared tablespace with small safety margin. An estimate in the case of partitioned tables - see the <a href="#">PARTITIONS</a> table.
Auto_increment	Next <a href="#">AUTO_INCREMENT</a> value.
Create_time	Time the table was created.

Update_time	Time the table was last updated. On Windows, the timestamp is not updated on update, so MyISAM values will be inaccurate. In <a href="#">InnoDB</a> , if shared tablespaces are used, will be NULL , while buffering can also delay the update, so the value will differ from the actual time of the last UPDATE , INSERT or DELETE .
Check_time	Time the table was last checked. Not kept by all storage engines, in which case will be NULL .
Collation	<a href="#">Character set and collation</a> .
Checksum	Live checksum value, if any.
Create_options	Extra  <a href="#">CREATE TABLE</a>  options.
Comment	Table comment provided when MariaDB created the table.
Max_index_length	Maximum index length (supported by MyISAM and Aria tables). Added in <a href="#">MariaDB 10.3.5</a> .
Temporary	Placeholder to signal that a table is a temporary table. Currently always "N", except "Y" for generated information_schema tables and NULL for <a href="#">views</a> . Added in <a href="#">MariaDB 10.3.5</a> .

Similar information can be found in the

[information\\_schema.TABLES](#)

table as well as by using

[mysqlshow](#)

:

```
mysqlshow --status db_name
```

## Example

```
show table status\G
***** 1. row *****
  Name: bus_routes
  Engine: InnoDB
  Version: 10
Row_format: Dynamic
  Rows: 5
Avg_row_length: 3276
  Data_length: 16384
Max_data_length: 0
  Index_length: 0
  Data_free: 0
Auto_increment: NULL
  Create_time: 2017-05-24 11:17:46
  Update_time: NULL
  Check_time: NULL
  Collation: latin1_swedish_ci
  Checksum: NULL
Create_options:
  Comment:
```

## 1.1.2.8.50 SHOW TABLES

### Syntax

```
SHOW [FULL] TABLES [FROM db_name]
      [LIKE 'pattern' | WHERE expr]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

SHOW TABLES  
lists the non-  
TEMPORARY  
tables, [sequences](#) and [views](#) in a given database.

The

LIKE  
clause, if present on its own, indicates which table names to match. The  
WHERE  
and  
LIKE  
clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#). For example, when searching for tables in  
the  
test  
database, the column name for use in the  
WHERE  
and  
LIKE  
clauses will be  
`Tables_in_test`

The

FULL  
modifier is supported such that  
SHOW FULL TABLES  
displays a second output column. Values for the second column.  
Table\_type  
, are  
BASE TABLE  
for a table,  
VIEW  
for a [view](#) and  
SEQUENCE  
for a [sequence](#).

You can also get this information using:

```
mysqlshow db_name
```

See [mysqlshow](#) for more details.

If you have no privileges for a base table or view, it does not show up in the output from

```
SHOW TABLES
or
mysqlshow
db_name
```

The

```
information_schema.TABLES
```

table, as well as the

```
SHOW TABLE STATUS
```

statement, provide extended information about tables.

## Examples

```
SHOW TABLES;
+-----+
| Tables_in_test |
+-----+
| animal_count   |
| animals        |
| are_the_mooses_loose |
| aria_test2     |
| t1             |
| view1          |
+-----+
```

Showing the tables beginning with a only.

```
SHOW TABLES WHERE Tables_in_test LIKE 'a%';
+-----+
| Tables_in_test |
+-----+
| animal_count   |
| animals        |
| are_the_mooses_loose |
| aria_test2     |
+-----+
```

Showing tables and table types:

```
SHOW FULL TABLES;
+-----+-----+
| Tables_in_test | Table_type |
+-----+-----+
| s1           | SEQUENCE   |
| student       | BASE TABLE |
| v1           | VIEW        |
+-----+-----+
```

## See Also

- [SHOW TABLE STATUS](#)
- The [information\\_schema.TABLES](#) table

## 1.1.2.8.51 SHOW TABLE\_STATISTICS

### Syntax

```
SHOW TABLE_STATISTICS
```

### Description

The

```
SHOW TABLE_STATISTICS
```

statement is part of the [User Statistics](#) feature. It was removed as a separate statement in [MariaDB 10.1.1](#), but effectively replaced by the generic [SHOW information\\_schema\\_table](#) statement. The [information\\_schema.TABLE\\_STATISTICS](#) table shows statistics on table usage

The [userstat](#) system variable must be set to 1 to activate this feature. See the [User Statistics](#) and [information\\_schema.TABLE\\_STATISTICS](#) articles for more information.

### Example

```

SHOW TABLE_STATISTICS[G]
*****
***** 1. row *****
  Table_schema: mysql
    Table_name: proxies_priv
      Rows_read: 2
      Rows_changed: 0
Rows_changed_x_indexes: 0
***** 2. row *****
  Table_schema: test
    Table_name: employees_example
      Rows_read: 7
      Rows_changed: 0
Rows_changed_x_indexes: 0
***** 3. row *****
  Table_schema: mysql
    Table_name: user
      Rows_read: 16
      Rows_changed: 0
Rows_changed_x_indexes: 0
***** 4. row *****
  Table_schema: mysql
    Table_name: db
      Rows_read: 2
      Rows_changed: 0
Rows_changed_x_indexes: 0

```

## 1.1.2.8.52 SHOW TRIGGERS

### Syntax

```

SHOW TRIGGERS [FROM db_name]
  [LIKE 'pattern' | WHERE expr]

```

### Contents

- 1. [Syntax](#)
- 2. [Description](#)
- 3. [Examples](#)
- 4. [See also](#)

### Description

`SHOW TRIGGERS`  
lists the triggers currently defined for tables in a database (the default database unless a `FROM` clause is given). This statement requires the

#### TRIGGER

privilege (prior to MySQL 5.1.22, it required the `SUPER` privilege).

The

`LIKE` clause, if present on its own, indicates which table names to match and causes the statement to display triggers for those tables. The `WHERE` and `LIKE` clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

Similar information is stored in the

#### `information_schema.TRIGGERS`

table.

MariaDB starting with 10.2.3

If there are multiple triggers for the same action, then the triggers are shown in action order.

## Examples

For the trigger defined at [Trigger Overview](#) :

```
SHOW triggers Like 'animals' \G
***** 1. row *****
    Trigger: the_mooses_are_loose
      Event: INSERT
      Table: animals
    Statement: BEGIN
IF NEW.name = 'Moose' THEN
  UPDATE animal_count SET animal_count.animals = animal_count.animals+100;
ELSE
  UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
END IF;
END
      Timing: AFTER
      Created: 2016-09-29 13:53:34.35
      sql_mode:
        Definer: root@localhost
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

Listing all triggers associated with a certain table:

```
SHOW TRIGGERS FROM test WHERE `Table` = 'user' \G
***** 1. row *****
    Trigger: user_ai
      Event: INSERT
      Table: user
    Statement: BEGIN END
      Timing: AFTER
      Created: 2016-09-29 13:53:34.35
      sql_mode:
        Definer: root@%
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

```
SHOW triggers WHERE Event Like 'Insert' \G
***** 1. row *****
    Trigger: the_mooses_are_loose
      Event: INSERT
      Table: animals
    Statement: BEGIN
IF NEW.name = 'Moose' THEN
  UPDATE animal_count SET animal_count.animals = animal_count.animals+100;
ELSE
  UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
END IF;
END
      Timing: AFTER
      Created: 2016-09-29 13:53:34.35
      sql_mode:
        Definer: root@localhost
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
```

- character\_set\_client  
is the session value of the  
[character\\_set\\_client](#)  
system variable when the trigger was created.
- collation\_connection  
is the session value of the

`collation_connection`

system variable when the trigger was created.

- Database Collation

is the collation of the database with which the trigger is associated.

These columns were added in MariaDB/MySQL 5.1.21.

Old triggers created before MySQL 5.7 and [MariaDB 10.2.3](#) has NULL in the  
Created  
column.

## See also

- [Trigger Overview](#)
- 

`CREATE TRIGGER`

- 

`DROP TRIGGER`

- 

`information_schema.TRIGGERS`

table

- 

`SHOW CREATE TRIGGER`

- [Trigger Limitations](#)

## 1.1.2.8.53 SHOW USER\_STATISTICS

### Syntax

```
SHOW USER_STATISTICS
```

### Description

The

```
SHOW USER_STATISTICS
```

statement is part of the [User Statistics](#) feature. It was removed as a separate statement in [MariaDB 10.1.1](#), but effectively replaced by the generic `SHOW information_schema_table` statement. The `information_schema.USER_STATISTICS` table holds statistics about user activity. You can use this table to find out such things as which user is causing the most load and which users are being abusive. You can also use this table to measure how close to capacity the server may be.

The `userstat` system variable must be set to 1 to activate this feature. See the [User Statistics](#) and `information_schema.USER_STATISTICS` table for more information.

### Example

```

SHOW USER_STATISTICS [G]
*****
***** 1. row *****
    User: root
    Total_connections: 1
    Concurrent_connections: 0
    Connected_time: 3297
        Busy_time: 0.1411340000000006
        Cpu_time: 0.01763700000000003
    Bytes_received: 969
        Bytes_sent: 22355
    Binlog_bytes_written: 0
        Rows_read: 10
        Rows_sent: 67
        Rows_deleted: 0
        Rows_inserted: 0
        Rows_updated: 0
    Select_commands: 7
    Update_commands: 0
    Other_commands: 0
    Commit_transactions: 1
    Rollback_transactions: 0
    Denied_connections: 0
    Lost_connections: 0
    Access_denied: 0
    Empty_queries: 7

```

## 1.1.2.8.54 SHOW VARIABLES

### Syntax

```

SHOW [GLOBAL | SESSION] VARIABLES
[LIKE 'pattern' | WHERE expr]

```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

### Description

`SHOW VARIABLES`  
shows the values of MariaDB [system variables](#). This information also can be obtained using the

`mysqladmin`

variables command. The  
`LIKE`  
clause, if present, indicates which variable names to match. The  
`WHERE`  
clause can be given to select rows using more general conditions.

With the

`GLOBAL`  
modifier,  
`SHOW VARIABLES`  
displays the values that are used for new connections to MariaDB. With  
`SESSION`  
, it displays the values that are in effect for the current connection. If no modifier is present, the default is  
`SESSION`  
  
`LOCAL`  
is a synonym for  
`SESSION`  
. With a  
`LIKE`  
clause, the statement displays only rows for those variables with names that match the pattern. To obtain the row for a specific variable, use a  
`LIKE`  
clause as shown:

```
SHOW VARIABLES LIKE 'maria_group_commit';
SHOW SESSION VARIABLES LIKE 'maria_group_commit';
```

To get a list of variables whose name match a pattern, use the "

%  
" wildcard character in a  
LIKE  
clause:

```
SHOW VARIABLES LIKE '%maria%';
SHOW GLOBAL VARIABLES LIKE '%maria%';
```

Wildcard characters can be used in any position within the pattern to be matched. Strictly speaking, because "

" is a wildcard that matches any single character, you should escape it as "  
\\_"  
" to match it literally. In practice, this is rarely necessary.

The

WHERE  
and  
LIKE

clauses can be given to select rows using more general conditions, as discussed in [Extended SHOW](#).

See

[SET](#)

for information on setting server system variables.

See [Server System Variables](#) for a list of all the variables that can be set.

You can also see the server variables by querying the [Information Schema GLOBAL\\_VARIABLES](#) and [SESSION\\_VARIABLES](#) tables.

## Examples

```
SHOW VARIABLES LIKE 'aria%';
+-----+-----+
| Variable_name          | Value   |
+-----+-----+
| aria_block_size         | 8192    |
| aria_checkpoint_interval | 30      |
| aria_checkpoint_log_activity | 1048576 |
| aria_force_start_after_recovery_failures | 0       |
| aria_group_commit        | none    |
| aria_group_commit_interval | 0       |
| aria_log_file_size       | 1073741824 |
| aria_log_purge_type      | immediate |
| aria_max_sort_file_size  | 9223372036853727232 |
| aria_page_checksum        | ON      |
| aria_pagecache_age_threshold | 300    |
| aria_pagecache_buffer_size | 134217728 |
| aria_pagecache_division_limit | 100    |
| aria_recover             | NORMAL  |
| aria_repair_threads       | 1       |
| aria_sort_buffer_size     | 134217728 |
| aria_stats_method         | nulls_unequal |
| aria_sync_log_dir         | NEWFILE |
| aria_used_for_temp_tables | ON      |
+-----+-----+
```

```

SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME LIKE 'max_error_count' OR
VARIABLE_NAME LIKE 'innodb_sync_spin_loops';
+-----+-----+-----+
| VARIABLE_NAME | SESSION_VALUE | GLOBAL_VALUE |
+-----+-----+-----+
| MAX_ERROR_COUNT | 64 | 64 |
| INNODB_SYNC_SPIN_LOOPS | NULL | 30 |
+-----+-----+-----+

SET GLOBAL max_error_count=128;

SELECT VARIABLE_NAME, SESSION_VALUE, GLOBAL_VALUE FROM
INFORMATION_SCHEMA.SYSTEM_VARIABLES WHERE
VARIABLE_NAME LIKE 'max_error_count' OR
VARIABLE_NAME LIKE 'innodb_sync_spin_loops';
+-----+-----+-----+
| VARIABLE_NAME | SESSION_VALUE | GLOBAL_VALUE |
+-----+-----+-----+
| MAX_ERROR_COUNT | 64 | 128 |
| INNODB_SYNC_SPIN_LOOPS | NULL | 30 |
+-----+-----+-----+

SET GLOBAL max_error_count=128;

SHOW VARIABLES LIKE 'max_error_count';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_error_count | 64 |
+-----+-----+

SHOW GLOBAL VARIABLES LIKE 'max_error_count';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_error_count | 128 |
+-----+-----+

```

Because the following variable only has a global scope, the global value is returned even when specifying SESSION (in this case by default):

```

SHOW VARIABLES LIKE 'innodb_sync_spin_loops';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| innodb_sync_spin_loops | 30 |
+-----+-----+

```

## 1.1.2.8.55 SHOW WARNINGS

### Syntax

```

SHOW WARNINGS [LIMIT [offset,] row_count]
SHOW ERRORS [LIMIT row_count OFFSET offset]
SHOW COUNT(*) WARNINGS

```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
  1. [Stack Trace](#)
4. [See Also](#)

### Description

`SHOW WARNINGS`

shows the error, warning, and note messages that resulted from the last statement that generated messages in the current session. It shows nothing if the last statement used a table and generated no messages. (That is, a statement that uses a table but generates no messages clears the

message list.) Statements that do not use tables and do not generate messages have no effect on the message list.

A note is different to a warning in that it only appears if the

#### `sql_notes`

variable is set to 1 (the default), and is not converted to an error if

#### `strict mode`

is enabled.

A related statement,

#### `SHOW ERRORS`

, shows only the errors.

The

`SHOW COUNT(*) WARNINGS`

statement displays the total number of errors, warnings, and notes. You can also retrieve this number from the `warning_count` variable:

```
SHOW COUNT(*) WARNINGS;  
SELECT @@warning_count;
```

The value of

#### `warning_count`

might be greater than the number of messages displayed by

`SHOW WARNINGS`

if the

#### `max_error_count`

system variable is set so low that not all messages are stored.

The

`LIMIT`

clause has the same syntax as for the

#### `SELECT statement`

.

.

`SHOW WARNINGS`  
can be used after

#### `EXPLAIN EXTENDED`

to see how a query is internally rewritten by MariaDB.

If the

#### `sql_notes`

server variable is set to 1, Notes are included in the output of

`SHOW WARNINGS`

; if it is set to 0, this statement will not show (or count) Notes.

The results of

`SHOW WARNINGS`

and

`SHOW COUNT(*) WARNINGS`

are directly sent to the client. If you need to access those information in a stored program, you can use the

#### `GET DIAGNOSTICS`

statement instead.

For a list of MariaDB error codes, see [MariaDB Error Codes](#).

The

```
mysql
```

client also has a number of options related to warnings. The  
\W  
command will show warnings after every statement, while  
\w  
will disable this. Starting the client with the  
--show-warnings  
option will show warnings after every statement.

MariaDB 10.3.1 implements a stored routine error stack trace.

```
SHOW WARNINGS
```

can also be used to show more information. See the example below.

## Examples

```
SELECT 1/0;
+-----+
| 1/0 |
+-----+
| NULL |
+-----+

SHOW COUNT(*) WARNINGS;
+-----+
| @@session.warning_count |
+-----+
| 1 |
+-----+

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message      |
+-----+-----+
| Warning | 1365 | Division by 0 |
+-----+-----+
```

## Stack Trace

From MariaDB 10.3.1 , displaying a stack trace:

```
DELIMITER $$  
CREATE OR REPLACE PROCEDURE p1()  
BEGIN  
    DECLARE c CURSOR FOR SELECT * FROM not_existing;  
    OPEN c;  
    CLOSE c;  
END;  
$$  
CREATE OR REPLACE PROCEDURE p2()  
BEGIN  
    CALL p1;  
END;  
$$  
DELIMITER ;  
CALL p2;  
ERROR 1146 (42S02): Table 'test.not_existing' doesn't exist

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message      |
+-----+-----+
| Error | 1146 | Table 'test.not_existing' doesn't exist |
| Note  | 4091 | At line 6 in test.p1          |
| Note  | 4091 | At line 4 in test.p2          |
+-----+-----+
```

SHOW WARNINGS  
displays a stack trace, showing where the error actually happened:

- Line 4 in test.p1 is the OPEN command which actually raised the error

- Line 3 in test.p2 is the CALL statement, calling p1 from p2.

## See Also

- [SHOW ERRORS](#)

## 1.1.2.8.56 SHOW WSREP\_MEMBERSHIP

SHOW WSREP\_MEMBERSHIP  
is part of the

[WSREP\\_INFO](#)

plugin.

## Syntax

```
SHOW WSREP_MEMBERSHIP
```

## Description

The

SHOW WSREP\_MEMBERSHIP  
statement returns [Galera](#) node cluster membership information. It returns the same information as found in the  
[information\\_schema.WSREP\\_MEMBERSHIP](#)  
table. Only users with the  
[SUPER](#)  
privilege can access this information.

## Examples

```
SHOW WSREP_MEMBERSHIP;
+-----+-----+-----+
| Index | Uuid                | Name      | Address      |
+-----+-----+-----+
|   0   | 19058073-8940-11e4-8570-16af7bf8fcfd | my_node1  | 10.0.2.15:16001 |
|   1   | 19f2b00-8942-11e4-9cb8-b39e8ee0b5dd | my_node3  | 10.0.2.15:16003 |
|   2   | d85e62db-8941-11e4-b1ef-4bc9980e476d | my_node2  | 10.0.2.15:16002 |
+-----+-----+-----+
```

## 1.1.2.8.57 SHOW WSREP\_STATUS

SHOW WSREP\_STATUS  
is part of the

[WSREP\\_INFO](#)

plugin.

## Syntax

```
SHOW WSREP_STATUS
```

## Description

The

SHOW WSREP\_STATUS  
statement returns [Galera](#) node and cluster status information. It returns the same information as found in the

## [information\\_schema.WSREP\\_STATUS](#)

table. Only users with the

[SUPER](#)

privilege can access this information.

## Examples

```
SHOW WSREP_STATUS;
+-----+-----+-----+
| Node_Index | Node_Status | Cluster_Status | Cluster_Size |
+-----+-----+-----+
|      0 | Synced     | Primary       |      3 |
+-----+-----+-----+
```

## 1.1.2.9 System Tables

### 1.1.2.9.1 Information Schema

#### 1.1.2.9.1.1 Information Schema Tables

##### 1.1.2.9.1.1.1 Information Schema InnoDB Tables

###### 1.1.2.9.1.1.1.1 Information Schema

###### **INNODB\_BUFFER\_PAGE** Table

The [Information Schema](#)

[INNODB\\_BUFFER\\_PAGE](#)

table contains information about pages in the [buffer pool](#).

The

[PROCESS](#)

[privilege](#) is required to view the table.

It has the following columns:

Column	Description
POOL_ID	Buffer Pool identifier. From <a href="#">MariaDB 10.5.1</a> returns a value of 0, since multiple InnoDB buffer pool instances has been removed.
BLOCK_ID	Buffer Pool Block identifier.
SPACE	Tablespace identifier. Matches the SPACE value in the <a href="#">INNODB_SYS_TABLES</a> table.
PAGE_NUMBER	Buffer pool page number.

PAGE_TYPE	Page type; one of allocated (newly-allocated page), index (B-tree node), undo_log (undo log page), inode (index node), ibuf_free_list (insert buffer free list), ibuf_bitmap (insert buffer bitmap), system (system page), trx_system (transaction system data), file_space_header (file space header), extent_descriptor (extent descriptor page), blob (uncompressed blob page), compressed_blob (first compressed blob page), compressed_blob2 (subsequent compressed blob page) or unknown .
FLUSH_TYPE	Flush type.
FIX_COUNT	Count of the threads using this block in the buffer pool. When it is zero, the block can be evicted from the buffer pool.
IS_HASHED	Whether or not a hash index has been built on this page.
NEWEST_MODIFICATION	Most recent modification's Log Sequence Number.
OLDEST_MODIFICATION	Oldest modification's Log Sequence Number.
ACCESS_TIME	Abstract number representing the time the page was first accessed.
TABLE_NAME	Table that the page belongs to.
INDEX_NAME	Index that the page belongs to, either a clustered index or a secondary index.
NUMBER_RECORDS	Number of records the page contains.

DATA_SIZE	Size in bytes of all the records contained in the page.
COMPRESSED_SIZE	Compressed size in bytes of the page, or NULL for pages that aren't compressed.
PAGE_STATE	Page state; one of FILE_PAGE (page from a file) or MEMORY (page from an in-memory object) for valid data, or one of NULL , READY_FOR_USE , NOT_USED , REMOVE_HASH
IO_FIX	Whether there is I/O pending for the page; one of IO_NONE (no pending I/O), IO_READ (read pending), IO_WRITE (write pending).
IS_OLD	Whether the page is old or not.
FREE_PAGE_CLOCK	Freed_page_clock counter, which tracks the number of blocks removed from the end of the least recently used (LRU) list, at the time the block was last placed at the head of the list.

The related [INFORMATION\\_SCHEMA.INNODB\\_BUFFER\\_PAGE\\_LRU](#) table contains the same information, but with an LRU (least recently used) position rather than block id.

## Examples

```
DESC information_schema.innodb_buffer_page;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| POOL_ID | bigint(21) unsigned | NO | | 0 | |
| BLOCK_ID | bigint(21) unsigned | NO | | 0 | |
| SPACE | bigint(21) unsigned | NO | | 0 | |
| PAGE_NUMBER | bigint(21) unsigned | NO | | 0 | |
| PAGE_TYPE | varchar(64) | YES | | NULL | |
| FLUSH_TYPE | bigint(21) unsigned | NO | | 0 | |
| FIX_COUNT | bigint(21) unsigned | NO | | 0 | |
| IS_HASHED | varchar(3) | YES | | NULL | |
| NEWEST_MODIFICATION | bigint(21) unsigned | NO | | 0 | |
| OLDEST_MODIFICATION | bigint(21) unsigned | NO | | 0 | |
| ACCESS_TIME | bigint(21) unsigned | NO | | 0 | |
| TABLE_NAME | varchar(1024) | YES | | NULL | |
| INDEX_NAME | varchar(1024) | YES | | NULL | |
| NUMBER_RECORDS | bigint(21) unsigned | NO | | 0 | |
| DATA_SIZE | bigint(21) unsigned | NO | | 0 | |
| COMPRESSED_SIZE | bigint(21) unsigned | NO | | 0 | |
| PAGE_STATE | varchar(64) | YES | | NULL | |
| IO_FIX | varchar(64) | YES | | NULL | |
| IS_OLD | varchar(3) | YES | | NULL | |
| FREE_PAGE_CLOCK | bigint(21) unsigned | NO | | 0 | |
+-----+-----+-----+-----+-----+
```

```

SELECT * FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE\G
...
***** 6. row *****
    POOL_ID: 0
    BLOCK_ID: 5
    SPACE: 0
  PAGE_NUMBER: 11
  PAGE_TYPE: INDEX
  FLUSH_TYPE: 1
  FIX_COUNT: 0
  IS_HASHED: NO
NEWEST_MODIFICATION: 2046835
OLDEST_MODIFICATION: 0
  ACCESS_TIME: 2585566280
  TABLE_NAME: `SYS_INDEXES`
  INDEX_NAME: CLUST_IND
NUMBER_RECORDS: 57
  DATA_SIZE: 4016
COMPRESSED_SIZE: 0
  PAGE_STATE: FILE_PAGE
  IO_FIX: IO_NONE
  IS_OLD: NO
FREE_PAGE_CLOCK: 0
...

```

## 1.1.2.9.1.1.1.2 Information Schema INNODB\_BUFFER\_PAGE\_LRU Table

The [Information Schema](#)

`INNODB_BUFFER_PAGE_LRU`

table contains information about pages in the [buffer pool](#) and how they are ordered for eviction purposes.

The

`PROCESS`

[privilege](#) is required to view the table.

It has the following columns:

Column	Description
<code>POOL_ID</code>	Buffer Pool identifier. From <a href="#">MariaDB 10.5.1</a> returns a value of 0, since multiple InnoDB buffer pool instances has been removed.
<code>LRU_POSITION</code>	LRU (Least recently-used), for determining eviction order from the buffer pool.
<code>SPACE</code>	Tablespace identifier. Matches the <code>SPACE</code> value on the <a href="#">INNODB_SYS_TABLES</a> table.
<code>PAGE_NUMBER</code>	Buffer pool page number.

PAGE_TYPE	Page type; one of allocated (newly-allocated page), index (B-tree node), undo_log (undo log page), inode (index node), ibuf_free_list (insert buffer free list), ibuf_bitmap (insert buffer bitmap), system (system page), trx_system (transaction system data), file_space_header (file space header), extent_descriptor (extent descriptor page), blob (uncompressed blob page), compressed_blob (first compressed blob page), compressed_blob2 (subsequent compressed blob page) or unknown .
FLUSH_TYPE	Flush type.
FIX_COUNT	Count of the threads using this block in the buffer pool. When it is zero, the block can be evicted from the buffer pool.
IS_HASHED	Whether or not a hash index has been built on this page.
NEWEST_MODIFICATION	Most recent modification's Log Sequence Number.
OLDEST_MODIFICATION	Oldest modification's Log Sequence Number.
ACCESS_TIME	Abstract number representing the time the page was first accessed.
TABLE_NAME	Table that the page belongs to.
INDEX_NAME	Index that the page belongs to, either a clustered index or a secondary index.
NUMBER_RECORDS	Number of records the page contains.
DATA_SIZE	Size in bytes of all the records contained in the page.

COMPRESSED_SIZE	Compressed size in bytes of the page, or NULL for pages that aren't compressed.
PAGE_STATE	Page state; one of FILE_PAGE (page from a file) or MEMORY (page from an in-memory object) for valid data, or one of NULL , READY_FOR_USE , NOT_USED , REMOVE_HASH .
IO_FIX	Whether there is I/O pending for the page; one of IO_NONE (no pending I/O), IO_READ (read pending), IO_WRITE (write pending).
IS_OLD	Whether the page is old or not.
FREE_PAGE_CLOCK	Freed_page_clock counter, which tracks the number of blocks removed from the end of the LRU list, at the time the block was last placed at the head of the list.

The related [INFORMATION\\_SCHEMA.INNODB\\_BUFFER\\_PAGE](#) table contains the same information, but with a block id rather than LRU position.

## Example

```
DESC information_schema.innodb_buffer_page_lru;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| POOL_ID | bigint(21) unsigned | NO | | 0 | |
| LRU_POSITION | bigint(21) unsigned | NO | | 0 | |
| SPACE | bigint(21) unsigned | NO | | 0 | |
| PAGE_NUMBER | bigint(21) unsigned | NO | | 0 | |
| PAGE_TYPE | varchar(64) | YES | | NULL | |
| FLUSH_TYPE | bigint(21) unsigned | NO | | 0 | |
| FIX_COUNT | bigint(21) unsigned | NO | | 0 | |
| IS_HASHED | varchar(3) | YES | | NULL | |
| NEWEST_MODIFICATION | bigint(21) unsigned | NO | | 0 | |
| OLDEST_MODIFICATION | bigint(21) unsigned | NO | | 0 | |
| ACCESS_TIME | bigint(21) unsigned | NO | | 0 | |
| TABLE_NAME | varchar(1024) | YES | | NULL | |
| INDEX_NAME | varchar(1024) | YES | | NULL | |
| NUMBER_RECORDS | bigint(21) unsigned | NO | | 0 | |
| DATA_SIZE | bigint(21) unsigned | NO | | 0 | |
| COMPRESSED_SIZE | bigint(21) unsigned | NO | | 0 | |
| COMPRESSED | varchar(3) | YES | | NULL | |
| IO_FIX | varchar(64) | YES | | NULL | |
| IS_OLD | varchar(3) | YES | | NULL | |
| FREE_PAGE_CLOCK | bigint(21) unsigned | NO | | 0 | |
+-----+-----+-----+-----+-----+
```

```

SELECT * FROM INFORMATION_SCHEMA.INNODB_BUFFER_PAGE_LRU\G
...
***** 6. row *****
    POOL_ID: 0
    LRU_POSITION: 5
    SPACE: 0
    PAGE_NUMBER: 11
    PAGE_TYPE: INDEX
    FLUSH_TYPE: 1
    FIX_COUNT: 0
    IS_HASHED: NO
    NEWEST_MODIFICATION: 2046835
    OLDEST_MODIFICATION: 0
    ACCESS_TIME: 2585566280
    TABLE_NAME: `SYS_INDEXES`
    INDEX_NAME: CLUST_IND
    NUMBER_RECORDS: 57
    DATA_SIZE: 4016
    COMPRESSED_SIZE: 0
    COMPRESSED: NO
    IO_FIX: IO_NONE
    IS_OLD: NO
    FREE_PAGE_CLOCK: 0
...

```

## 1.1.2.9.1.1.3 Information Schema INNODB\_BUFFER\_POOL\_PAGES Table

The [Information Schema](#)

`INNODB_BUFFER_POOL_PAGES` table is a Percona enhancement, and is only available for XtraDB, not InnoDB (see [XtraDB](#) and [InnoDB](#) ). It contains a record for each page in the [buffer pool](#) .

It has the following columns:

Column	Description
PAGE_TYPE	Type of page; one of index , undo_log , inode , ibuf_free_list , allocated, bitmap , sys , trx_sys , fsp_hdr , xdes , blob , zblob , zblob2 and unknown .
SPACE_ID	Tablespace ID.
PAGE_NO	Page offset within tablespace.

LRU_POSITION	Page position in the LRU (least-recently-used) list.
FIX_COUNT	Page reference count, incremented each time the page is accessed. 0 if the page is not currently being accessed.
FLUSH_TYPE	Flush type of the most recent flush. 0 (LRU), 2 (flush_list)

## 1.1.2.9.1.1.1.4 Information Schema INNODB\_BUFFER\_POOL\_PAGES\_BLOB Table

The [Information Schema](#)

`INNODB_BUFFER_POOL_PAGES_BLOB` table is a Percona enhancement, and is only available for XtraDB, not InnoDB (see [XtraDB](#) and [InnoDB](#) ). It contains information about `buffer pool` blob pages.

It has the following columns:

Column	Description
SPACE_ID	Tablespace ID.
PAGE_NO	Page offset within tablespace.
COMPRESSED	1 if the blob contains compressed data, 0 if not.
PART_LEN	Page data length.
NEXT_PAGE_NO	Next page number.
LRU_POSITION	Page position in the LRU (least-recently-used) list.
FIX_COUNT	Page reference count, incremented each time the page is accessed. 0 if the page is not currently being accessed.
FLUSH_TYPE	Flush type of the most recent flush. 0 (LRU), 2 (flush_list)

## 1.1.2.9.1.1.1.5 Information Schema INNODB\_BUFFER\_POOL\_PAGES\_INDEX Table

The [Information Schema](#)

`INNODB_BUFFER_POOL_PAGES` table is a Percona enhancement, and is only available for XtraDB, not InnoDB (see [XtraDB](#) and [InnoDB](#) ). It contains information about `buffer pool` index pages.

It has the following columns:

Column	Description
--------	-------------

INDEX_ID	Index name
SPACE_ID	Tablespace ID
PAGE_NO	Page offset within tablespace.
N_RECS	Number of user records on the page.
DATA_SIZE	Total data size in bytes of records in the page.
HASHED	1 if the block is in the adaptive hash index, 0 if not.
ACCESS_TIME	Page's last access time.
MODIFIED	1 if the page has been modified since being loaded, 0 if not.
DIRTY	1 if the page has been modified since it was last flushed, 0 if not
OLD	1 if the page in the in the <i>old</i> blocks of the LRU (least-recently-used) list, 0 if not.
LRU_POSITION	Position in the LRU (least-recently-used) list.
FIX_COUNT	Page reference count, incremented each time the page is accessed. 0 if the page is not currently being accessed.
FLUSH_TYPE	Flush type of the most recent flush. 0 (LRU), 2 (flush_list)

## 1.1.2.9.1.1.1.6 Information Schema INNODB\_BUFFER\_POOL\_STATS Table

The [Information Schema](#)

INNODB\_BUFFER\_POOL\_STATS

table contains information about pages in the [buffer pool](#), similar to what is returned with the

`SHOW ENGINE INNODB STATUS`

statement.

The

## PROCESS

[privilege](#) is required to view the table.

It has the following columns:

Column	Description
POOL_ID	Buffer Pool identifier. From <a href="#">MariaDB 10.5.1</a> returns a value of 0, since multiple InnoDB buffer pool instances has been removed.
POOL_SIZE	Size in pages of the buffer pool.
FREE_BUFFERS	Number of free pages in the buffer pool.
DATABASE_PAGES	Total number of pages in the buffer pool.
OLD_DATABASE_PAGES	Number of pages in the <i>old</i> sublist.
MODIFIED_DATABASE_PAGES	Number of dirty pages.
PENDING_DECOMPRESS	Number of pages pending decompression.
PENDING_READS	Pending buffer pool level reads.
PENDING_FLUSH_LRU	Number of pages in the LRU pending flush.
PENDING_FLUSH_LIST	Number of pages in the flush list pending flush.
PAGES_MADE_YOUNG	Pages moved from the <i>old</i> sublist to the <i>new</i> sublist.
PAGES_NOT_MADE_YOUNG	Pages that have remained in the <i>old</i> sublist without moving to the <i>new</i> sublist.
PAGES_MADE_YOUNG_RATE	Hits that cause blocks to move to the top of the <i>new</i> sublist.
PAGES_MADE_NOT_YOUNG_RATE	Hits that do not cause blocks to move to the top of the <i>new</i> sublist due to the <a href="#">innodb_old_blocks</a> delay not being met.
NUMBER_PAGES_READ	Number of pages read.
NUMBER_PAGES_CREATED	Number of pages created.
NUMBER_PAGES_WRITTEN	Number of pages written.

PAGES_READ_RATE	Number of pages read since the last printout divided by the time elapsed, giving pages read per second.
PAGES_CREATE_RATE	Number of pages created since the last printout divided by the time elapsed, giving pages created per second.
PAGES_WRITTEN_RATE	Number of pages written since the last printout divided by the time elapsed, giving pages written per second.
NUMBER_PAGES_GET	Number of logical read requests.
HIT_RATE	Buffer pool hit rate.
YOUNG_MAKE_PER_THOUSAND_GETS	For every 1000 gets, the number of pages made young.
NOT_YOUNG_MAKE_PER_THOUSAND_GETS	For every 1000 gets, the number of pages not made young.
NUMBER_PAGES_READ_AHEAD	Number of pages read ahead.
NUMBER_READ_AHEAD_EVICTED	Number of pages read ahead by the read-ahead thread that were later evicted without being accessed by any queries.
READ_AHEAD_RATE	Pages read ahead since the last printout divided by the time elapsed, giving read-ahead rate per second.
READ_AHEAD_EVICTED_RATE	Read-ahead pages not accessed since the last printout divided by time elapsed, giving the number of read-ahead pages evicted without access per second.
LRU_IO_TOTAL	Total least-recently used I/O.
LRU_IO_CURRENT	Least-recently used I/O for the current interval.
UNCOMPRESS_TOTAL	Total number of pages decompressed.
UNCOMPRESS_CURRENT	Number of pages decompressed in the current interval

## Examples

Field	Type	Null	Key	Default	Extra
POOL_ID	bigint(21) unsigned	NO		0	
POOL_SIZE	bigint(21) unsigned	NO		0	
FREE_BUFFERS	bigint(21) unsigned	NO		0	
DATABASE_PAGES	bigint(21) unsigned	NO		0	
OLD_DATABASE_PAGES	bigint(21) unsigned	NO		0	
MODIFIED_DATABASE_PAGES	bigint(21) unsigned	NO		0	
PENDING_DECOMPRESS	bigint(21) unsigned	NO		0	
PENDING_READS	bigint(21) unsigned	NO		0	
PENDING_FLUSH_LRU	bigint(21) unsigned	NO		0	
PENDING_FLUSH_LIST	bigint(21) unsigned	NO		0	
PAGES_MADE_YOUNG	bigint(21) unsigned	NO		0	
PAGES_NOT_MADE_YOUNG	bigint(21) unsigned	NO		0	
PAGES_MADE_YOUNG_RATE	double	NO		0	
PAGES_MADE_NOT_YOUNG_RATE	double	NO		0	
NUMBER_PAGES_READ	bigint(21) unsigned	NO		0	
NUMBER_PAGES_CREATED	bigint(21) unsigned	NO		0	
NUMBER_PAGES_WRITTEN	bigint(21) unsigned	NO		0	
PAGES_READ_RATE	double	NO		0	
PAGES_CREATE_RATE	double	NO		0	
PAGES_WRITTEN_RATE	double	NO		0	
NUMBER_PAGES_GET	bigint(21) unsigned	NO		0	
HIT_RATE	bigint(21) unsigned	NO		0	
YOUNG_MAKE_PER_THOUSAND_GETS	bigint(21) unsigned	NO		0	
NOT_YOUNG_MAKE_PER_THOUSAND_GETS	bigint(21) unsigned	NO		0	
NUMBER_PAGES_READ_AHEAD	bigint(21) unsigned	NO		0	
NUMBER_READ_AHEAD_EVICTED	bigint(21) unsigned	NO		0	
READ_AHEAD_RATE	double	NO		0	
READ_AHEAD_EVICTED_RATE	double	NO		0	
LRU_IO_TOTAL	bigint(21) unsigned	NO		0	
LRU_IO_CURRENT	bigint(21) unsigned	NO		0	
UNCOMPRESS_TOTAL	bigint(21) unsigned	NO		0	
UNCOMPRESS_CURRENT	bigint(21) unsigned	NO		0	

## 1.1.2.9.1.1.1.7 Information Schema INNODB\_CHANGED\_PAGES Table

The [Information Schema](#)

### INNODB\_CHANGED\_PAGES

Table contains data about modified pages from the bitmap file. It is updated at checkpoints by the log tracking thread parsing the log, so does not contain real-time data.

The number of records is limited by the value of the [innodb\\_max\\_changed\\_pages](#) system variable.

The

#### PROCESS

[privilege](#) is required to view the table.

It has the following columns:

Column	Description
SPACE_ID	Modified page space id
PAGE_ID	Modified page id
START_LSN	Interval start after which page was changed (equal to checkpoint LSN)
END_LSN	Interval end before which page was changed (equal to checkpoint LSN)

## 1.1.2.9.1.1.1.8 Information Schema INNODB\_CMP and

# INNODB\_CMP\_RESET Tables

The

INNODB\_CMP

and

INNODB\_CMP\_RESET

tables contain status information on compression operations related to [compressed XtraDB/InnoDB tables](#).

The

PROCESS

privilege is required to query this table.

These tables contain the following columns:

Column Name	Description
PAGE_SIZE	Compressed page size, in bytes. This value is unique in the table; other values are totals which refer to pages of this size.
COMPRESS_OPS	How many times a page of the size PAGE_SIZE has been compressed. This happens when a new page is created because the compression log runs out of space. This value includes both successful operations and <i>compression failures</i> .
COMPRESS_OPS_OK	How many times a page of the size PAGE_SIZE has been successfully compressed. This value should be as close as possible to COMPRESS_OPS . If it is notably lower, either avoid compressing some tables, or increase the KEY_BLOCK_SIZE for some compressed tables.
COMPRESS_TIME	Time (in seconds) spent to compress pages of the size PAGE_SIZE . This value includes time spent in <i>compression failures</i> .
UNCOMPRESS_OPS	How many times a page of the size PAGE_SIZE has been uncompressed. This happens when an uncompressed version of a page is created in the buffer pool, or when a <i>compression failure</i> occurs.
UNCOMPRESS_TIME	Time (in seconds) spent to uncompress pages of the size PAGE_SIZE .

These tables can be used to measure the effectiveness of XtraDB/InnoDB table compression. When you have to decide a value for

KEY\_BLOCK\_SIZE

, you can create more than one version of the table (one for each candidate value) and run a realistic workload on them. Then, these tables can be used to see how the operations performed with different page sizes.

INNODB\_CMP

and

INNODB\_CMP\_RESET

have the same columns and always contain the same values, but when

INNODB\_CMP\_RESET

is queried, both the tables are cleared.

INNODB\_CMP\_RESET

can be used, for example, if a script periodically logs the performances of compression in the last period of time.

INNODB\_CMP

can be used to see the cumulated statistics.

## Examples

```

SELECT * FROM information_schema.INNODB_CMP\G
*****
1. row ****
page_size: 1024
compress_ops: 0
compress_ops_ok: 0
compress_time: 0
uncompress_ops: 0
uncompress_time: 0
...

```

## See Also

Other tables that can be used to monitor XtraDB/InnoDB compressed tables:

- [INNODB\\_CMP\\_PER\\_INDEX](#) and [INNODB\\_CMP\\_PER\\_INDEX\\_RESET](#)
- [INNODB\\_CMPPMEM](#) and [INNODB\\_CMPPMEM\\_RESET](#)

## 1.1.2.9.1.1.1.9 Information Schema INNODB\_CMPPMEM and INNODB\_CMPPMEM\_RESET Tables

The

[INNODB\\_CMPPMEM](#)

and

[INNODB\\_CMPPMEM\\_RESET](#)

tables contain status information on compressed pages in the [buffer pool](#) (see InnoDB [COMPRESSED](#) format).

The [PROCESS](#) privilege is required to query this table.

These tables contain the following columns:

Column Name	Description
PAGE_SIZE	Compressed page size, in bytes. This value is unique in the table; other values are totals which refer to pages of this size.
BUFFER_POOL_INSTANCE	Buffer Pool identifier. From <a href="#">MariaDB 10.5.1</a> returns a value of 0, since multiple InnoDB buffer pool instances has been removed.
PAGES_USED	Number of pages of the size PAGE_SIZE which are currently in the buffer pool.
PAGES_FREE	Number of pages of the size PAGE_SIZE which are currently free, and thus are available for allocation. This value represents the buffer pool's fragmentation. A totally unfragmented buffer pool has at most 1 free page.
RELOCATION_OPS	How many times a page of the size PAGE_SIZE has been relocated. This happens when data exceeds a page (because a row must be copied into a new page) and when two pages are merged (because their data shrunk and can now be contained in one page).
RELOCATION_TIME	Time (in seconds) spent in relocation operations for pages of the size PAGE_SIZE . This column is reset when the <a href="#">INNODB_CMPPMEM_RESET</a> table is queried.

These tables can be used to measure the effectiveness of InnoDB table compression. When you have to decide a value for

[KEY\\_BLOCK\\_SIZE](#)

, you can create more than one version of the table (one for each candidate value) and run a realistic workload on them. Then, these tables can be used to see how the operations performed with different page sizes.

[INNODB\\_CMPPMEM](#)

and

[INNODB\\_CMPPMEM\\_RESET](#)

have the same columns and always contain the same values, but when

[INNODB\\_CMPPMEM\\_RESET](#)

is queried, the  
**RELOCATION\_TIME**  
column from both the tables are cleared.  
**INNODB\_CMPMEM\_RESET**  
can be used, for example, if a script periodically logs the performances of compression in the last period of time.  
**INNODB\_CMPMEM**  
can be used to see the cumulated statistics.

## Example

```
SELECT * FROM information_schema.INNODB_CMPMEM\G
*****
1. row *****
    page_size: 1024
buffer_pool_instance: 0
    pages_used: 0
    pages_free: 0
relocation_ops: 0
relocation_time: 0
```

## See Also

Other tables that can be used to monitor InnoDB compressed tables:

- [INNODB\\_CMP](#) and [INNODB\\_CMP\\_RESET](#)
- [INNODB\\_CMP\\_PER\\_INDEX](#) and [INNODB\\_CMP\\_PER\\_INDEX\\_RESET](#)

### 1.1.2.9.1.1.10 Information Schema **INNODB\_CMP\_PER\_INDEX** and **INNODB\_CMP\_PER\_INDEX\_RESET** Tables

The

[INNODB\\_CMP\\_PER\\_INDEX](#)

and

[INNODB\\_CMP\\_PER\\_INDEX\\_RESET](#)

tables contain status information on compression operations related to [compressed XtraDB/InnoDB tables](#), grouped by individual indexes.

These tables are only populated if the [innodb\\_cmp\\_per\\_index\\_enabled](#) system variable is set to

ON

The [PROCESS](#) privilege is required to query this table.

These tables contains the following columns:

Column Name	Description
<b>DATABASE_NAME</b>	Database containing the index.
<b>TABLE_NAME</b>	Table containing the index.
<b>INDEX_NAME</b>	Other values are totals which refer to this index's compression.
<b>COMPRESS_OPS</b>	How many times a page of <b>INDEX_NAME</b> has been compressed. This happens when a new page is created because the compression log runs out of space. This value includes both successful operations and <i>compression failures</i> .
<b>COMPRESS_OPS_OK</b>	How many times a page of <b>INDEX_NAME</b> has been successfully compressed. This value should be as close as possible to <b>COMPRESS_OPS</b> . If it is notably lower, either avoid compressing some tables, or increase the <b>KEY_BLOCK_SIZE</b> for some compressed tables.

COMPRESS_TIME	Time (in seconds) spent to compress pages of the size PAGE_SIZE . This value includes time spent in <i>compression failures</i> .
UNCOMPRESS_OPS	How many times a page of INDEX_NAME has been uncompressed. This happens when an uncompressed version of a page is created in the buffer pool, or when a <i>compression failure</i> occurs.
UNCOMPRESS_TIME	Time (in seconds) spent to uncompress pages of INDEX_NAME .

These tables can be used to measure the effectiveness of XtraDB/InnoDB compression, per table or per index. The values in these tables show which tables perform better with index compression, and which tables cause too many *compression failures* or perform too many compression/uncompression operations. When compression performs badly for a table, this might mean that you should change its

KEY\_BLOCK\_SIZE  
, or that the table should not be compressed.

INNODB\_CMP\_PER\_INDEX  
and  
INNODB\_CMP\_PER\_INDEX\_RESET  
have the same columns and always contain the same values, but when  
INNODB\_CMP\_PER\_INDEX\_RESET  
is queried, both the tables are cleared.  
INNODB\_CMP\_PER\_INDEX\_RESET  
can be used, for example, if a script periodically logs the performances of compression in the last period of time.  
INNODB\_CMP\_PER\_INDEX  
can be used to see the cumulated statistics.

## See Also

Other tables that can be used to monitor XtraDB/InnoDB compressed tables:

- [INNODB\\_CMP](#) and [INNODB\\_CMP\\_RESET](#)
- [INNODB\\_CMPPMEM](#) and [INNODB\\_CMPPMEM\\_RESET](#)

## 1.1.2.9.1.1.11 Information Schema INNODB\_FT\_BEING\_DELETED Table

The [Information Schema](#)

INNODB\_FT\_BEING\_DELETED  
table is only used while document ID's in the related [INNODB\\_FT\\_DELETED](#) are being removed from an InnoDB [fulltext index](#) while an [OPTIMIZE TABLE](#) is underway. At all other times the table will be empty.

The

SUPER  
[privilege](#) is required to view the table, and it also requires the [innodb\\_ft\\_aux\\_table](#) system variable to be set.

It has the following column:

Column	Description
DOC_ID	Document ID of the row being deleted. Either an underlying ID value, or a sequence value generated by InnoDB if no usable option exists.

## 1.1.2.9.1.1.12 Information Schema INNODB\_FT\_CONFIG Table

The [Information Schema](#)

INNODB\_FT\_CONFIG  
table contains InnoDB [fulltext index](#) metadata.

The

SUPER  
[privilege](#) is required to view the table, and it also requires the [innodb\\_ft\\_aux\\_table](#) system variable to be set.

It has the following columns:

Column	Description
KEY	Metadata item name.
VALUE	Associated value.

## Example

```
SELECT * FROM INNODB_FT_CONFIG;
+-----+-----+
| KEY      | VALUE |
+-----+-----+
| optimize_checkpoint_limit | 180   |
| synced_doc_id           | 6     |
| last_optimized_word     |       |
| deleted_doc_count       | 0     |
| total_word_count        |       |
| optimize_start_time     |       |
| optimize_end_time       |       |
| stopword_table_name     |       |
| use_stopword            | 1     |
| table_state              | 0     |
+-----+-----+
```

## 1.1.2.9.1.1.1.13 Information Schema INNODB\_FT\_DEFAULT\_STOPWORD Table

The [Information Schema](#)

INNODB\_FT\_DEFAULT\_STOPWORD  
table contains a list of default [stopwords](#) used when creating an InnoDB [fulltext index](#).

The

PROCESS  
[privilege](#) is required to view the table.

It has the following column:

Column	Description
VALUE	Default  stopword  for an InnoDB <a href="#">fulltext index</a> . Setting either the <a href="#">innodb_ft_server_stopword_table</a> or the <a href="#">innodb_ft_user_stopword_table</a> system variable will override this.

## Example

```
SELECT * FROM information_schema.INNODB_FT_DEFAULT_STOPWORD\G
***** 1. row *****
value: a
***** 2. row *****
value: about
***** 3. row *****
value: an
***** 4. row *****
value: are
...
***** 36. row *****
value: www
```

## 1.1.2.9.1.1.1.14 Information Schema INNODB\_FT\_DELETED Table

## The Information Schema

### INNODB\_FT\_DELETED

table contains rows that have been deleted from an InnoDB fulltext index . This information is then used to filter results on subsequent searches, removing the need to expensively reorganise the index each time a row is deleted.

The fulltext index is then only reorganized when an [OPTIMIZE TABLE](#) statement is underway. The related [INNODB\\_FT\\_BEING\\_DELETED](#) table contains rows being deleted while an

```
OPTIMIZE TABLE  
is in the process of running.
```

The

SUPER

[privilege](#) is required to view the table, and it also requires the [innodb\\_ft\\_aux\\_table](#) system variable to be set.

It has the following column:

Column	Description
DOC_ID	Document ID of the deleted row deleted. Either an underlying ID value, or a sequence value generated by InnoDB if no usable option exists.

## Example

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;  
+-----+  
| DOC_ID |  
+-----+  
|      2 |  
+-----+  
  
DELETE FROM test.ft_innodb LIMIT 1;  
  
SELECT * FROM INFORMATION_SCHEMA.INNODB_FT_DELETED;  
+-----+  
| DOC_ID |  
+-----+  
|      2 |  
|      3 |  
+-----+
```

## 1.1.2.9.1.1.1.15 Information Schema INNODB\_FT\_INDEX\_CACHE Table

### The Information Schema

#### INNODB\_FT\_INDEX\_CACHE

table contains information about rows that have recently been inserted into an InnoDB fulltext index . To avoid re-organizing the fulltext index each time a change is made, which would be very expensive, new changes are stored separately and only integrated when an [OPTIMIZE TABLE](#) is run.

The

SUPER

[privilege](#) is required to view the table, and it also requires the [innodb\\_ft\\_aux\\_table](#) system variable to be set.

It has the following columns:

Column	Description
WORD	Word from the text of a newly added row. Words can appear multiple times in the table, once per DOC_ID and POSITION combination.
FIRST_DOC_ID	First document ID where this word appears in the index.
LAST_DOC_ID	Last document ID where this word appears in the index.

DOC_COUNT	Number of rows containing this word in the index.
DOC_ID	Document ID of the newly added row, either an appropriate ID column or an internal InnoDB value.
POSITION	Position of this word instance within the DOC_ID , as an offset added to the previous POSITION instance.

Note that for

```
OPTIMIZE TABLE
to process InnoDB fulltext index data, the innodb_optimize_fulltext_only system variable needs to be set to
1
. When this is done, and an
OPTIMIZE TABLE
statement run, the
INNODB_FT_INDEX_CACHE
table will be emptied, and the INNODB_FT_INDEX_TABLE table will be updated.
```

## Examples

```
SELECT * FROM INNODB_FT_INDEX_CACHE;
+-----+-----+-----+-----+-----+
| WORD      | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+-----+-----+-----+-----+-----+
| and       |        4 |        4 |        1 |    4 |      0 |
| arrived   |        4 |        4 |        1 |    4 |     20 |
| ate       |        1 |        1 |        1 |    1 |      4 |
| everybody |        1 |        1 |        1 |    1 |      8 |
| goldilocks |        4 |        4 |        1 |    4 |      9 |
| hungry    |        3 |        3 |        1 |    3 |      8 |
| then      |        4 |        4 |        1 |    4 |      4 |
| wicked   |        2 |        2 |        1 |    2 |      4 |
| witch    |        2 |        2 |        1 |    2 |     11 |
+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

```
INSERT INTO test.ft_innodb VALUES(3,'And she ate a pear');
```

```
SELECT * FROM INNODB_FT_INDEX_CACHE;
+-----+-----+-----+-----+-----+
| WORD      | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+-----+-----+-----+-----+-----+
| and       |        4 |        5 |        2 |    4 |      0 |
| and       |        4 |        5 |        2 |    5 |      0 |
| arrived   |        4 |        4 |        1 |    4 |     20 |
| ate       |        1 |        5 |        2 |    1 |      4 |
| ate       |        1 |        5 |        2 |    5 |      8 |
| everybody |        1 |        1 |        1 |    1 |      8 |
| goldilocks |        4 |        4 |        1 |    4 |      9 |
| hungry    |        3 |        3 |        1 |    3 |      8 |
| pear      |        5 |        5 |        1 |    5 |     14 |
| she       |        5 |        5 |        1 |    5 |      4 |
| then      |        4 |        4 |        1 |    4 |      4 |
| wicked   |        2 |        2 |        1 |    2 |      4 |
| witch    |        2 |        2 |        1 |    2 |     11 |
+-----+-----+-----+-----+-----+
```

```

OPTIMIZE TABLE test.ft_innodb\G
***** 1. row *****
Table: test.ft_innodb
Op: optimize
Msg_type: note
Msg_text: Table does not support optimize, doing recreate + analyze instead
***** 2. row *****
Table: test.ft_innodb
Op: optimize
Msg_type: status
Msg_text: OK
2 rows in set (2.24 sec)

SELECT * FROM INNODB_FT_INDEX_CACHE;
+-----+-----+-----+-----+-----+-----+
| WORD      | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+-----+-----+-----+-----+-----+-----+
| and       |        4 |        5 |        2 |        4 |        0 |
| and       |        4 |        5 |        2 |        5 |        0 |
| arrived   |        4 |        4 |        1 |        4 |       20 |
| ate       |        1 |        5 |        2 |        1 |        4 |
| ate       |        1 |        5 |        2 |        5 |        8 |
| everybody |        1 |        1 |        1 |        1 |        8 |
| goldilocks |        4 |        4 |        1 |        4 |        9 |
| hungry    |        3 |        3 |        1 |        3 |        8 |
| pear      |        5 |        5 |        1 |        5 |       14 |
| she       |        5 |        5 |        1 |        5 |        4 |
| then      |        4 |        4 |        1 |        4 |        4 |
| wicked    |        2 |        2 |        1 |        2 |        4 |
| witch     |        2 |        2 |        1 |        2 |       11 |
+-----+-----+-----+-----+-----+-----+
13 rows in set (0.00 sec)

```

The

```

OPTIMIZE TABLE
statement has no effect, because the innodb\_optimize\_fulltext\_only variable wasn't set:

```

```

SHOW VARIABLES LIKE 'innodb_optimize_fulltext_only';
+-----+-----+
| Variable_name      | Value |
+-----+-----+
| innodb_optimize_fulltext_only | OFF   |
+-----+-----+

SET GLOBAL innodb_optimize_fulltext_only =1;

OPTIMIZE TABLE test.ft_innodb;
+-----+-----+-----+-----+
| Table      | Op       | Msg_type | Msg_text |
+-----+-----+-----+-----+
| test.ft_innodb | optimize | status   | OK       |
+-----+-----+-----+-----+

SELECT * FROM INNODB_FT_INDEX_CACHE;
Empty set (0.00 sec)

```

## 1.1.2.9.1.1.1.16 Information Schema INNODB\_FT\_INDEX\_TABLE Table

The [Information Schema](#)

`INNODB_FT_INDEX_TABLE`

table contains information about InnoDB [fulltext indexes](#). To avoid re-organizing the fulltext index each time a change is made, which would be very expensive, new changes are stored separately and only integrated when an `OPTIMIZE TABLE` is run. See the [INNODB\\_FT\\_INDEX\\_CACHE](#) table.

The

`SUPER`

`privilege` is required to view the table, and it also requires the `innodb_ft_aux_table` system variable to be set.

It has the following columns:

Column	Description
--------	-------------

WORD	Word from the text of a column with a fulltext index. Words can appear multiple times in the table, once per DOC_ID and POSITION combination.
FIRST_DOC_ID	First document ID where this word appears in the index.
LAST_DOC_ID	Last document ID where this word appears in the index.
DOC_COUNT	Number of rows containing this word in the index.
DOC_ID	Document ID of the newly added row, either an appropriate ID column or an internal InnoDB value.
POSITION	Position of this word instance within the DOC_ID , as an offset added to the previous POSITION instance.

Note that for

```
OPTIMIZE TABLE
to process InnoDB fulltext index data, the innodb\_optimize\_fulltext\_only system variable needs to be set to
1
. When this is done, and an
OPTIMIZE TABLE
statement run, the INNODB\_FT\_INDEX\_CACHE table will be emptied, and the
INNODB_FT_INDEX_TABLE
table will be updated.
```

## Examples

```
SELECT * FROM INNODB_FT_INDEX_TABLE;
Empty set (0.00 sec)

SET GLOBAL innodb_optimize_fulltext_only =1;

OPTIMIZE TABLE test.ft_innodb;
+-----+-----+-----+-----+
| Table | Op      | Msg_type | Msg_text |
+-----+-----+-----+-----+
| test.ft_innodb | optimize | status   | OK       |
+-----+-----+-----+-----+

SELECT * FROM INNODB_FT_INDEX_TABLE;
+-----+-----+-----+-----+-----+-----+
| WORD    | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
+-----+-----+-----+-----+-----+-----+
| and     |        4 |        5 |        2 |        4 |        0 |
| and     |        4 |        5 |        2 |        5 |        0 |
| arrived |        4 |        4 |        1 |        4 |      20 |
| ate     |        1 |        5 |        2 |        1 |        4 |
| ate     |        1 |        5 |        2 |        5 |        8 |
| everybody |        1 |        1 |        1 |        1 |        8 |
| goldilocks |        4 |        4 |        1 |        4 |        9 |
| hungry  |        3 |        3 |        1 |        3 |        8 |
| pear    |        5 |        5 |        1 |        5 |      14 |
| she    |        5 |        5 |        1 |        5 |        4 |
| then   |        4 |        4 |        1 |        4 |        4 |
| wicked |        2 |        2 |        1 |        2 |        4 |
| witch  |        2 |        2 |        1 |        2 |      11 |
+-----+-----+-----+-----+-----+
```

## 1.1.2.9.1.1.1.17 Information Schema

# INNODB\_LOCK\_WAITS Table

The [Information Schema](#)

INNODB\_LOCK\_WAITS  
table contains information about blocked InnoDB transactions. The  
PROCESS  
[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
REQUESTING_TRX_ID	Requesting transaction ID from the <a href="#">INNODB_TRX</a> table.
REQUESTED_LOCK_ID	Lock ID from the <a href="#">INNODB_LOCKS</a> table for the waiting transaction.
BLOCKING_TRX_ID	Blocking transaction ID from the <a href="#">INNODB_TRX</a> table.
BLOCKING_LOCK_ID	Lock ID from the <a href="#">INNODB_LOCKS</a> table of a lock held by a transaction that is blocking another transaction.

The table is often used in conjunction with the [INNODB\\_LOCKS](#) and [INNODB\\_TRX](#) tables to diagnose problematic locks and transactions.

## 1.1.2.9.1.1.1.18 Information Schema INNODB\_LOCKS Table

The [Information Schema](#)

INNODB\_LOCKS  
table stores information about locks that InnoDB transactions have requested but not yet acquired, or that are blocking another transaction.

It has the following columns:

Column	Description
LOCK_ID	Lock ID number - the format is not fixed, so do not rely upon the number for information.
LOCK_TRX_ID	Lock's transaction ID. Matches the <a href="#">INNODB_TRX.TRX_ID</a> column.
LOCK_MODE	<a href="#">Lock mode</a> . One of S (shared), X (exclusive), IS (intention shared), IX (intention exclusive row lock), S_GAP (shared gap lock), X_GAP (exclusive gap lock), IS_GAP (intention shared gap lock), IX_GAP (intention exclusive gap lock) or AUTO_INC ( <a href="#">auto-increment table level lock</a> ).
LOCK_TYPE	Whether the lock is RECORD (row level) or TABLE level.

LOCK_TABLE	Name of the locked table, or table containing locked rows.
LOCK_INDEX	Index name if a RECORD  , or NULL if not.
LOCK_SPACE	Tablespace ID if a RECORD  , or NULL if not.
LOCK_PAGE	Locked record page number if a RECORD  , or NULL if not.
LOCK_REC	Locked record heap number if a RECORD  , or NULL if not.
LOCK_DATA	Locked record primary key as an SQL string if a RECORD  , or NULL if not. If no primary key exists, the internal InnoDB row_id number is instead used. To avoid unnecessary IO,  also NULL if the locked record page is not in the <a href="#">buffer pool</a>

The table is often used in conjunction with the [INNODB\\_LOCK\\_WAIT](#)s and [INNODB\\_TRX](#) tables to diagnose problematic locks and transactions

## Example

```
-- session 1
START TRANSACTION;
UPDATE t SET id = 15 WHERE id = 10;

-- session 2
DELETE FROM t WHERE id = 10;

-- session 1
USE information_schema;
SELECT 1.* , t.* 
  FROM information_schema.INNODB_LOCKS l 
  JOIN information_schema.INNODB_TRX t 
    ON l.lock trx_id = t.trx_id 
   WHERE trx_state = 'LOCK WAIT' \G
*****
1. row ****
lock_id: 840:40:3:2
lock_trx_id: 840
lock_mode: X
lock_type: RECORD
lock_table: `test`.`t`
lock_index: PRIMARY
lock_space: 40
lock_page: 3
lock_rec: 2
lock_data: 10
trx_id: 840
trx_state: LOCK WAIT
trx_started: 2019-12-23 18:43:46
trx_requested_lock_id: 840:40:3:2
trx_wait_started: 2019-12-23 18:43:46
trx_weight: 2
trx_mysql_thread_id: 46
trx_query: DELETE FROM t WHERE id = 10
trx_operation_state: starting index read
trx_tables_in_use: 1
trx_tables_locked: 1
trx_lock_structs: 2
trx_lock_memory_bytes: 1136
trx_rows_locked: 1
trx_rows_modified: 0
trx_concurrency_tickets: 0
trx_isolation_level: REPEATABLE READ
trx_unique_checks: 1
trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
trx_is_read_only: 0
trx_autocommit_non_locking: 0
```

## 1.1.2.9.1.1.1.19 Information Schema INNODB\_METRICS Table

### Contents

1. [Enabling and Disabling Counters](#)
2. [Resetting Counters](#)
3. [Simplifying from MariaDB 10.6](#)
4. [Examples](#)

The [Information Schema](#)

`INNODB_METRICS`

table contains a list of useful InnoDB performance metrics. Each row in the table represents an instrumented counter that can be stopped, started and reset, and which can be grouped together by module.

The

`PROCESS`

`privilege` is required to view the table.

It has the following columns:

Column	Description
--------	-------------

NAME	Unique counter name.
SUBSYSTEM	InnoDB subsystem. See below for the matching module to use to enable/disable monitoring this subsystem with the <a href="#">innodb_monitor_enable</a> and <a href="#">innodb_monitor_disable</a> system variables.
COUNT	Count since being enabled.
MAX_COUNT	Maximum value since being enabled.
MIN_COUNT	Minimum value since being enabled.
AVG_COUNT	Average value since being enabled.
COUNT_RESET	Count since last being reset.
MAX_COUNT_RESET	Maximum value since last being reset.
MIN_COUNT_RESET	Minimum value since last being reset.
AVG_COUNT_RESET	Average value since last being reset.
TIME_ENABLED	Time last enabled.
TIME_DISABLED	Time last disabled
TIME_ELAPSED	Time since enabled
TIME_RESET	Time last reset.
STATUS	Whether the counter is currently enabled to disabled.

TYPE	Item type; one of counter ,, value ,, status_counter ,, set_owner ,, set_member .
COMMENT	Counter description.

## Enabling and Disabling Counters

Most of the counters are disabled by default. To enable them, use the `innodb_monitor_enable` system variable. You can either enable a variable by its name, for example:

```
SET GLOBAL innodb_monitor_enable = icp_match;
```

or enable a number of counters grouped by module. The

SUBSYSTEM

field indicates which counters are grouped together, but the following module names need to be used:

Module Name	Subsystem Field
module_metadata	metadata
module_lock	lock
module_buffer	buffer
module_buf_page	buffer_page_io
module_os	os
module_trx	transaction
module_purge	purge
module_compress	compression
module_file	file_system
module_index	index

module_adaptive_hash	adaptive_hash_index From <a href="#">MariaDB 10.6.2</a> , if <code>innodb_adaptive_hash_index</code> is disabled (the default), <code>adaptive_hash_index</code> will not be updated.
module_ibuf_system	change_buffer
module_srv	server
module_ddl	ddl
module_dml	dml
module_log	recovery
module_icp	icp

There are four counters in the

icp  
subsystem:

```
SELECT NAME, SUBSYSTEM FROM INNODB_METRICS WHERE SUBSYSTEM='icp';
+-----+-----+
| NAME      | SUBSYSTEM |
+-----+-----+
| icp_attempts    | icp      |
| icp_no_match    | icp      |
| icp_out_of_range | icp      |
| icp_match      | icp      |
+-----+-----+
```

To enable them all, use the associated module name from the table above,

`module_icp`

```
SET GLOBAL innodb_monitor_enable = module_icp;
```

The

`%`

wildcard, used to represent any number of characters, can also be used when naming counters, for example:

```
SET GLOBAL innodb_monitor_enable = 'buffer%'
```

To disable counters, use the `innodb_monitor_disable` system variable, using the same naming rules as described above for enabling.

Counter status is not persistent, and will be reset when the server restarts. It is possible to use the options on the command line, or the

`innodb_monitor_enable`

option only in a configuration file.

## Resetting Counters

Counters can also be reset. Resetting sets all the

`*_COUNT_RESET`

values to zero, while leaving the

`*_COUNT`

values, which perform counts since the counter was enabled, untouched. Resetting is performed with the `innodb_monitor_reset` (for individual counters) and `innodb_monitor_reset_all` (for all counters) system variables.

## Simplifying from MariaDB 10.6

MariaDB starting with 10.6

From MariaDB 10.6 , the interface was simplified by removing the following:

- buffer\_LRU\_batches\_flush
- buffer\_LRU\_batch\_flush\_pages
- buffer\_LRU\_batches\_evict
- buffer\_LRU\_batch\_evict\_pages

and by making the following reflect the status variables:

- buffer\_LRU\_batch\_flush\_total\_pages: innodb\_buffer\_pool\_pages\_LRU\_flushed
- buffer\_LRU\_batch\_evict\_total\_pages: innodb\_buffer\_pool\_pages\_LRU\_freed

The intention is to eventually remove the interface entirely (see [MDEV-15706](#) ).

## Examples

Until MariaDB 10.5 :

name	subsystem	type	comment
metadata_table_handles_opened	metadata	counter	Number of table handles opened
metadata_table_handles_closed	metadata	counter	Number of table handles closed
metadata_table_reference_count	metadata	counter	Table reference counter
lock_deadlocks	lock	counter	Number of deadlocks
lock_timeouts	lock	counter	Number of lock timeouts
lock_rec_lock_waits	lock	counter	Number of times enqueued into record lock
lock_table_lock_waits	lock	counter	Number of times enqueued into table lock w
lock_rec_lock_requests	lock	counter	Number of record locks requested
lock_rec_lock_created	lock	counter	Number of record locks created
lock_rec_lock_removed	lock	counter	Number of record locks removed from the lo
lock_rec_locks	lock	counter	Current number of record locks on tables
lock_table_lock_created	lock	counter	Number of table locks created
lock_table_lock_removed	lock	counter	Number of table locks removed from the loc
lock_table_locks	lock	counter	Current number of table locks on tables
lock_row_lock_current_waits	lock	status_counter	Number of row locks currently being waited
lock_row_lock_time	lock	status_counter	Time spent in acquiring row locks, in mill
lock_row_lock_time_max	lock	value	The maximum time to acquire a row lock, in
lock_row_lock_waits	lock	status_counter	Number of times a row lock had to be waite
lock_row_lock_time_avg	lock	value	The average time to acquire a row lock, in
buffer_pool_size	server	value	Server buffer pool size (all buffer pools)
buffer_pool_reads	buffer	status_counter	Number of reads directly from disk (innodb
buffer_pool_read_requests	buffer	status_counter	Number of logical read requests (innodb_bu
buffer_pool_write_requests	buffer	status_counter	Number of write requests (innodb_buffer_po
buffer_pool_wait_free	buffer	status_counter	Number of times waited for free buffer (in
buffer_pool_read_ahead	buffer	status_counter	Number of pages read as read ahead (innodb
buffer_pool_read_ahead_evicted	buffer	status_counter	Read-ahead pages evicted without being acc
buffer_pool_pages_total	buffer	value	Total buffer pool size in pages (innodb_bu
buffer_pool_pages_misc	buffer	value	Buffer pages for misc use such as row lock
buffer_pool_pages_data	buffer	value	Buffer pages containing data (innodb_buffe
buffer_pool_bytes_data	buffer	value	Buffer bytes containing data (innodb_buffe
buffer_pool_pages_dirty	buffer	value	Buffer pages currently dirty (innodb_buffe
buffer_pool_bytes_dirty	buffer	value	Buffer bytes currently dirty (innodb_buffe
buffer_pool_pages_free	buffer	value	Buffer pages currently free (innodb_buffer
buffer_pages_created	buffer	status_counter	Number of pages created (innodb_pages_crea
buffer_pages_written	buffer	status_counter	Number of pages written (innodb_pages_writ
buffer_index_pages_written	buffer	status_counter	Number of index pages written (innodb_inde
buffer_non_index_pages_written	buffer	status_counter	Number of non index pages written (innodb_
buffer_pages_read	buffer	status_counter	Number of pages read (innodb_pages_read)
buffer_index_sec_rec_cluster_reads	buffer	status_counter	Number of secondary record reads triggered
buffer_index_sec_rec_cluster_reads_ignored	buffer	status_counter	Number of secondary record reads avoided t
buffer_data_reads	buffer	status_counter	Amount of data read in bytes (innodb_data_
buffer_data_written	buffer	status_counter	Amount of data written in bytes (innodb_da
buffer_flush_batch_scanned	buffer	set_owner	Total pages scanned as part of flush batch
buffer_flush_batch_num_scan	buffer	set_member	Number of times buffer flush list flush is
buffer_flush_batch_scanned_per_call	buffer	set_member	Pages scanned per flush batch scan
buffer_flush_batch_total_pages	buffer	set_owner	Total pages flushed as part of flush batch
buffer_flush_batches	buffer	set_member	Number of flush batches
buffer_flush_batch_pages	buffer	set_member	Pages queued as a flush batch
buffer_flush_neighbor_total_pages	buffer	set_owner	Total neighbors flushed as part of neighbo
buffer_flush_neighbor	buffer	set_member	Number of times neighbors flushing is invo
buffer_flush_neighbor_pages	buffer	set_member	Pages queued as a neighbor batch
buffer_flush_n_to_flush_requested	buffer	counter	Number of pages requested for flushing.
buffer_flush_n_to_flush_by_age	buffer	counter	Number of pages target by LSN Age for flus

buffer_flush_adaptive_avg_time	buffer	counter	Avg time (ms) spent for adaptive flushing
buffer_flush_adaptive_avg_pass	buffer	counter	Number of adaptive flushes passed during t
buffer_LRU_get_free_loops	buffer	counter	Total loops in LRU get free.
buffer_LRU_get_free_waits	buffer	counter	Total sleep waits in LRU get free.
buffer_flush_avg_page_rate	buffer	counter	Average number of pages at which flushing
buffer_flush_lsn_avg_rate	buffer	counter	Average redo generation rate
buffer_flush_pct_for_dirty	buffer	counter	Percent of IO capacity used to avoid max d
buffer_flush_pct_for_lsn	buffer	counter	Percent of IO capacity used to avoid reusa
buffer_flush_sync_waits	buffer	counter	Number of times a wait happens due to sync
buffer_flush_adaptive_total_pages	buffer	set_owner	Total pages flushed as part of adaptive fl
buffer_flush_adaptive	buffer	set_member	Number of adaptive batches
buffer_flush_adaptive_pages	buffer	set_member	Pages queued as an adaptive batch
buffer_flush_sync_total_pages	buffer	set_owner	Total pages flushed as part of sync batch
buffer_flush_sync	buffer	set_member	Number of sync batches
buffer_flush_sync_pages	buffer	set_member	Pages queued as a sync batch
buffer_flush_background_total_pages	buffer	set_owner	Total pages flushed as part of background
buffer_flush_background	buffer	set_member	Number of background batches
buffer_flush_background_pages	buffer	set_member	Pages queued as a background batch
buffer_LRU_batch_scanned	buffer	set_owner	Total pages scanned as part of LRU batch
buffer_LRU_batch_num_scan	buffer	set_member	Number of times LRU batch is called
buffer_LRU_batch_scanned_per_call	buffer	set_member	Pages scanned per LRU batch call
buffer_LRU_batch_flush_total_pages	buffer	set_owner	Total pages flushed as part of LRU batches
buffer_LRU_batches_flush	buffer	set_member	Number of LRU batches
buffer_LRU_batch_flush_pages	buffer	set_member	Pages queued as an LRU batch
buffer_LRU_batch_evict_total_pages	buffer	set_owner	Total pages evicted as part of LRU batches
buffer_LRU_batches_evict	buffer	set_member	Number of LRU batches
buffer_LRU_batch_evict_pages	buffer	set_member	Pages queued as an LRU batch
buffer_LRU_single_flush_failure_count	Buffer	counter	Number of times attempt to flush a single
buffer_LRU_get_free_search	Buffer	counter	Number of searches performed for a clean p
buffer_LRU_search_scanned	buffer	set_owner	Total pages scanned as part of LRU search
buffer_LRU_search_num_scan	buffer	set_member	Number of times LRU search is performed
buffer_LRU_search_scanned_per_call	buffer	set_member	Page scanned per single LRU search
buffer_LRU_unzip_search_scanned	buffer	set_owner	Total pages scanned as part of LRU unzip s
buffer_LRU_unzip_search_num_scan	buffer	set_member	Number of times LRU unzip search is perfor
buffer_LRU_unzip_search_scanned_per_call	buffer	set_member	Page scanned per single LRU unzip search
buffer_page_read_index_leaf	buffer_page_io	counter	Number of Index Leaf Pages read
buffer_page_read_index_non_leaf	buffer_page_io	counter	Number of Index Non-leaf Pages read
buffer_page_read_index_ibuf_leaf	buffer_page_io	counter	Number of Insert Buffer Index Leaf Pages r
buffer_page_read_index_ibuf_non_leaf	buffer_page_io	counter	Number of Insert Buffer Index Non-Leaf Pag
buffer_page_read_undo_log	buffer_page_io	counter	Number of Undo Log Pages read
buffer_page_read_index_inode	buffer_page_io	counter	Number of Index Inode Pages read
buffer_page_read_ibuf_free_list	buffer_page_io	counter	Number of Insert Buffer Free List Pages re
buffer_page_read_ibuf_bitmap	buffer_page_io	counter	Number of Insert Buffer Bitmap Pages read
buffer_page_read_system_page	buffer_page_io	counter	Number of System Pages read
buffer_page_read_trx_system	buffer_page_io	counter	Number of Transaction System Pages read
buffer_page_read_fsp_hdr	buffer_page_io	counter	Number of File Space Header Pages read
buffer_page_read_xdes	buffer_page_io	counter	Number of Extent Descriptor Pages read
buffer_page_read_blob	buffer_page_io	counter	Number of Uncompressed BLOB Pages read
buffer_page_read_zblob	buffer_page_io	counter	Number of First Compressed BLOB Pages read
buffer_page_read_zblob2	buffer_page_io	counter	Number of Subsequent Compressed BLOB Pages
buffer_page_read_other	buffer_page_io	counter	Number of other/unknown (old version of In
buffer_page_written_index_leaf	buffer_page_io	counter	Number of Index Leaf Pages written
buffer_page_written_index_non_leaf	buffer_page_io	counter	Number of Index Non-leaf Pages written
buffer_page_written_index_ibuf_leaf	buffer_page_io	counter	Number of Insert Buffer Index Leaf Pages w
buffer_page_written_index_ibuf_non_leaf	buffer_page_io	counter	Number of Insert Buffer Index Non-Leaf Pag
buffer_page_written_undo_log	buffer_page_io	counter	Number of Undo Log Pages written
buffer_page_written_index_inode	buffer_page_io	counter	Number of Index Inode Pages written
buffer_page_written_ibuf_free_list	buffer_page_io	counter	Number of Insert Buffer Free List Pages wr
buffer_page_written_ibuf_bitmap	buffer_page_io	counter	Number of Insert Buffer Bitmap Pages writt
buffer_page_written_system_page	buffer_page_io	counter	Number of System Pages written
buffer_page_written_trx_system	buffer_page_io	counter	Number of Transaction System Pages written
buffer_page_written_fsp_hdr	buffer_page_io	counter	Number of File Space Header Pages written
buffer_page_written_xdes	buffer_page_io	counter	Number of Extent Descriptor Pages written
buffer_page_written_blob	buffer_page_io	counter	Number of Uncompressed BLOB Pages written
buffer_page_written_zblob	buffer_page_io	counter	Number of First Compressed BLOB Pages writ
buffer_page_written_zblob2	buffer_page_io	counter	Number of Subsequent Compressed BLOB Pages
buffer_page_written_other	buffer_page_io	counter	Number of other/unknown (old version InnoDB)
os_data_reads	os	status_counter	Number of reads initiated (innodb_data_rea
os_data_writes	os	status_counter	Number of writes initiated (innodb_data_wr
os_data_fsyncs	os	status_counter	Number of fsync() calls (innodb_data_fsync
os_pending_reads	os	counter	Number of reads pending
os_pending_writes	os	counter	Number of writes pending
os_log_bytes_written	os	status_counter	Bytes of log written (innodb_os_log_writte
os_log_fsyncs	os	status_counter	Number of fsync log writes (innodb_os_log_
os_log_pending_fsyncs	os	status_counter	Number of pending fsync write (innodb_os_l
os_log_pending_writes	os	status_counter	Number of pending log file writes (innodb_
trx_rw_commits	transaction	counter	Number of read-write transactions committ
trx_ro_commits	transaction	counter	Number of read-only transactions committed

trx_nl_ro_commits	transaction	counter	Number of non-locking auto-commit read-onl
trx_commits_insert_update	transaction	counter	Number of transactions committed with inse
trx_rollbacks	transaction	counter	Number of transactions rolled back
trx_rollbacks_savepoint	transaction	counter	Number of transactions rolled back to save
trx_active_transactions	transaction	counter	Number of active transactions
trx_rseg_history_len	transaction	value	Length of the TRX_RSEG_HISTORY list
trx_undo_slots_used	transaction	counter	Number of undo slots used
trx_undo_slots_cached	transaction	counter	Number of undo slots cached
trx_rseg_current_size	transaction	value	Current rollback segment size in pages
purge_del_mark_records	purge	counter	Number of delete-marked rows purged
purge_upd_exist_or_extern_records	purge	counter	Number of purges on updates of existing re
purge_invoked	purge	counter	Number of times purge was invoked
purge_undo_log_pages	purge	counter	Number of undo log pages handled by the pu
purge_dml_delay_usec	purge	value	Microseconds DML to be delayed due to purg
purge_stop_count	purge	value	Number of times purge was stopped
purge_resume_count	purge	value	Number of times purge was resumed
log_checkpoints	recovery	counter	Number of checkpoints
log_lsn_last_flush	recovery	value	LSN of Last flush
log_lsn_last_checkpoint	recovery	value	LSN at last checkpoint
log_lsn_current	recovery	value	Current LSN value
log_lsn_checkpoint_age	recovery	value	Current LSN value minus LSN at last checkp
log_lsn_buf_pool_oldest	recovery	value	The oldest modified block LSN in the buffe
log_max_modified_age_async	recovery	value	Maximum LSN difference; when exceeded, sta
log_pending_log_flushes	recovery	value	Pending log flushes
log_pending_checkpoint_writes	recovery	value	Pending checkpoints
log_num_log_io	recovery	value	Number of log I/Os
log_waits	recovery	status_counter	Number of log waits due to small log buffe
log_write_requests	recovery	status_counter	Number of log write requests (innodb_log_w
log_writes	recovery	status_counter	Number of log writes (innodb_log_writes)
log_padded	recovery	status_counter	Bytes of log padded for log write ahead
compress_pages_compressed	compression	counter	Number of pages compressed
compress_pages_decompressed	compression	counter	Number of pages decompressed
compression_pad_increments	compression	counter	Number of times padding is incremented to
compression_pad_decrements	compression	counter	Number of times padding is decremented due
compress_saved	compression	counter	Number of bytes saved by page compression
compress_pages_page_compressed	compression	counter	Number of pages compressed by page compres
compress_page_compressed_trim_op	compression	counter	Number of TRIM operation performed by page
compress_pages_page_decompressed	compression	counter	Number of pages decompressed by page compr
compress_pages_page_compression_error	compression	counter	Number of page compression errors
compress_pages_encrypted	compression	counter	Number of pages encrypted
compress_pages_decrypted	compression	counter	Number of pages decrypted
index_page_splits	index	counter	Number of index page splits
index_page_merge_attempts	index	counter	Number of index page merge attempts
index_page_merge_successful	index	counter	Number of successful index page merges
index_page_reorg_attempts	index	counter	Number of index page reorganization attemp
index_page_reorg_successful	index	counter	Number of successful index page reorganiza
index_page_discards	index	counter	Number of index pages discarded
adaptive_hash_searches	adaptive_hash_index	status_counter	Number of successful searches using Adapti
adaptive_hash_searches_btree	adaptive_hash_index	status_counter	Number of searches using B-tree on an inde
adaptive_hash_pages_added	adaptive_hash_index	counter	Number of index pages on which the Adaptiv
adaptive_hash_pages_removed	adaptive_hash_index	counter	Number of index pages whose corresponding
adaptive_hash_rows_added	adaptive_hash_index	counter	Number of Adaptive Hash Index rows added
adaptive_hash_rows_removed	adaptive_hash_index	counter	Number of Adaptive Hash Index rows removed
adaptive_hash_rows_deleted_no_hash_entry	adaptive_hash_index	counter	Number of rows deleted that did not have c
adaptive_hash_rows_updated	adaptive_hash_index	counter	Number of Adaptive Hash Index rows updated
file_num_open_files	file_system	value	Number of files currently open (innodb_num
ibuf_merges_insert	change_buffer	status_counter	Number of inserted records merged by chang
ibuf_merges_delete_mark	change_buffer	status_counter	Number of deleted records merged by change
ibuf_merges_delete	change_buffer	status_counter	Number of purge records merged by change b
ibuf_merges_discard_insert	change_buffer	status_counter	Number of insert merged operations discard
ibuf_merges_discard_delete_mark	change_buffer	status_counter	Number of deleted merged operations discar
ibuf_merges_discard_delete	change_buffer	status_counter	Number of purge merged operations discard
ibuf_merges	change_buffer	status_counter	Number of change buffer merges
ibuf_size	change_buffer	status_counter	Change buffer size in pages
innodb_master_thread_sleeps	server	counter	Number of times (seconds) master thread sl
innodb_activity_count	server	status_counter	Current server activity count
innodb_master_active_loops	server	counter	Number of times master thread performs its
innodb_master_idle_loops	server	counter	Number of times master thread performs its
innodb_background_drop_table_usec	server	counter	Time (in microseconds) spent to process dr
innodb_log_flush_usec	server	counter	Time (in microseconds) spent to flush log
innodb_dict_lru_usec	server	counter	Time (in microseconds) spent to process DI
innodb_dict_lru_count_active	server	counter	Number of tables evicted from DICT LRU lis
innodb_dict_lru_count_idle	server	counter	Number of tables evicted from DICT LRU lis
innodb dblwr_writes	server	status_counter	Number of doublewrite operations that have
innodb dblwr_pages_written	server	status_counter	Number of pages that have been written for
innodb_page_size	server	value	InnoDB page size in bytes (innodb_page_siz
innodb_rwlock_s_spin_waits	server	status_counter	Number of rwlock spin waits due to shared
innodb_rwlock_x_spin_waits	server	status_counter	Number of rwlock spin waits due to exclusive

innodb_rwlock_x_spin_waits	server	status_counter	Number of rwlock spin waits due to exclusive
innodb_rwlock_sx_spin_waits	server	status_counter	Number of rwlock spin waits due to sx latch
innodb_rwlock_s_spin_rounds	server	status_counter	Number of rwlock spin loop rounds due to s
innodb_rwlock_x_spin_rounds	server	status_counter	Number of rwlock spin loop rounds due to e
innodb_rwlock_sx_spin_rounds	server	status_counter	Number of rwlock spin loop rounds due to s
innodb_rwlock_s_os_waits	server	status_counter	Number of OS waits due to shared latch req
innodb_rwlock_x_os_waits	server	status_counter	Number of OS waits due to exclusive latch
innodb_rwlock_sx_os_waits	server	status_counter	Number of OS waits due to sx latch request
dml_reads	dml	status_counter	Number of rows read
dml_inserts	dml	status_counter	Number of rows inserted
dml_deletes	dml	status_counter	Number of rows deleted
dml_updates	dml	status_counter	Number of rows updated
dml_system_reads	dml	status_counter	Number of system rows read
dml_system_inserts	dml	status_counter	Number of system rows inserted
dml_system Deletes	dml	status_counter	Number of system rows deleted
dml_system_updates	dml	status_counter	Number of system rows updated
ddl_background_drop_indexes	ddl	counter	Number of indexes waiting to be dropped af
ddl_background_drop_tables	ddl	counter	Number of tables in background drop table
ddl_online_create_index	ddl	counter	Number of indexes being created online
ddl_pending_alter_table	ddl	counter	Number of ALTER TABLE, CREATE INDEX, DROP
ddl_sort_file_alter_table	ddl	counter	Number of sort files created during alter
ddl_log_file_alter_table	ddl	counter	Number of log files created during alter t
icp_attempts	icp	counter	Number of attempts for index push-down con
icp_no_match	icp	counter	Index push-down condition does not match
icp_out_of_range	icp	counter	Index push-down condition out of range
icp_match	icp	counter	Index push-down condition matches

234 rows in set (0.001 sec)

From MariaDB 10.6

name	subsystem	type	comment
metadata_table_handles_opened	metadata	counter	Number of table handles opened
lock_deadlocks	lock	value	Number of deadlocks
lock_timeouts	lock	value	Number of lock timeouts
lock_rec_lock_waits	lock	counter	Number of times enqueued into record lock
lock_table_lock_waits	lock	counter	Number of times enqueued into table lock w
lock_rec_lock_requests	lock	counter	Number of record locks requested
lock_rec_lock_created	lock	counter	Number of record locks created
lock_rec_lock_removed	lock	counter	Number of record locks removed from the lo
lock_rec_locks	lock	counter	Current number of record locks on tables
lock_table_lock_created	lock	counter	Number of table locks created
lock_table_lock_removed	lock	counter	Number of table locks removed from the loc
lock_table_locks	lock	counter	Current number of table locks on tables
lock_row_lock_current_waits	lock	status_counter	Number of row locks currently being waited
lock_row_lock_time	lock	status_counter	Time spent in acquiring row locks, in mill
lock_row_lock_time_max	lock	value	The maximum time to acquire a row lock, in
lock_row_lock_waits	lock	status_counter	Number of times a row lock had to be waited
lock_row_lock_time_avg	lock	value	The average time to acquire a row lock, in
buffer_pool_size	server	value	Server buffer pool size (all buffer pools)
buffer_pool_reads	buffer	status_counter	Number of reads directly from disk (innodb bu
buffer_pool_read_requests	buffer	status_counter	Number of logical read requests (innodb bu
buffer_pool_write_requests	buffer	status_counter	Number of write requests (innodb_buffer_po
buffer_pool_wait_free	buffer	status_counter	Number of times waited for free buffer (in
buffer_pool_read_ahead	buffer	status_counter	Number of pages read as read ahead (innodb bu
buffer_pool_read_ahead_evicted	buffer	status_counter	Read-ahead pages evicted without being acc
buffer_pool_pages_total	buffer	value	Total buffer pool size in pages (innodb_bu
buffer_pool_pages_misc	buffer	value	Buffer pages for misc use such as row lock
buffer_pool_pages_data	buffer	value	Buffer pages containing data (innodb_buffe
buffer_pool_bytes_data	buffer	value	Buffer bytes containing data (innodb_buffe
buffer_pool_pages_dirty	buffer	value	Buffer pages currently dirty (innodb_buffe
buffer_pool_bytes_dirty	buffer	value	Buffer bytes currently dirty (innodb_buffe
buffer_pool_pages_free	buffer	value	Buffer pages currently free (innodb_buffer
buffer_pages_created	buffer	status_counter	Number of pages created (innodb_pages_crea
buffer_pages_written	buffer	status_counter	Number of pages written (innodb_pages_writ
buffer_index_pages_written	buffer	status_counter	Number of index pages written (innodb_inde
buffer_non_index_pages_written	buffer	status_counter	Number of non index pages written (innodb_
buffer_pages_read	buffer	status_counter	Number of pages read (innodb_pages_read)
buffer_index_sec_rec_cluster_reads	buffer	status_counter	Number of secondary record reads triggered
buffer_index_sec_rec_cluster_reads_avoided	buffer	status_counter	Number of secondary record reads avoided t
buffer_data_reads	buffer	status_counter	Amount of data read in bytes (innodb_da
buffer_data_written	buffer	status_counter	Amount of data written in bytes (innodb_da
buffer_flush_batch_scanned	buffer	set_owner	Total pages scanned as part of flush batch
buffer_flush_batch_num_scan	buffer	set_member	Number of times buffer flush list flush is

buffer_flush_batch_scanned_per_call	buffer	set_member	Pages scanned per flush batch scan
buffer_flush_batch_total_pages	buffer	set_owner	Total pages flushed as part of flush batch
buffer_flush_batches	buffer	set_member	Number of flush batches
buffer_flush_batch_pages	buffer	set_member	Pages queued as a flush batch
buffer_flush_neighbor_total_pages	buffer	set_owner	Total neighbors flushed as part of neighbor
buffer_flush_neighbor	buffer	set_member	Number of times neighbors flushing is involved
buffer_flush_neighbor_pages	buffer	set_member	Pages queued as a neighbor batch
buffer_flush_n_to_flush_requested	buffer	counter	Number of pages requested for flushing.
buffer_flush_n_to_flush_by_age	buffer	counter	Number of pages target by LSN Age for flush
buffer_flush_adaptive_avg_time	buffer	counter	Avg time (ms) spent for adaptive flushing
buffer_flush_adaptive_avg_pass	buffer	counter	Number of adaptive flushes passed during time
buffer_LRU_get_free_loops	buffer	counter	Total loops in LRU get free.
buffer_LRU_get_free_waits	buffer	counter	Total sleep waits in LRU get free.
buffer_flush_avg_page_rate	buffer	counter	Average number of pages at which flushing
buffer_flush_lsn_avg_rate	buffer	counter	Average redo generation rate
buffer_flush_pct_for_dirty	buffer	counter	Percent of IO capacity used to avoid max dirty
buffer_flush_pct_for_lsn	buffer	counter	Percent of IO capacity used to avoid reuse
buffer_flush_sync_waits	buffer	counter	Number of times a wait happens due to sync
buffer_flush_adaptive_total_pages	buffer	set_owner	Total pages flushed as part of adaptive flush
buffer_flush_adaptive	buffer	set_member	Number of adaptive batches
buffer_flush_adaptive_pages	buffer	set_member	Pages queued as an adaptive batch
buffer_flush_sync_total_pages	buffer	set_owner	Total pages flushed as part of sync batches
buffer_flush_sync	buffer	set_member	Number of sync batches
buffer_flush_sync_pages	buffer	set_member	Pages queued as a sync batch
buffer_flush_background_total_pages	buffer	set_owner	Total pages flushed as part of background
buffer_flush_background	buffer	set_member	Number of background batches
buffer_flush_background_pages	buffer	set_member	Pages queued as a background batch
buffer_LRU_batch_scanned	buffer	set_owner	Total pages scanned as part of LRU batch
buffer_LRU_batch_num_scan	buffer	set_member	Number of times LRU batch is called
buffer_LRU_batch_scanned_per_call	buffer	set_member	Pages scanned per LRU batch call
buffer_LRU_batch_flush_total_pages	buffer	status_counter	Total pages flushed as part of LRU batches
buffer_LRU_batch_evict_total_pages	buffer	status_counter	Total pages evicted as part of LRU batches
buffer_LRU_single_flush_failure_count	Buffer	counter	Number of times attempt to flush a single page
buffer_LRU_get_free_search	Buffer	counter	Number of searches performed for a clean page
buffer_LRU_search_scanned	buffer	set_owner	Total pages scanned as part of LRU search
buffer_LRU_search_num_scan	buffer	set_member	Number of times LRU search is performed
buffer_LRU_search_scanned_per_call	buffer	set_member	Page scanned per single LRU search
buffer_LRU_unzip_search_scanned	buffer	set_owner	Total pages scanned as part of LRU unzip search
buffer_LRU_unzip_search_num_scan	buffer	set_member	Number of times LRU unzip search is performed
buffer_LRU_unzip_search_scanned_per_call	buffer	set_member	Page scanned per single LRU unzip search
buffer_page_read_index_leaf	buffer_page_io	counter	Number of Index Leaf Pages read
buffer_page_read_index_non_leaf	buffer_page_io	counter	Number of Index Non-leaf Pages read
buffer_page_read_index_ibuf_leaf	buffer_page_io	counter	Number of Insert Buffer Index Leaf Pages read
buffer_page_read_index_ibuf_non_leaf	buffer_page_io	counter	Number of Insert Buffer Index Non-Leaf Pages read
buffer_page_read_undo_log	buffer_page_io	counter	Number of Undo Log Pages read
buffer_page_read_index_inode	buffer_page_io	counter	Number of Index Inode Pages read
buffer_page_read_ibuf_free_list	buffer_page_io	counter	Number of Insert Buffer Free List Pages read
buffer_page_read_ibuf_bitmap	buffer_page_io	counter	Number of Insert Buffer Bitmap Pages read
buffer_page_read_system_page	buffer_page_io	counter	Number of System Pages read
buffer_page_read_trx_system	buffer_page_io	counter	Number of Transaction System Pages read
buffer_page_read_fsp_hdr	buffer_page_io	counter	Number of File Space Header Pages read
buffer_page_read_xdes	buffer_page_io	counter	Number of Extent Descriptor Pages read
buffer_page_read_blob	buffer_page_io	counter	Number of Uncompressed BLOB Pages read
buffer_page_read_zblob	buffer_page_io	counter	Number of First Compressed BLOB Pages read
buffer_page_read_zblob2	buffer_page_io	counter	Number of Subsequent Compressed BLOB Pages read
buffer_page_read_other	buffer_page_io	counter	Number of other/unknown (old version of InnoDB)
buffer_page_written_index_leaf	buffer_page_io	counter	Number of Index Leaf Pages written
buffer_page_written_index_non_leaf	buffer_page_io	counter	Number of Index Non-leaf Pages written
buffer_page_written_index_ibuf_leaf	buffer_page_io	counter	Number of Insert Buffer Index Leaf Pages written
buffer_page_written_index_ibuf_non_leaf	buffer_page_io	counter	Number of Insert Buffer Index Non-Leaf Pages written
buffer_page_written_undo_log	buffer_page_io	counter	Number of Undo Log Pages written
buffer_page_written_index_inode	buffer_page_io	counter	Number of Index Inode Pages written
buffer_page_written_ibuf_free_list	buffer_page_io	counter	Number of Insert Buffer Free List Pages written
buffer_page_written_ibuf_bitmap	buffer_page_io	counter	Number of Insert Buffer Bitmap Pages written
buffer_page_written_system_page	buffer_page_io	counter	Number of System Pages written
buffer_page_written_trx_system	buffer_page_io	counter	Number of Transaction System Pages written
buffer_page_written_fsp_hdr	buffer_page_io	counter	Number of File Space Header Pages written
buffer_page_written_xdes	buffer_page_io	counter	Number of Extent Descriptor Pages written
buffer_page_written_blob	buffer_page_io	counter	Number of Uncompressed BLOB Pages written
buffer_page_written_zblob	buffer_page_io	counter	Number of First Compressed BLOB Pages written
buffer_page_written_zblob2	buffer_page_io	counter	Number of Subsequent Compressed BLOB Pages written
buffer_page_written_other	buffer_page_io	counter	Number of other/unknown (old version InnoDB)
os_data_reads	os	status_counter	Number of reads initiated (innodb_data_reads)
os_data_writes	os	status_counter	Number of writes initiated (innodb_data_writes)
os_data_fsyncs	os	status_counter	Number of fsync() calls (innodb_data_fsyncs)
os_pending_reads	os	counter	Number of reads pending
os_pending_writes	os	counter	Number of writes pending
os_log_bytes_written	os	status_counter	Bytes of log written (innodb_log_written)

innodb_log_bytes_written	os	status_counter	Bytes of log written (innodb_os_log_written)
os_log_fsyncs	os	status_counter	Number of fsync log writes (innodb_os_log_fsyncs)
os_log_pending_fsyncs	os	status_counter	Number of pending fsync write (innodb_os_log_pending_fsyncs)
os_log_pending_writes	os	status_counter	Number of pending log file writes (innodb_os_log_pending_writes)
trx_rw_commits	transaction	counter	Number of read-write transactions committed
trx_ro_commits	transaction	counter	Number of read-only transactions committed
trx_nl_ro_commits	transaction	counter	Number of non-locking auto-commit read-only transactions committed
trx_commits_insert_update	transaction	counter	Number of transactions committed with insert or update
trx_rollbacks	transaction	counter	Number of transactions rolled back
trx_rollbacks_savepoint	transaction	counter	Number of transactions rolled back to save point
trx_rseg_history_len	transaction	value	Length of the TRX_RSEG_HISTORY list
trx_undo_slots_used	transaction	counter	Number of undo slots used
trx_undo_slots_cached	transaction	counter	Number of undo slots cached
trx_rseg_current_size	transaction	value	Current rollback segment size in pages
purge_del_mark_records	purge	counter	Number of delete-marked rows purged
purge_upd_exist_or_extern_records	purge	counter	Number of purges on updates of existing records
purge_invoked	purge	counter	Number of times purge was invoked
purge_undo_log_pages	purge	counter	Number of undo log pages handled by the purge operation
purge_dml_delay_usec	purge	value	Microseconds DML to be delayed due to purge
purge_stop_count	purge	value	Number of times purge was stopped
purge_resume_count	purge	value	Number of times purge was resumed
log_checkpoints	recovery	counter	Number of checkpoints
log_lsn_last_flush	recovery	value	LSN of Last flush
log_lsn_last_checkpoint	recovery	value	LSN at last checkpoint
log_lsn_current	recovery	value	Current LSN value
log_lsn_checkpoint_age	recovery	value	Current LSN value minus LSN at last checkpoint
log_lsn_buf_pool_oldest	recovery	value	The oldest modified block LSN in the buffer pool
log_max_modified_age_async	recovery	value	Maximum LSN difference; when exceeded, start a log flush
log_pending_log_flushes	recovery	value	Pending log flushes
log_pending_checkpoint_writes	recovery	value	Pending checkpoints
log_num_log_io	recovery	value	Number of log I/Os
log_waits	recovery	status_counter	Number of log waits due to small log buffer pool
log_write_requests	recovery	status_counter	Number of log write requests (innodb_log_write_requests)
log_writes	recovery	status_counter	Number of log writes (innodb_log_writes)
log_padded	recovery	status_counter	Bytes of log padded for log write ahead
compress_pages_compressed	compression	counter	Number of pages compressed
compress_pages_decompressed	compression	counter	Number of pages decompressed
compression_pad_increments	compression	counter	Number of times padding is incremented due to page compression
compression_pad_decrements	compression	counter	Number of times padding is decremented due to page compression
compress_saved	compression	counter	Number of bytes saved by page compression
compress_pages_page_compressed	compression	counter	Number of pages compressed by page compression
compress_page_compressed_trim_op	compression	counter	Number of TRIM operation performed by page compression
compress_pages_page_decompressed	compression	counter	Number of pages decompressed by page compression
compress_pages_page_compression_error	compression	counter	Number of page compression errors
compress_pages_encrypted	compression	counter	Number of pages encrypted
compress_pages_decrypted	compression	counter	Number of pages decrypted
index_page_splits	index	counter	Number of index page splits
index_page_merge_attempts	index	counter	Number of index page merge attempts
index_page_merge_successful	index	counter	Number of successful index page merges
index_page_reorg_attempts	index	counter	Number of index page reorganization attempts
index_page_reorg_successful	index	counter	Number of successful index page reorganization attempts
index_page_discards	index	counter	Number of index pages discarded
adaptive_hash_searches	adaptive_hash_index	status_counter	Number of successful searches using Adaptive Hash Index
adaptive_hash_searches_btree	adaptive_hash_index	status_counter	Number of searches using B-tree on an index
adaptive_hash_pages_added	adaptive_hash_index	counter	Number of index pages on which the Adaptive Hash Index was added
adaptive_hash_pages_removed	adaptive_hash_index	counter	Number of index pages whose corresponding Adaptive Hash Index was removed
adaptive_hash_rows_added	adaptive_hash_index	counter	Number of Adaptive Hash Index rows added
adaptive_hash_rows_removed	adaptive_hash_index	counter	Number of Adaptive Hash Index rows removed
adaptive_hash_rows_deleted_no_hash_entry	adaptive_hash_index	counter	Number of rows deleted that did not have a corresponding hash entry
adaptive_hash_rows_updated	adaptive_hash_index	counter	Number of Adaptive Hash Index rows updated
file_num_open_files	file_system	value	Number of files currently open (innodb_num_open_files)
ibuf_merges_insert	change_buffer	status_counter	Number of inserted records merged by change buffer
ibuf_merges_delete_mark	change_buffer	status_counter	Number of deleted records merged by change buffer
ibuf_merges_delete	change_buffer	status_counter	Number of purge records merged by change buffer
ibuf_merges_discard_insert	change_buffer	status_counter	Number of insert merged operations discard
ibuf_merges_discard_delete_mark	change_buffer	status_counter	Number of deleted merged operations discard
ibuf_merges_discard_delete	change_buffer	status_counter	Number of purge merged operations discard
ibuf_merges	change_buffer	status_counter	Number of change buffer merges
ibuf_size	change_buffer	status_counter	Change buffer size in pages
innodb_master_thread_sleeps	server	counter	Number of times (seconds) master thread slept
innodb_activity_count	server	status_counter	Current server activity count
innodb_master_active_loops	server	counter	Number of times master thread performs its active loops
innodb_master_idle_loops	server	counter	Number of times master thread performs its idle loops
innodb_log_flush_usec	server	counter	Time (in microseconds) spent to flush log
innodb_dict_lru_usec	server	counter	Time (in microseconds) spent to process DICT LRU
innodb_dict_lru_count_active	server	counter	Number of tables evicted from DICT LRU list
innodb_dict_lru_count_idle	server	counter	Number of tables evicted from DICT LRU list
innodb dblwr_writes	server	status_counter	Number of doublewrite operations that have been written

```

| innodb dblwr_pages_written | server | status_counter | Number of pages that have been written for
| innodb_page_size           | server | value          | InnoDB page size in bytes (innodb_page_size)
| dml_reads                  | dml   | status_counter | Number of rows read
| dml_inserts                | dml   | status_counter | Number of rows inserted
| dml_deletes                | dml   | status_counter | Number of rows deleted
| dml_updates                | dml   | status_counter | Number of rows updated
| dml_system_reads           | dml   | status_counter | Number of system rows read
| dml_system_inserts          | dml   | status_counter | Number of system rows inserted
| dml_system_deletes          | dml   | status_counter | Number of system rows deleted
| dml_system_updates          | dml   | status_counter | Number of system rows updated
| ddl_background_drop_indexes | ddl   | counter        | Number of indexes waiting to be dropped af
| ddl_online_create_index    | ddl   | counter        | Number of indexes being created online
| ddl_pending_alter_table    | ddl   | counter        | Number of ALTER TABLE, CREATE INDEX, DROP
| ddl_sort_file_alter_table  | ddl   | counter        | Number of sort files created during alter
| ddl_log_file_alter_table   | ddl   | counter        | Number of log files created during alter t
| icp_attempts               | icp   | counter        | Number of attempts for index push-down con
| icp_no_match               | icp   | counter        | Index push-down condition does not match
| icp_out_of_range           | icp   | counter        | Index push-down condition out of range
| icp_match                  | icp   | counter        | Index push-down condition matches
+-----+-----+-----+
216 rows in set (0.000 sec)

```

## 1.1.2.9.1.1.1.20 Information Schema INNODB\_MUTEXES Table

The

### INNODB\_MUTEXES

table monitors mutex and rw locks waits. It has the following columns:

Column	Description
NAME	Name of the lock, as it appears in the source code.
CREATE_FILE	File name of the mutex implementation.
CREATE_LINE	Line number of the mutex implementation.
OS_WAITS	How many times the mutex occurred.

The

CREATE\_FILE

and

CREATE\_LINE

columns depend on the InnoDB/XtraDB version.

Note that since [MariaDB 10.2.2](#), the table has only been providing information about `rw_lock_t`, not any mutexes. From [MariaDB 10.2.2](#) until [MariaDB 10.2.32](#), [MariaDB 10.3.23](#), [MariaDB 10.4.13](#) and [MariaDB 10.5.1](#), the

NAME

column was not populated ([MDEV-21636](#)).

The [SHOW ENGINE INNODB STATUS](#) statement provides similar information.

## Examples

```

SELECT * FROM INNODB_MUTEXES;
+-----+-----+-----+-----+
| NAME           | CREATE_FILE      | CREATE_LINE | OS_WAITS |
+-----+-----+-----+-----+
| &dict_sys->mutex      | dict0dict.cc    |      989 |      2 |
| &buf_pool->flush_state_mutex | buf0buf.cc     |     1388 |      1 |
| &log_sys->checkpoint_lock | log0log.cc      |     1014 |      2 |
| &block->lock          | combined buf0buf.cc |    1120 |      1 |
+-----+-----+-----+-----+

```

## 1.1.2.9.1.1.1.21 Information Schema INNODB\_SYS\_COLUMNS Table

The [Information Schema](#)

INNODB\_SYS\_COLUMNS  
table contains information about InnoDB fields.

The

PROCESS  
privilege is required to view the table.

It has the following columns:

Column	Description
TABLE_ID	Table identifier, matching the value from <a href="#">INNODB_SYS_TABLES.TABLE_ID</a> .
NAME	Column name.
POS	Ordinal position of the column in the table, starting from 0. This value is adjusted when columns are added or removed.
MTYPE	Numeric column type identifier, (see the table below for an explanation of its values).
PRTYPE	Binary value of the InnoDB precise type, representing the data type, character set code and nullability.
LEN	Column length. For multi-byte character sets, represents the length in bytes.

The column

[MTYPE](#)  
uses a numeric column type identifier, which has the following values:

Column Type Identifier	Description
1	VARCHAR
2	CHAR
3	FIXBINARY
4	BINARY
5	BLOB

6	INT
7	SYS_CHILD
8	SYS
9	FLOAT
10	DOUBLE
11	DECIMAL
12	VARMYSQL
13	MYSQL

## Example

```

SELECT * FROM information_schema.INNODB_SYS_COLUMNS LIMIT 3\G
*****
1. row ****
TABLE_ID: 11
  NAME: ID
  POS: 0
  MTYPE: 1
  PRTYPE: 524292
  LEN: 0
*****
2. row ****
TABLE_ID: 11
  NAME: FOR_NAME
  POS: 0
  MTYPE: 1
  PRTYPE: 524292
  LEN: 0
*****
3. row ****
TABLE_ID: 11
  NAME: REF_NAME
  POS: 0
  MTYPE: 1
  PRTYPE: 524292
  LEN: 0
3 rows in set (0.00 sec)

```

## 1.1.2.9.1.1.1.22 Information Schema INNODB\_SYS\_DATAFILES Table

MariaDB until 10.5

The

**INNODB\_SYS\_DATAFILES**  
table was added in [MariaDB 10.0.4](#) , and removed in [MariaDB 10.6.0](#) .

The [Information Schema](#)

**INNODB\_SYS\_DATAFILES**  
table contains information about InnoDB datafile paths. It was intended to provide metadata for tablespaces inside InnoDB tables, which was never implemented in MariaDB and was removed in [MariaDB 10.6](#) . The [PROCESS privilege](#) is required to view the table.

It contains the following columns:

Column	Description
SPACE	Numeric tablespace. Matches the <a href="#">INNODB_SYS_TABLES.SPACE</a> value.
PATH	Tablespace datafile path.

## Example

```
SELECT * FROM INNODB_SYS_DATAFILES;
+-----+-----+
| SPACE | PATH           |
+-----+-----+
|   19  | ./test/t2.ibd  |
|   20  | ./test/t3.ibd  |
...
|   68  | ./test/animals.ibd |
|   69  | ./test/animal_count.ibd |
|   70  | ./test/t.ibd    |
+-----+-----+
```

## 1.1.2.9.1.1.1.23 Information Schema INNODB\_SYS\_FIELDS Table

The [Information Schema](#)

**INNODB\_SYS\_FIELDS**  
table contains information about fields that are part of an InnoDB index.

The

[PROCESS privilege](#) is required to view the table.

It has the following columns:

Column	Description
INDEX_ID	Index identifier, matching the value from <a href="#">INNODB_SYS_INDEXES.INDEX_ID</a> .
NAME	Field name, matching the value from <a href="#">INNODB_SYS_COLUMNS.NAME</a> .
POS	Ordinal position of the field within the index, starting from 0 . This is adjusted as columns are removed.

## Example

```

SELECT * FROM information_schema.INNODB_SYS_FIELDS LIMIT 3\G
***** 1. row *****
INDEX_ID: 11
  NAME: ID
  POS: 0
***** 2. row *****
INDEX_ID: 12
  NAME: FOR_NAME
  POS: 0
***** 3. row *****
INDEX_ID: 13
  NAME: REF_NAME
  POS: 0
3 rows in set (0.00 sec)

```

## 1.1.2.9.1.1.1.24 Information Schema INNODB\_SYS\_FOREIGN Table

The [Information Schema](#)

`INNODB_SYS_FOREIGN`  
table contains information about InnoDB [foreign keys](#).

The

`PROCESS`  
`privilege` is required to view the table.

It has the following columns:

Column	Description
ID	Database name and foreign key name.
FOR_NAME	Database and table name of the foreign key child.
REF_NAME	Database and table name of the foreign key parent.
N_COLS	Number of foreign key index columns.
TYPE	Bit flag providing information about the foreign key.

The

`TYPE`  
column provides a bit flag with information about the foreign key. This information is  
OR  
'ed together to read:

Bit Flag	Description
1	ON DELETE CASCADE
2	ON UPDATE SET NULL
4	ON UPDATE CASCADE
8	ON UPDATE SET NULL

16	ON DELETE NO ACTION
32	ON UPDATE NO ACTION

## Example

```
SELECT * FROM INNODB_SYS_FOREIGN\G
***** 1. row *****
ID: mysql/innodb_index_stats_ibfk_1
FOR_NAME: mysql/innodb_index_stats
REF_NAME: mysql/innodb_table_stats
N_COLS: 2
TYPE: 0
...
...
```

## 1.1.2.9.1.1.1.25 Information Schema INNODB\_SYS\_FOREIGN\_COLS Table

The [Information Schema](#)

`INNODB_SYS_FOREIGN_COLS`  
table contains information about InnoDB [foreign key](#) columns.

The

`PROCESS`  
[privilege](#) is required to view the table.

It has the following columns:

Column	Description
ID	Foreign key index associated with this column, matching the <code>INNODB_SYS_FOREIGN.ID</code> field.
FOR_COL_NAME	Child column name.
REF_COL_NAME	Parent column name.
POS	Ordinal position of the column in the table, starting from 0.

## 1.1.2.9.1.1.1.26 Information Schema INNODB\_SYS\_INDEXES Table

The [Information Schema](#)

`INNODB_SYS_INDEXES`  
table contains information about InnoDB indexes.

The

`PROCESS`  
[privilege](#) is required to view the table.

It has the following columns:

Field	Type	Null	Key	Default	Description
INDEX_ID	bigint(21) unsigned	NO		0	A unique index identifier.

NAME	varchar(64)	NO			<p>Index name, lowercase for all user-created indexes, or uppercase for implicitly-created indexes;</p> <pre> PRIMARY (primary key), GEN_CLUST_INDEX (index representing primary key where there isn't one), ID_IND , FOR_IND (validating foreign key constraint) , REF_IND .</pre>
TABLE_ID	bigint(21) unsigned	NO		0	Table identifier, matching the value from <a href="#">INNODB_SYS_TABLES.TABLE_ID</a> .
TYPE	int(11)	NO		0	<p>Numeric type identifier; one of</p> <ul style="list-style-type: none"> <li>0 (secondary index),</li> <li>1 (clustered index),</li> <li>2 (unique index),</li> <li>3 (primary index),</li> <li>32 ( <a href="#">full-text index</a> ).</li> </ul>
N_FIELDS	int(11)	NO		0	Number of columns in the index. GEN_CLUST_INDEX's have a value of 0 as the index is not based on an actual column in the table.
PAGE_NO	int(11)	NO		0	<p>Index B-tree's root page number.</p> <ul style="list-style-type: none"> <li>-1 (unused) for full-text indexes, as they are laid out over several auxiliary tables.</li> </ul>
SPACE	int(11)	NO		0	<p>Tablespace identifier where the index resides.</p> <ul style="list-style-type: none"> <li>0 represents the InnoDB system tablespace, while any other value represents a table created in file-per-table mode (see the <a href="#">innodb_file_per_table</a> system variable).</li> </ul> <p>Remains unchanged after a <a href="#">TRUNCATE TABLE</a> statement, and not necessarily unique.</p>
MERGE_THRESHOLD	int(11)	NO		0	

## Example

```

SELECT * FROM information_schema.INNODB_SYS_INDEXES LIMIT 3\G
***** 1. row *****
INDEX_ID: 11
  NAME: ID_IND
TABLE_ID: 11
  TYPE: 3
N_FIELDS: 1
PAGE_NO: 302
  SPACE: 0
MERGE_THRESHOLD: 50
***** 2. row *****
INDEX_ID: 12
  NAME: FOR_IND
TABLE_ID: 11
  TYPE: 0
N_FIELDS: 1
PAGE_NO: 303
  SPACE: 0
MERGE_THRESHOLD: 50
***** 3. row *****
INDEX_ID: 13
  NAME: REF_IND
TABLE_ID: 11
  TYPE: 3
N_FIELDS: 1
PAGE_NO: 304
  SPACE: 0
MERGE_THRESHOLD: 50
3 rows in set (0.00 sec)

```

## 1.1.2.9.1.1.1.27 Information Schema INNODB\_SYS\_SEMAPHORE\_WAITS Table

The [Information Schema](#) INNODB\_SYS\_SEMAPHORE\_WAITS table is meant to contain information about current semaphore waits. At present it is not correctly populated. See [MDEV-21330](#).

The [PROCESS privilege](#) is required to view the table.

It contains the following columns:

Column	Description
THREAD_ID	Thread id waiting for semaphore
OBJECT_NAME	Semaphore name
FILE	File name where semaphore was requested
LINE	Line number on above file
WAIT_TIME	Wait time
WAIT_OBJECT	
WAIT_TYPE	Object type (mutex, rw-lock)
HOLDER_THREAD_ID	Holder thread id
HOLDER_FILE	File name where semaphore was acquired
HOLDER_LINE	Line number for above
CREATED_FILE	Creation file name
CREATED_LINE	Line number for above
WRITER_THREAD	Last write request thread id
RESERVATION_MODE	Reservation mode (shared, exclusive)
READERS	Number of readers if only shared mode
WAITERS_FLAG	Flags
LOCK_WORD	Lock word (for developers)
LAST_READER_FILE	Removed
LAST_READER_LINE	Removed

LAST_WRITER_FILE	Last writer file name
LAST_WRITER_LINE	Above line number
OS_WAIT_COUNT	Wait count

## 1.1.2.9.1.1.1.28 Information Schema INNODB\_SYS\_TABLES Table

The [Information Schema](#)

INNODB\_SYS\_TABLES

table contains information about InnoDB tables.

The

PROCESS

[privilege](#) is required to view the table.

It has the following columns:

Field	Type	Null	Key	Default	Description
TABLE_ID	bigint(21) unsigned	NO		0	Unique InnoDB table identifier.
NAME	varchar(655)	NO			Database and table name, or the uppercase InnoDB system table name.
FLAG	int(11)	NO		0	See <a href="#">Flag</a> below
N_COLS	int(11) unsigned (>= <a href="#">MariaDB 10.5</a> ) int(11) (<= <a href="#">MariaDB 10.4</a> )	NO		0	Number of columns in the table.
SPACE	int(11) unsigned (>= <a href="#">MariaDB 10.5</a> ) int(11) (<= <a href="#">MariaDB 10.4</a> )	NO		0	Tablespace identifier where the index resides.  0 represents the InnoDB system tablespace, while any other value represents a table created in file-per-table mode (see the <a href="#">innodb_file_per_table</a> system variable). Remains unchanged after a <a href="#">TRUNCATE TABLE</a> statement.
FILE_FORMAT	varchar(10)	YES		NULL	<a href="#">InnoDB file format</a> (Antelope or Barracuda). Removed in <a href="#">MariaDB 10.3</a> .
ROW_FORMAT	enum('Redundant', 'Compact', 'Compressed', 'Dynamic') (>= <a href="#">MariaDB 10.5</a> ) varchar(12) (<= <a href="#">MariaDB 10.4</a> )	YES		NULL	<a href="#">InnoDB storage format</a> (Compact, Redundant, Dynamic, or Compressed).
ZIP_PAGE_SIZE	int(11) unsigned	NO		0	For Compressed tables, the zipped page size.
SPACE_TYPE	enum('Single','System') (>= <a href="#">MariaDB 10.5</a> ) varchar(10) (<= <a href="#">MariaDB 10.4</a> )	YES		NULL	

### Flag

The flag field returns the dict\_table\_t::flags that correspond to the data dictionary record.

Bit	Description
0	Set if ROW_FORMAT is not REDUNDANT.
1 to 4	0 , except for ROW_FORMAT=COMPRESSED, where they will determine the KEY_BLOCK_SIZE (the compressed page size).
5	Set for ROW_FORMAT=DYNAMIC or ROW_FORMAT=COMPRESSED.

	6	Set if the DATA DIRECTORY attribute was present when the table was originally created.
	7	Set if the page_compressed attribute is present.
	8 to 11	Determine the page_compression_level.
	12 13	Normally 00 , but 11 for "no-rollback tables" ( <a href="#">MariaDB 10.3 CREATE SEQUENCE</a> ). In <a href="#">MariaDB 10.1</a> , these bits could be 01 or 10 for ATOMIC_WRITES=ON or ATOMIC_WRITES=OFF.

Note that the table flags returned here are not the same as tablespace flags (FSP\_SPACE\_FLAGS).

## Example

```
SELECT * FROM information_schema.INNODB_SYS_TABLES LIMIT 2\G
*****
1. row ****
TABLE_ID: 14
  NAME: SYS_DATAFILES
  FLAG: 0
  N_COLS: 5
  SPACE: 0
FILE_FORMAT: Antelope
ROW_FORMAT: Redundant
ZIP_PAGE_SIZE: 0
SPACE_TYPE: System
*****
2. row ****
TABLE_ID: 11
  NAME: SYS_FOREIGN
  FLAG: 0
  N_COLS: 7
  SPACE: 0
FILE_FORMAT: Antelope
ROW_FORMAT: Redundant
ZIP_PAGE_SIZE: 0
SPACE_TYPE: System
2 rows in set (0.00 sec)
```

## See Also

- [InnoDB Data Dictionary Troubleshooting](#)

## 1.1.2.9.1.1.1.29 Information Schema INNODB\_SYS\_TABLESPACES Table

The [Information Schema](#)

INNODB\_SYS\_TABLESPACES

table contains information about InnoDB tablespaces. Until [MariaDB 10.5](#) it was based on the internal

SYS\_TABLESPACES

table. This internal table was removed in [MariaDB 10.6.0](#), so this Information Schema table has been repurposed to directly reflect the filesystem (fil\_system.space\_list).

The

PROCESS

privilege is required to view the table.

It has the following columns:

Column	Description
SPACE	Unique InnoDB tablespace identifier.
NAME	Database and table name separated by a backslash, or the uppercase InnoDB system table name.
FLAG	<pre> 1 if a DATA DIRECTORY option has been specified in  CREATE TABLE  , otherwise 0 . </pre>
FILE_FORMAT	InnoDB file format .
ROW_FORMAT	InnoDB storage format used for this tablespace. If the Antelope file format is used, this value is always Compact or Redundant .
PAGE_SIZE	Page size in bytes for this tablespace. Until MariaDB 10.5.0 , this was the value of the innodb_page_size variable. From MariaDB 10.6.0 , contains the physical page size of a page (previously ZIP_PAGE_SIZE ).
ZIP_PAGE_SIZE	Zip page size for this tablespace. Removed in MariaDB 10.6.0 .
SPACE_TYPE	Tablespace type. Can be General for general tablespaces or Single for file-per-table tablespaces. Introduced MariaDB 10.2.1 . Removed MariaDB 10.5.0 .
FS_BLOCK_SIZE	File system block size. Introduced MariaDB 10.2.1 .
FILE_SIZE	Maximum size of the file, uncompressed. Introduced MariaDB 10.2.1 .
ALLOCATED_SIZE	Actual size of the file as per space allocated on disk. Introduced MariaDB 10.2.1 .
FILENAME	Tablespace datafile path, previously part of the INNODB_SYS_DATAFILES table . Added in MariaDB 10.6.0 .

## Examples

MariaDB 10.4 :

```
DESC information_schema.innodb_sys_tablespaces;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| SPACE | int(11) unsigned | NO | 0 | NULL | |
| NAME | varchar(655) | NO | 0 | NULL | |
| FLAG | int(11) unsigned | NO | 0 | NULL | |
| FILE_FORMAT | varchar(10) | YES | NULL | NULL | |
| ROW_FORMAT | varchar(22) | YES | NULL | NULL | |
| PAGE_SIZE | int(11) unsigned | NO | 0 | NULL | |
| ZIP_PAGE_SIZE | int(11) unsigned | NO | 0 | NULL | |
| SPACE_TYPE | varchar(10) | YES | NULL | NULL | |
| FS_BLOCK_SIZE | int(11) unsigned | NO | 0 | NULL | |
| FILE_SIZE | bigint(21) unsigned | NO | 0 | NULL | |
| ALLOCATED_SIZE | bigint(21) unsigned | NO | 0 | NULL | |
+-----+-----+-----+-----+-----+
```

From MariaDB 10.4 :

```
SELECT * FROM information_schema.INNODB_SYS_TABLESPACES LIMIT 2\G
*****
1. row *****
    SPACE: 2
        NAME: mysql/innodb_table_stats
        FLAG: 33
    FILE_FORMAT: Barracuda
    ROW_FORMAT: Dynamic
    PAGE_SIZE: 16384
    ZIP_PAGE_SIZE: 0
    SPACE_TYPE: Single
    FS_BLOCK_SIZE: 4096
        FILE_SIZE: 98304
    ALLOCATED_SIZE: 98304
*****
2. row *****
    SPACE: 3
        NAME: mysql/innodb_index_stats
        FLAG: 33
    FILE_FORMAT: Barracuda
    ROW_FORMAT: Dynamic
    PAGE_SIZE: 16384
    ZIP_PAGE_SIZE: 0
    SPACE_TYPE: Single
    FS_BLOCK_SIZE: 4096
        FILE_SIZE: 98304
    ALLOCATED_SIZE: 98304
```

## 1.1.2.9.1.1.30 Information Schema INNODB\_SYS\_TABLESTATS Table

The [Information Schema](#)

### INNODB\_SYS\_TABLESTATS

table contains InnoDB status information. It can be used for developing new performance-related extensions, or high-level performance monitoring.

The

### PROCESS

[privilege](#) is required to view the table.

Note that the MySQL InnoDB and Percona XtraDB versions of the tables differ (see [XtraDB and InnoDB](#) ).

It contains the following columns:

Column	Description
TABLE_ID	Table ID, matching the <a href="#">INNODB_SYS_TABLES.TABLE_ID</a> value.
SCHEMA	Database name (XtraDB only).
NAME	Table name, matching the <a href="#">INNODB_SYS_TABLES.NAME</a> value.

STATS_INITIALIZED	Initialized if statistics have already been collected, otherwise Uninitialized
NUM_ROWS	Estimated number of rows currently in the table. Updated after each statement modifying the data, but uncommitted transactions mean it may not be accurate.
CLUST_INDEX_SIZE	Number of pages on disk storing the clustered index, holding InnoDB table data in primary key order, or NULL if not statistics yet collected.
OTHER_INDEX_SIZE	Number of pages on disk storing secondary indexes for the table, or NULL if not statistics yet collected.
MODIFIED_COUNTER	Number of rows modified by statements modifying data.
AUTOINC	<a href="#">Auto_increment</a> value.
REF_COUNT	Countdown to zero, when table metadata can be removed from the table cache. (InnoDB only)
MYSQL_HANDLES_OPENED	(XtraDB only).

## 1.1.2.9.1.1.1.31 Information Schema INNODB\_SYS\_VIRTUAL Table

MariaDB starting with [10.2](#)

The

`INNODB_SYS_VIRTUAL`  
table was added in [MariaDB 10.2](#).

The [Information Schema](#)

`INNODB_SYS_VIRTUAL`  
table contains information about base columns of [virtual columns](#). The  
`PROCESS`  
`privilege` is required to view the table.

It contains the following columns:

Field	Type	Null	Key	Default	Description
TABLE_ID	bigint(21) unsigned	NO		0	
POS	int(11) unsigned	NO		0	
BASE_POS	int(11) unsigned	NO		0	

## 1.1.2.9.1.1.1.32 Information Schema INNODB\_TABLESPACES\_ENCRYPTION Table

The [Information Schema](#)

`INNODB_TABLESPACES_ENCRYPTION`  
table contains metadata about [encrypted InnoDB tablespaces](#). When you [enable encryption for an InnoDB tablespace](#), an entry for the  
tablespace is added to this table. If you later [disable encryption for the InnoDB tablespace](#), then the row still remains in this table, but the  
`ENCRYPTION_SCHEME`  
and

CURRENT\_KEY\_VERSION  
columns will be set to  
0

Viewing this table requires the [PROCESS](#) privilege, although a bug in versions before MariaDB 10.1.46 , 10.2.33 , 10.3.24 , 10.4.14 and 10.5.5 mean the [SUPER](#) privilege was required ( [MDEV-23003](#) ).

It has the following columns:

Column	Description	Added
SPACE	InnoDB tablespace ID.	
NAME	Path to the InnoDB tablespace file, without the extension.	
ENCRYPTION_SCHEME	Key derivation algorithm. Only 1 is currently used to represent an algorithm. If this value is 0 , then the tablespace is unencrypted.	
KEYSERVER_REQUESTS	Number of times InnoDB has had to request a key from the <a href="#">encryption key management plugin</a> . The three most recent keys are cached internally.	
MIN_KEY_VERSION	Minimum key version used to encrypt a page in the tablespace. Different pages may be encrypted with different key versions.	
CURRENT_KEY_VERSION	Key version that will be used to encrypt pages. If this value is 0 , then the tablespace is unencrypted.	
KEY_ROTATION_PAGE_NUMBER	Page that a <a href="#">background encryption thread</a> is currently rotating. If key rotation is not enabled, then the value will be NULL	
KEY_ROTATION_MAX_PAGE_NUMBER	When a <a href="#">background encryption thread</a> starts rotating a tablespace, the field contains its current size. If key rotation is not enabled, then the value will be NULL	
CURRENT_KEY_ID	Key ID for the encryption key currently in use.	MariaDB 10.1.13
ROTATING_OR_FLUSHING	Current key rotation status. If this value is 1 , then the <a href="#">background encryption threads</a> are working on the tablespace. See <a href="#">MDEV-11738</a>	MariaDB 10.2.5 , MariaDB 10.1.23

When the [InnoDB system tablespace](#) is encrypted, it is represented in this table with the special name:

innodb\_system

## Example

```

SELECT * FROM information_schema.INNODB_TABLESPACES_ENCRYPTION
WHERE NAME LIKE 'db_encrypt%';
+-----+-----+-----+-----+-----+
| SPACE | NAME | ENCRYPTION_SCHEME | KEYSERVER_REQUESTS | MIN_KEY_VERSION | CURRENT_KEY |
+-----+-----+-----+-----+-----+
| 18 | db_encrypt/t_encrypted_existing_key | 1 | 1 | 1 | |
| 19 | db_encrypt/t_not_encrypted_existing_key | 1 | 0 | 1 |
| 20 | db_encrypt/t_not_encrypted_non_existing_key | 1 | 0 | 4294967295 | 429 |
| 21 | db_encrypt/t_default_encryption_existing_key | 1 | 1 | 1 |
| 22 | db_encrypt/t_encrypted_default_key | 1 | 1 | 1 |
| 23 | db_encrypt/t_not_encrypted_default_key | 1 | 0 | 1 |
| 24 | db_encrypt/t_defaults | 1 | 1 | 1 |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

## See Also

- [Encrypting Data for InnoDB / XtraDB](#)
- [Data at Rest Encryption](#)
- [Why Encrypt MariaDB Data?](#)
- [Encryption Key Management](#)

## 1.1.2.9.1.1.1.33 Information Schema INNODB\_TABLESPACES\_SCRUBBING Table

MariaDB 10.1.3 - 10.5.1

InnoDB and XtraDB data scrubbing was introduced in [MariaDB 10.1.3](#). The table was removed in [MariaDB 10.5.2](#) - see [MDEV-15528](#).

The [Information Schema](#)

INNODB\_TABLESPACES\_SCRUBBING  
table contains [data scrubbing](#) information.

The

PROCESS  
[privilege](#) is required to view the table.

It has the following columns:

Column	Description
SPACE	InnoDB table space id number.
NAME	Path to the table space file, without the extension.
COMPRESSED	The compressed page size, or zero if uncompressed.
LAST_SCRUB_COMPLETED	Date and time when the last scrub was completed, or NULL if never been performed.
CURRENT_SCRUB_STARTED	Date and time when the current scrub started, or NULL if never been performed.
CURRENT_SCRUB_ACTIVE_THREADS	Number of threads currently scrubbing the tablespace.
CURRENT_SCRUB_PAGE_NUMBER	Page that the scrubbing thread is currently scrubbing, or NULL if not enabled.
CURRENT_SCRUB_MAX_PAGE_NUMBER	When a scrubbing starts rotating a table space, the field contains its current size. NULL if not enabled.

## Example

```
SELECT * FROM information_schema.INNODB_TABLESPACES_SCRUBBING LIMIT 1\G
***** 1. row *****
    SPACE: 1
        NAME: mysql/innodb_table_stats
    COMPRESSED: 0
    LAST_SCRUB_COMPLETED: NULL
    CURRENT_SCRUB_STARTED: NULL
    CURRENT_SCRUB_PAGE_NUMBER: NULL
CURRENT_SCRUB_MAX_PAGE_NUMBER: 0
    ROTATING_OR_FLUSHING: 0
1 rows in set (0.00 sec)
```

## 1.1.2.9.1.1.1.34 Information Schema INNODB\_TRX Table

The [Information Schema](#)

INNODB\_TRX

table stores information about all currently executing InnoDB transactions.

It has the following columns:

Column	Description
TRX_ID	Unique transaction ID number.
TRX_STATE	Transaction execution state; one of RUNNING , LOCK_WAIT , ROLLING BACK or COMMITTING .
TRX_STARTED	Time that the transaction started.
TRX_REQUESTED_LOCK_ID	If TRX_STATE is LOCK_WAIT , the  INNODB_LOCKS.LOCK_ID  value of the lock being waited on. NULL if any other state.
TRX_WAIT_STARTED	If TRX_STATE is LOCK_WAIT , the time the transaction started waiting for the lock, otherwise NULL .
TRX_WEIGHT	Transaction weight, based on the number of locked rows and the number of altered rows. To resolve deadlocks, lower weighted transactions are rolled back first. Transactions that have affected non-transactional tables are always treated as having a heavier weight.
TRX_MYSQL_THREAD_ID	Thread ID from the  PROCESSTLIST  table (note that the locking and transaction information schema tables use a different snapshot from the processlist, so records may appear in one but not the other).

TRX_QUERY	SQL that the transaction is currently running.
TRX_OPERATION_STATE	Transaction's current state, or NULL
TRX_TABLES_IN_USE	Number of InnoDB tables currently being used for processing the current SQL statement.
TRX_TABLES_LOCKED	Number of InnoDB tables that have row locks held by the current SQL statement.
TRX_LOCK_STRUCTS	Number of locks reserved by the transaction.
TRX_LOCK_MEMORY_BYTES	Total size in bytes of the memory used to hold the lock structures for the current transaction in memory.
TRX_ROWS_LOCKED	Number of rows the current transaction has locked. An approximation, and may include rows not visible to the current transaction that are delete-marked but physically present.
TRX_ROWS_MODIFIED	Number of rows added or changed in the current transaction.
TRX_CONCURRENCY_TICKETS	Indicates how much work the current transaction can do before being swapped out, see the <a href="#">innodb_concurrency_tickets</a> system variable.
TRX_ISOLATION_LEVEL	<a href="#">Isolation level</a> of the current transaction.
TRX_UNIQUE_CHECKS	Whether unique checks are on or off for the current transaction. Bulk data are a case where unique checks would be off.
TRX_FOREIGN_KEY_CHECKS	Whether foreign key checks are on or off for the current transaction. Bulk data are a case where foreign keys checks would be off.
TRX_LAST_FOREIGN_KEY_ERROR	Error message for the most recent foreign key error, or NULL if none.
TRX_ADAPTIVE_HASH_LATCHED	Whether the adaptive hash index is locked by the current transaction or not. One transaction at a time can change the adaptive hash index.
TRX_ADAPTIVE_HASH_TIMEOUT	Whether the adaptive hash index search latch should be relinquished immediately or reserved across all MariaDB calls. 0 if there is no contention on the adaptive hash index, in which case the latch is reserved until completion, otherwise counts down to zero and the latch is released after each row lookup.

TRX_IS_READ_ONLY	<pre> 1 if a read-only transaction, otherwise 0 . </pre>
TRX_AUTOCOMMIT_NON_LOCKING	<pre> 1 if the transaction only contains this one statement, that is, a  SELECT  statement not using FOR UPDATE or LOCK IN SHARED MODE , and with autocommit on. If this and TRX_IS_READ_ONLY are both 1, the transaction can be optimized by the storage engine to reduce some overheads </pre>

The table is often used in conjunction with the [INNODB\\_LOCKS](#) and [INNODB\\_LOCK\\_WAITS](#) tables to diagnose problematic locks and transactions.

[XA transactions](#) are not stored in this table. To see them,

`XA RECOVER`  
can be used.

## Example

```
-- session 1
START TRANSACTION;
UPDATE t SET id = 15 WHERE id = 10;

-- session 2
DELETE FROM t WHERE id = 10;

-- session 1
USE information_schema;
SELECT 1.* , t.* 
  FROM information_schema.INNODB_LOCKS l 
  JOIN information_schema.INNODB_TRX t 
    ON l.lock trx_id = t.trx_id 
   WHERE trx_state = 'LOCK WAIT' \G
*****
1. row ****
lock_id: 840:40:3:2
lock_trx_id: 840
lock_mode: X
lock_type: RECORD
lock_table: `test`.`t`
lock_index: PRIMARY
lock_space: 40
lock_page: 3
lock_rec: 2
lock_data: 10
trx_id: 840
trx_state: LOCK WAIT
trx_started: 2019-12-23 18:43:46
trx_requested_lock_id: 840:40:3:2
trx_wait_started: 2019-12-23 18:43:46
trx_weight: 2
trx_mysql_thread_id: 46
trx_query: DELETE FROM t WHERE id = 10
trx_operation_state: starting index read
trx_tables_in_use: 1
trx_tables_locked: 1
trx_lock_structs: 2
trx_lock_memory_bytes: 1136
trx_rows_locked: 1
trx_rows_modified: 0
trx_concurrency_tickets: 0
trx_isolation_level: REPEATABLE READ
trx_unique_checks: 1
trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
trx_is_read_only: 0
trx_autocommit_non_locking: 0
```

## 1.1.2.9.1.1.35 Information Schema TEMP\_TABLES\_INFO Table

MariaDB 10.2.2 - 10.2.3

The

`TEMP_TABLES_INFO`

table was introduced in MariaDB 10.2.2 and was removed in MariaDB 10.2.4 . See [MDEV-12459](#) progress on an alternative.

The [Information Schema](#)

`TEMP_TABLES_INFO`

table contains information about active InnoDB temporary tables. All user and system-created temporary tables are reported when querying this table, with the exception of optimized internal temporary tables. The data is stored in memory.

Previously, InnoDB temp table metadata was rather stored in InnoDB system tables.

It has the following columns:

Column	Description
TABLE_ID	Table ID.

NAME	Table name.
N_COLS	Number of columns in the temporary table, including three hidden columns that InnoDB creates ( DB_ROW_ID , DB_TRX_ID , and DB_ROLL_PTR ).
SPACE	Numerical identifier for the tablespace identifier holding the temporary table. Compressed temporary tables are stored by default in separate per-table tablespaces in the temporary file directory. For non-compressed tables, the shared temporary table is named ibtmp1 , found in the data directory. Always a non-zero value, and regenerated on server restart.
PER_TABLE_TABLESPACE	If TRUE , the temporary table resides in a separate per-table tablespace. If FALSE , it resides in the shared temporary tablespace.
IS_COMPRESSED	TRUE if the table is compressed.

The `PROCESS` privilege is required to view the table.

## Examples

```
CREATE TEMPORARY TABLE t (i INT) ENGINE=INNODB;

SELECT * FROM INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO;
+-----+-----+-----+-----+
| TABLE_ID | NAME      | N_COLS | SPACE | PER_TABLE_TABLESPACE | IS_COMPRESSED |
+-----+-----+-----+-----+
|     39 | #sql1c93_3_1 |      4 |    64 | FALSE             | FALSE          |
+-----+-----+-----+-----+
```

Adding a compressed table:

```
SET GLOBAL innodb_file_format="Barracuda";

CREATE TEMPORARY TABLE t2 (i INT) ROW_FORMAT=COMPRESSED ENGINE=INNODB;

SELECT * FROM INFORMATION_SCHEMA.INNODB_TEMP_TABLE_INFO;
+-----+-----+-----+-----+
| TABLE_ID | NAME      | N_COLS | SPACE | PER_TABLE_TABLESPACE | IS_COMPRESSED |
+-----+-----+-----+-----+
|     40 | #sql1c93_3_3 |      4 |    65 | TRUE              | TRUE           |
|     39 | #sql1c93_3_1 |      4 |    64 | FALSE             | FALSE          |
+-----+-----+-----+-----+
```

### 1.1.2.9.1.1.2 Information Schema MyRocks Tables

#### 1.1.2.9.1.1.2.1 Information Schema ROCKSDB\_CFSTATS Table

The [Information Schema](#)

`ROCKSDB_CFSTATS` table is included as part of the [MyRocks](#) storage engine.

The

`PROCESS` privilege is required to view the table.

It contains the following columns:

Column	Description
CF_NAME	
STAT_TYPE	
VALUE	

## 1.1.2.9.1.1.2.2 Information Schema ROCKSDB\_CF\_OPTIONS Table

The [Information Schema](#)

ROCKSDB\_CF\_OPTIONS table is included as part of the [MyRocks](#) storage engine, and contains information about MyRocks [column families](#).

The

[PROCESS](#)  
[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
CF_NAME	<a href="#">Column family</a> name.
OPTION_TYPE	
VALUE	

## 1.1.2.9.1.1.2.3 Information Schema ROCKSDB\_COMPACTION\_STATS Table

The [Information Schema](#)

ROCKSDB\_COMPACTION\_STATS table is included as part of the [MyRocks](#) storage engine.

The

[PROCESS](#)  
[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
CF_NAME	
LEVEL	
TYPE	
VALUE	

## 1.1.2.9.1.1.2.4 Information Schema ROCKSDB\_DBSTATS Table

The [Information Schema](#)

ROCKSDB\_DBSTATS  
table is included as part of the [MyRocks](#) storage engine.

The

PROCESS  
privilege is required to view the table.

It contains the following columns:

Column	Description
STAT_TYPE	
VALUE	

## 1.1.2.9.1.1.2.5 Information Schema ROCKSDB\_DDL Table

The [Information Schema](#)

ROCKSDB\_DDL  
table is included as part of the [MyRocks](#) storage engine.

The

PROCESS  
privilege is required to view the table.

It contains the following columns:

Column	Description
TABLE_SCHEMA	
TABLE_NAME	
PARTITION_NAME	
INDEX_NAME	
COLUMN_FAMILY	
INDEX_NUMBER	
INDEX_TYPE	
KV_FORMAT_VERSION	
TTL_DURATION	
INDEX_FLAGS	

CF	
AUTO_INCREMENT	

## 1.1.2.9.1.1.2.6 Information Schema ROCKSDB\_DEADLOCK Table

The [Information Schema](#)

ROCKSDB\_DEADLOCK  
table is included as part of the [MyRocks](#) storage engine.

The

PROCESS  
[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
DEADLOCK_ID	
TIMESTAMP	
TRANSACTION_ID	
CF_NAME	
WAITING_KEY	
LOCK_TYPE	
INDEX_NAME	
TABLE_NAME	
ROLLED_BACK	

## 1.1.2.9.1.1.2.7 Information Schema ROCKSDB\_GLOBAL\_INFO Table

The [Information Schema](#)

ROCKSDB\_GLOBAL\_INFO  
table is included as part of the [MyRocks](#) storage engine.

The

PROCESS  
[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
--------	-------------

TYPE	
NAME	
VALUE	

## 1.1.2.9.1.1.2.8 Information Schema ROCKSDB\_INDEX\_FILE\_MAP Table

The [Information Schema](#)

ROCKSDB\_INDEX\_FILE\_MAP  
table is included as part of the [MyRocks](#) storage engine.

The

PROCESS  
[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
COLUMN_FAMILY	
INDEX_NUMBER	
SST_NAME	
NUM_ROWS	
DATA_SIZE	
ENTRY_DELETES	
ENTRY_SINGLEDELETES	
ENTRY_MERGES	
ENTRY_OTHERS	
DISTINCT_KEYS_PREFIX	

## 1.1.2.9.1.1.2.9 Information Schema ROCKSDB\_LOCKS Table

The [Information Schema](#)

ROCKSDB\_LOCKS  
table is included as part of the [MyRocks](#) storage engine.

The

PROCESS

[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
COLUMN_FAMILY_ID	
TRANSACTION_ID	
KEY	
MODE	

## 1.1.2.9.1.1.2.10 Information Schema ROCKSDB\_PERF\_CONTEXT Table

The [Information Schema](#)

ROCKSDB\_PERF\_CONTEXT

table is included as part of the [MyRocks](#) storage engine and includes per-table/partition counters .

The

PROCESS

[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
TABLE_SCHEMA	
TABLE_NAME	
PARTITION_NAME	
STAT_TYPE	
VALUE	

Note: for multi-table queries, all counter increments are "billed" to the first table in the query: <https://github.com/facebook/mysql-5.6/issues/1018>

## 1.1.2.9.1.1.2.11 Information Schema ROCKSDB\_PERF\_CONTEXT\_GLOBAL Table

The [Information Schema](#)

ROCKSDB\_PERF\_CONTEXT\_GLOBAL

table is included as part of the [MyRocks](#) storage engine and includes global counter information.

The

PROCESS

[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
--------	-------------

STAT_TYPE	
VALUE	

## 1.1.2.9.1.1.2.12 Information Schema ROCKSDB\_SST\_PROPS Table

The [Information Schema](#)

ROCKSDB\_SST\_PROPS  
table is included as part of the [MyRocks](#) storage engine.

The

PROCESS  
privilege is required to view the table.

It contains the following columns:

Column	Description
SST_NAME	
COLUMN_FAMILY	
DATA_BLOCKS	
ENTRIES	
RAW_KEY_SIZE	
RAW_VALUE_SIZE	
DATA_BLOCK_SIZE	
INDEX_BLOCK_SIZE	
INDEX_PARTITIONS	
TOP_LEVEL_INDEX_SIZE	
FILTER_BLOCK_SIZE	
COMPRESSION_ALGO	
CREATION_TIME	

## 1.1.2.9.1.1.2.13 Information Schema ROCKSDB\_TRX

# Table

The [Information Schema](#)

ROCKSDB\_TRX  
table is included as part of the [MyRocks](#) storage engine.

The

PROCESS  
[privilege](#) is required to view the table.

It contains the following columns:

Column	Description
TRANSACTION_ID	
STATE	
NAME	
WRITE_COUNT	
LOCK_COUNT	
TIMEOUT_SEC	
WAITING_KEY	
WAITING_COLUMN_FAMILY_ID	
IS_REPLICATION	
SKIP_TRX_API	
READ_ONLY	
HAS_DEADLOCK_DETECTION	
NUM_ONGOING_BULKLOAD	
THREAD_ID	
QUERY	

## 1.1.2.9.1.1.3 ColumnStore Information Schema Tables

1. COLUMNSTORE\_TABLES
2. COLUMNSTORE\_COLUMNS
3. COLUMNSTORE\_EXTENTS
4. COLUMNSTORE\_FILES
5. Stored Procedures
  1. total\_usage()
  2. table\_usage()
  3. compression\_ratio()

MariaDB ColumnStore has four Information Schema tables that expose information about the table and column storage. These tables were added in version 1.0.5 of ColumnStore and were heavily modified for 1.0.6.

## COLUMNSTORE\_TABLES

The first table is the INFORMATION\_SCHEMA.COLUMNSTORE\_TABLES. This contains information about the tables inside ColumnStore. The table layout is as follows:

Column	Description
TABLE_SCHEMA	The database schema for the table
TABLE_NAME	The table name
OBJECT_ID	The ColumnStore object ID for the table
CREATION_DATE	The date the table was created
COLUMN_COUNT	The number of columns in the table
AUTOINCREMENT	The start autoincrement value for the table set during CREATE TABLE

**Note:** Tables created with ColumnStore 1.0.4 or lower will have the year field of the creation data set incorrectly by 1900 years.

## COLUMNSTORE\_COLUMNS

The INFORMATION\_SCHEMA.COLUMNSTORE\_COLUMNS table contains information about every single column inside ColumnStore. The table layout is as follows:

Column	Description
TABLE_SCHEMA	The database schema for the table
TABLE_NAME	The table name for the column
COLUMN_NAME	The column name
OBJECT_ID	The object ID for the column
DICTIONARY_OBJECT_ID	The dictionary object ID for the column (NULL if there is no dictionary object)
LIST_OBJECT_ID	Placeholder for future information
TREE_OBJECT_ID	Placeholder for future information
DATA_TYPE	The data type for the column
COLUMN_LENGTH	The data length for the column
COLUMN_POSITION	The position of the column in the table, starting at 0
COLUMN_DEFAULT	The default value for the column
IS_NULLABLE	Whether or not the column can be set to NULL
NUMERIC_PRECISION	The numeric precision for the column
NUMERIC_SCALE	The numeric scale for the column
IS_AUTOINCREMENT	Set to 1 if the column is an autoincrement column
COMPRESSION_TYPE	The type of compression (either "None" or "Snappy")

## COLUMNSTORE\_EXTENTS

This table displays the extent map in a user consumable form. An extent is a collection of details about a section of data related to a columnstore column. A majority of columns in ColumnStore will have multiple extents and the columns table above can be joined to this one to filter results by table or column. The table layout is as follows:

Column	Description
OBJECT_ID	The object ID for the extent
OBJECT_TYPE	Whether this is a "Column" or "Dictionary" extent
LOGICAL_BLOCK_START	ColumnStore's internal start LBID for this extent
LOGICAL_BLOCK_END	ColumnStore's internal end LBID for this extent
MIN_VALUE	This minimum value stored in this extent
MAX_VALUE	The maximum value stored in this extent
WIDTH	The data width for the extent
DBROOT	The DBRoot number for the extent
PARTITION_ID	The partition ID for the extent
SEGMENT_ID	The segment ID for the extent
BLOCK_OFFSET	The block offset for the data file, each data file can contain multiple extents for a column
MAX_BLOCKS	The maximum number of blocks for the extent
HIGH_WATER_MARK	The last block committed to the extent (starting at 0)
STATE	The state of the extent (see below)
STATUS	The availability status for the column which is either "Available", "Unavailable" or "Out of service"
DATA_SIZE	The uncompressed data size for the extent calculated as $(HWM + 1) * BLOCK\_SIZE$

#### Notes:

1. The state is "Valid" for a normal state, "Invalid" if a cpimport has completed but the table has not yet been accessed (min/max values will be invalid) or "Updating" if there is a DML statement writing to the column
2. In ColumnStore the block size is 8192 bytes
3. By default ColumnStore will write create an extent file of  $256 * 1024 * WIDTH$  bytes for the first partition, if this is too small then for uncompressed data it will create a file of the maximum size for the extent ( $MAX\_BLOCKS * BLOCK\_SIZE$ ). Snappy always compression adds a header block.
4. Object IDs of less than 3000 are for internal tables and will not appear in any of the information schema tables
5. Prior to 1.0.12 / 1.1.2 DATA\_SIZE was incorrectly calculated
6. HWM is set to zero for the lower segments when there are multiple segments in an extent file, these can be observed when BLOCK\_OFFSET > 0
7. When HWM is 0 the DATA\_SIZE will show 0 instead of 8192 to avoid confusion when there is multiple segments in an extent file

## COLUMNSTORE\_FILES

The columnstore\_files table provides information about each file associated with extensions. Each extension can reuse a file at different block offsets so this is not a 1:1 relationship to the columnstore\_extents table.

Column	Description
OBJECT_ID	The object ID for the extent
SEGMENT_ID	The segment ID for the extent
PARTITION_ID	The partition ID for the extent
FILENAME	The full path and filename for the extent file, multiple extents for the same column can point to this file with different BLOCK_OFFSETS
FILE_SIZE	The disk file size for the extent
COMPRESSED_DATA_SIZE	The amount of the compressed file used, NULL if this is an uncompressed file

## Stored Procedures

A few stored procedures were added in 1.0.6 to provide summaries based on the information schema tables. These can be accessed from the COLUMNSTORE\_INFO schema.

### total\_usage()

The total\_usage() procedure gives a total disk usage summary for all the columns in ColumnStore with the exception of the columns used for internal maintenance. It is executed using the following query:

```
> call columnstore_info.total_usage();
```

## table\_usage()

The table\_usage() procedure gives a the total data disk usage, dictionary disk usage and grand total disk usage per-table. It can be called in several ways, the first gives a total for each table:

```
> call columnstore_info.table_usage(NULL, NULL);
```

Or for a specific table, my\_table in my\_schema in this example:

```
> call columnstore_info.table_usage('my_schema', 'my_table');
```

You can also request all tables for a specified schema:

```
> call columnstore_info.table_usage('my_schema', NULL);
```

**Note:** The quotes around the table name are required, an error will occur without them.

## compression\_ratio()

The compression\_ratio() procedure calculates the average compression ratio across all the compressed extents in ColumnStore. It is called using:

```
> call columnstore_info.compression_ratio();
```

**Note:** The compression ratio is incorrectly calculated before versions 1.0.12 / 1.1.2

## 1.1.2.9.1.1.4 Information Schema ALL\_PLUGINS Table Description

The [Information Schema](#)

ALL\_PLUGINS

table contains information about [server plugins](#), whether installed or not.

It contains the following columns:

Column	Description
PLUGIN_NAME	Name of the plugin.
PLUGIN_VERSION	Version from the plugin's general type descriptor.
PLUGIN_STATUS	Plugin status, one of ACTIVE , INACTIVE , DISABLED , DELETED or NOT INSTALLED .

PLUGIN_TYPE	Plugin type: STORAGE ENGINE , INFORMATION_SCHEMA , AUTHENTICATION , REPLICATION , DAEMON or AUDIT .
PLUGIN_TYPE_VERSION	Version from the plugin's type-specific descriptor.
PLUGIN_LIBRARY	Plugin's shared object file name, located in the directory specified by the  <code>plugin_dir</code>  system variable, and used by the  <code>INSTALL PLUGIN</code>  and  <code>UNINSTALL PLUGIN</code>  statements. NULL if the plugin is compiled in and cannot be uninstalled.
PLUGIN_LIBRARY_VERSION	Version from the plugin's API interface.
PLUGIN_AUTHOR	Author of the plugin.
PLUGIN_DESCRIPTION	Description.
PLUGIN_LICENSE	Plugin's licence.
LOAD_OPTION	How the plugin was loaded; one of OFF , ON , FORCE or FORCE_PLUS_PERMANENT . See <a href="#">Installing Plugins</a> .
PLUGIN_MATURITY	Plugin's maturity level; one of Unknown , Experimental , Alpha , Beta , 'Gamma , and Stable .

PLUGIN_AUTH_VERSION	Plugin's version as determined by the plugin author. An example would be '0.99 beta 1'.
---------------------	---

It provides a superset of the information shown by the

`SHOW PLUGINS SONAME`

statement, as well as the

`information_schema.PLUGINS`

table. For specific information about storage engines (a particular type of plugin), see the [Information Schema ENGINES table](#) and the

`SHOW ENGINES`

statement.

The table is not a standard Information Schema table, and is a MariaDB extension.

## Example

```

SELECT * FROM information_schema.all_plugins\G
***** 1. row *****
  PLUGIN_NAME: binlog
  PLUGIN_VERSION: 1.0
  PLUGIN_STATUS: ACTIVE
  PLUGIN_TYPE: STORAGE ENGINE
  PLUGIN_TYPE_VERSION: 100314.0
  PLUGIN_LIBRARY: NULL
  PLUGIN_LIBRARY_VERSION: NULL
  PLUGIN_AUTHOR: MySQL AB
  PLUGIN_DESCRIPTION: This is a pseudo storage engine to represent the binlog in a transaction
  PLUGIN_LICENSE: GPL
  LOAD_OPTION: FORCE
  PLUGIN_MATURITY: Stable
  PLUGIN_AUTH_VERSION: 1.0
*****
***** 2. row *****
  PLUGIN_NAME: mysql_native_password
  PLUGIN_VERSION: 1.0
  PLUGIN_STATUS: ACTIVE
  PLUGIN_TYPE: AUTHENTICATION
  PLUGIN_TYPE_VERSION: 2.1
  PLUGIN_LIBRARY: NULL
  PLUGIN_LIBRARY_VERSION: NULL
  PLUGIN_AUTHOR: R.J.Silk, Sergei Golubchik
  PLUGIN_DESCRIPTION: Native MySQL authentication
  PLUGIN_LICENSE: GPL
  LOAD_OPTION: FORCE
  PLUGIN_MATURITY: Stable
  PLUGIN_AUTH_VERSION: 1.0
*****
***** 3. row *****
  PLUGIN_NAME: mysql_old_password
  PLUGIN_VERSION: 1.0
  PLUGIN_STATUS: ACTIVE
  PLUGIN_TYPE: AUTHENTICATION
  PLUGIN_TYPE_VERSION: 2.1
  PLUGIN_LIBRARY: NULL
  PLUGIN_LIBRARY_VERSION: NULL
  PLUGIN_AUTHOR: R.J.Silk, Sergei Golubchik
  PLUGIN_DESCRIPTION: Old MySQL-4.0 authentication
  PLUGIN_LICENSE: GPL
  LOAD_OPTION: FORCE
  PLUGIN_MATURITY: Stable
  PLUGIN_AUTH_VERSION: 1.0
...
*****
***** 104. row *****
  PLUGIN_NAME: WSREP_MEMBERSHIP
  PLUGIN_VERSION: 1.0
  PLUGIN_STATUS: NOT INSTALLED
  PLUGIN_TYPE: INFORMATION SCHEMA
  PLUGIN_TYPE_VERSION: 100314.0
  PLUGIN_LIBRARY: wsrep_info.so
  PLUGIN_LIBRARY_VERSION: 1.13
  PLUGIN_AUTHOR: Nirbhay Choubey
  PLUGIN_DESCRIPTION: Information about group members
  PLUGIN_LICENSE: GPL
  LOAD_OPTION: OFF
  PLUGIN_MATURITY: Stable
  PLUGIN_AUTH_VERSION: 1.0
*****
***** 105. row *****
  PLUGIN_NAME: WSREP_STATUS
  PLUGIN_VERSION: 1.0
  PLUGIN_STATUS: NOT INSTALLED
  PLUGIN_TYPE: INFORMATION SCHEMA
  PLUGIN_TYPE_VERSION: 100314.0
  PLUGIN_LIBRARY: wsrep_info.so
  PLUGIN_LIBRARY_VERSION: 1.13
  PLUGIN_AUTHOR: Nirbhay Choubey
  PLUGIN_DESCRIPTION: Group view information
  PLUGIN_LICENSE: GPL
  LOAD_OPTION: OFF
  PLUGIN_MATURITY: Stable

```

## 1.1.2.9.1.1.5 Information Schema APPLICABLE\_ROLES Table

The [Information Schema](#)

APPLICABLE\_ROLES

table shows the [role authorizations](#) that the current user may use.

It contains the following columns:

Column	Description	Added
GRANTEE	Account that the role was granted to.	
ROLE_NAME	Name of the role.	
IS_GRANTABLE	Whether the role can be granted or not.	
IS_DEFAULT	Whether the role is the user's default role or not	MariaDB 10.1.3

The current role is in the [ENABLED\\_ROLES](#) Information Schema table.

## Example

```
SELECT * FROM information_schema.APPLICABLE_ROLES;
+-----+-----+-----+
| GRANTEE      | ROLE_NAME    | IS_GRANTABLE | IS_DEFAULT |
+-----+-----+-----+
| root@localhost | journalist   | YES          | NO         |
| root@localhost | staff        | YES          | NO         |
| root@localhost | dd           | YES          | NO         |
| root@localhost | dog          | YES          | NO         |
+-----+-----+-----+
```

## 1.1.2.9.1.1.6 Information Schema CHARACTER\_SETS Table

The [Information Schema](#)

CHARACTER\_SETS

table contains a list of supported [character sets](#), their default collations and maximum lengths.

It contains the following columns:

Column	Description
CHARACTER_SET_NAME	Name of the character set.
DEFAULT_COLLATE_NAME	Default collation used.
DESCRIPTION	Character set description.
MAXLEN	Maximum length.

The [SHOW CHARACTER SET](#) statement returns the same results (although in a different order), and both can be refined in the same way. For example, the following two statements return the same results:

```
SHOW CHARACTER SET WHERE Maxlen LIKE '2';
```

and

```
SELECT * FROM information_schema.CHARACTER_SETS
WHERE MAXLEN LIKE '2';
```

See [Setting Character Sets and Collations](#) for details on specifying the character set at the server, database, table and column levels, and [Supported Character Sets and Collations](#) for a full list of supported characters sets and collations.

## Example

```
SELECT CHARACTER_SET_NAME FROM information_schema.CHARACTER_SETS
WHERE DEFAULT_COLLATE_NAME LIKE '%chinese%';
+-----+
| CHARACTER_SET_NAME |
+-----+
| big5                |
| gb2312              |
| gbk                 |
+-----+
```

## 1.1.2.9.1.1.7 Information Schema CHECK\_CONSTRAINTS Table

MariaDB starting with [10.2.22](#)

The Information Schema CHECK\_CONSTRAINTS Table was introduced in [MariaDB 10.3.10](#) and [MariaDB 10.2.22](#).

The [Information Schema](#)

CHECK\_CONSTRAINTS  
table stores metadata about the [constraints](#) defined for tables in all databases.

It contains the following columns:

Column	Description
CONSTRAINT_CATALOG	Always contains the string 'def'.
CONSTRAINT_SCHEMA	Database name.
TABLE_NAME	Table name.
CONSTRAINT_NAME	Constraint name.

MariaDB starting with [10.5.10](#)

LEVEL	Type of the constraint ('Column' or 'Table').
-------	---

CHECK_CLAUSE	Constraint clause.
--------------	--------------------

## Example

A table with a numeric table check constraint and with a default check constraint name:

```
CREATE TABLE t ( a int, CHECK (a>10));
```

To see check constraint call

```
check_constraints  
table from information schema .
```

```
SELECT * from INFORMATION_SCHEMA.CHECK_CONSTRAINTS\G
```

```
***** 1. row *****  
CONSTRAINT_CATALOG: def  
CONSTRAINT_SCHEMA: test  
CONSTRAINT_NAME: CONSTRAINT_1  
TABLE_NAME: t  
CHECK_CLAUSE: `a` > 10
```

A new table check constraint called

```
a_upper  
:
```

```
ALTER TABLE t ADD CONSTRAINT a_upper CHECK (a<100);
```

```
SELECT * from INFORMATION_SCHEMA.CHECK_CONSTRAINTS\G
```

```
***** 1. row *****  
CONSTRAINT_CATALOG: def  
CONSTRAINT_SCHEMA: test  
CONSTRAINT_NAME: CONSTRAINT_1  
TABLE_NAME: t  
CHECK_CLAUSE: `a` > 10  
***** 2. row *****  
CONSTRAINT_CATALOG: def  
CONSTRAINT_SCHEMA: test  
CONSTRAINT_NAME: a_upper  
TABLE_NAME: t  
CHECK_CLAUSE: `a` < 100
```

A new table

```
tt
```

with a field check constraint called

```
b
```

, as well as a table check constraint called

```
b_upper  
:
```

```
CREATE TABLE tt(b int CHECK(b>0),CONSTRAINT b_upper CHECK(b<50));
```

```
SELECT * from INFORMATION_SCHEMA.CHECK_CONSTRAINTS;
```

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_NAME	CHECK_CLAUSE
def	test	b	tt	`b` > 0
def	test	b_upper	tt	`b` < 50
def	test	CONSTRAINT_1	t	`a` > 10
def	test	a_upper	t	`a` < 100

Note: The name of the field constraint is the same as the field name.

After dropping the default table constraint called

```
CONSTRAINT_1  
:
```

```

ALTER TABLE t DROP CONSTRAINT CONSTRAINT_1;

SELECT * FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS;
+-----+-----+-----+-----+-----+
| CONSTRAINT_CATALOG | CONSTRAINT_SCHEMA | CONSTRAINT_NAME | TABLE_NAME | CHECK_CLAUSE |
+-----+-----+-----+-----+
| def                | test            | b                 | tt          | `b` > 0      |
| def                | test            | b_upper           | tt          | `b` < 50     |
| def                | test            | a_upper           | t           | `a` < 100    |
+-----+-----+-----+-----+

```

Trying to insert invalid arguments into table

t  
and  
tt  
generates an error.

```

INSERT INTO t VALUES (10),(20),(100);
ERROR 4025 (23000): CONSTRAINT `a_upper` failed for `test`.`t`

INSERT INTO tt VALUES (10),(-10),(100);
ERROR 4025 (23000): CONSTRAINT `b` failed for `test`.`tt`

INSERT INTO tt VALUES (10),(20),(100);
ERROR 4025 (23000): CONSTRAINT `b_upper` failed for `test`.`tt`

```

From MariaDB 10.5.10 :

```

create table majra(check(x>0), x int, y int check(y < 0), z int,
                    constraint z check(z>0), constraint xyz check(x<10 and y<10 and z<10));
Query OK, 0 rows affected (0.036 sec)

show create table majra;
+-----+-----+
| Table | Create Table
+-----+-----+
| majra | CREATE TABLE `majra` (
  `x` int(11) DEFAULT NULL,
  `y` int(11) DEFAULT NULL CHECK (`y` < 0),
  `z` int(11) DEFAULT NULL,
  CONSTRAINT `CONSTRAINT_1` CHECK (`x` > 0),
  CONSTRAINT `z` CHECK (`z` > 0),
  CONSTRAINT `xyz` CHECK (`x` < 10 and `y` < 10 and `z` < 10)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
+-----+-----+
1 row in set (0.000 sec)

select * from information_schema.check_constraints where table_name='majra';
+-----+-----+-----+-----+-----+-----+
| CONSTRAINT_CATALOG | CONSTRAINT_SCHEMA | TABLE_NAME | CONSTRAINT_NAME | LEVEL | CHECK_CLAUSE |
+-----+-----+-----+-----+-----+
| def                | test            | majra     | y             | Column | `y` < 0      |
| def                | test            | majra     | CONSTRAINT_1  | Table  | `x` > 0      |
| def                | test            | majra     | z             | Table  | `z` > 0      |
| def                | test            | majra     | xyz           | Table  | `x` < 10 and `y` < 10 and `z` < 10 |
+-----+-----+-----+-----+-----+
4 rows in set (0.001 sec)

```

## 1.1.2.9.1.1.8 Information Schema CLIENT\_STATISTICS Table

The [Information Schema](#)

`CLIENT_STATISTICS`

table holds statistics about client connections. This is part of the [User Statistics](#) feature, which is not enabled by default.

It contains the following columns:

Field	Type	Notes
-------	------	-------

CLIENT	VARCHAR(64)	The IP address or hostname the connection originated from.
TOTAL_CONNECTIONS	INT(21)	The number of connections created for this client.
CONCURRENT_CONNECTIONS	INT(21)	The number of concurrent connections for this client.
CONNECTED_TIME	INT(21)	The cumulative number of seconds elapsed while there were connections from this client.
BUSY_TIME	DOUBLE	The cumulative number of seconds there was activity on connections from this client.
CPU_TIME	DOUBLE	The cumulative CPU time elapsed while servicing this client's connections. Note that this number may be wrong on SMP system if there was a CPU migration for the thread during the execution of the query.
BYTES_RECEIVED	INT(21)	The number of bytes received from this client's connections.
BYTES_SENT	INT(21)	The number of bytes sent to this client's connections.
BINLOG_BYTES_WRITTEN	INT(21)	The number of bytes written to the <a href="#">binary log</a> from this client's connections.
ROWS_READ	INT(21)	The number of rows read by this client's connections.
ROWS_SENT	INT(21)	The number of rows sent by this client's connections.
ROWS_DELETED	INT(21)	The number of rows deleted by this client's connections.
ROWS_INSERTED	INT(21)	The number of rows inserted by this client's connections.
ROWS_UPDATED	INT(21)	The number of rows updated by this client's connections.
SELECT_COMMANDS	INT(21)	The number of  SELECT  commands executed from this client's connections.

UPDATE_COMMANDS	INT(21)	The number of  <b>UPDATE</b>  commands executed from this client's connections.
OTHER_COMMANDS	INT(21)	The number of other commands executed from this client's connections.
COMMIT_TRANSACTIONS	INT(21)	The number of  <b>COMMIT</b>  commands issued by this client's connections.
ROLLBACK_TRANSACTIONS	INT(21)	The number of  <b>ROLLBACK</b>  commands issued by this client's connections.
DENIED_CONNECTIONS	INT(21)	The number of connections denied to this client.
LOST_CONNECTIONS	INT(21)	The number of this client's connections that were terminated uncleanly.
ACCESS_DENIED	INT(21)	The number of times this client's connections issued commands that were denied.
EMPTY_QUERIES	INT(21)	The number of times this client's connections sent queries that returned no results to the server.
TOTAL_SSL_CONNECTIONS	INT(21)	The number of <b>TLS connections</b> created for this client. (>= <a href="#">MariaDB 10.1.1</a> )
MAX_STATEMENT_TIME_EXCEEDED	INT(21)	The number of times a statement was aborted, because it was executed longer than its  <b>MAX_STATEMENT_TIME</b>  threshold. (>= <a href="#">MariaDB 10.1.1</a> )

## Example

```

SELECT * FROM information_schema.CLIENT_STATISTICS\G
***** 1. row *****
    CLIENT: localhost
    TOTAL_CONNECTIONS: 3
    CONCURRENT_CONNECTIONS: 0
    CONNECTED_TIME: 4883
        BUSY_TIME: 0.009722
        CPU_TIME: 0.0102131
    BYTES RECEIVED: 841
        BYTES SENT: 13897
    BINLOG_BYTES_WRITTEN: 0
        ROWS_READ: 0
        ROWS_SENT: 214
        ROWS_DELETED: 0
        ROWS_INSERTED: 207
        ROWS_UPDATED: 0
    SELECT_COMMANDS: 10
    UPDATE_COMMANDS: 0
    OTHER_COMMANDS: 13
    COMMIT_TRANSACTIONS: 0
    ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
    LOST_CONNECTIONS: 0
    ACCESS_DENIED: 0
    EMPTY_QUERIES: 1

```

## 1.1.2.9.1.1.9 Information Schema COLLATION\_CHARACTER\_SET\_APPLICABILITY Table

The [Information Schema](#)

`COLLATION_CHARACTER_SET_APPLICABILITY`  
table shows which [character sets](#) are associated with which collations.

It contains the following columns:

Column	Description
<code>COLLATION_NAME</code>	Collation name.
<code>CHARACTER_SET_NAME</code>	Name of the associated character set.

`COLLATION_CHARACTER_SET_APPLICABILITY`  
is essentially a subset of the

[COLLATIONS](#)

table.

```
SELECT COLLATION_NAME,CHARACTER_SET_NAME FROM information_schema.COLLATIONS;
```

and

```
SELECT * FROM information_schema.COLLATION_CHARACTER_SET_APPLICABILITY;
```

will return identical results.

See [Setting Character Sets and Collations](#) for details on specifying the character set at the server, database, table and column levels.

## Example

```

SELECT * FROM information_schema.COLLATION_CHARACTER_SET_APPLICABILITY
WHERE CHARACTER_SET_NAME='utf32';
+-----+-----+
| COLLATION_NAME | CHARACTER_SET_NAME |
+-----+-----+
| utf32_general_ci | utf32 |
| utf32_bin | utf32 |
| utf32_unicode_ci | utf32 |
| utf32_icelandic_ci | utf32 |
| utf32_latvian_ci | utf32 |
| utf32_romanian_ci | utf32 |
| utf32_slovenian_ci | utf32 |
| utf32_polish_ci | utf32 |
| utf32_estonian_ci | utf32 |
| utf32_spanish_ci | utf32 |
| utf32_swedish_ci | utf32 |
| utf32_turkish_ci | utf32 |
| utf32_czech_ci | utf32 |
| utf32_danish_ci | utf32 |
| utf32_lithuanian_ci | utf32 |
| utf32_slovak_ci | utf32 |
| utf32_spanish2_ci | utf32 |
| utf32_roman_ci | utf32 |
| utf32_persian_ci | utf32 |
| utf32_esperanto_ci | utf32 |
| utf32_hungarian_ci | utf32 |
| utf32_sinhala_ci | utf32 |
| utf32_german2_ci | utf32 |
| utf32_croatian_ci | utf32 |
+-----+-----+

```

## 1.1.2.9.1.1.10 Information Schema COLLATIONS Table

### Contents

1. [NO PAD collations](#)
2. [Example](#)
3. [See Also](#)

The [Information Schema](#)

`COLLATIONS`

table contains a list of supported [collations](#).

It contains the following columns:

Column	Description
<code>COLLATION_NAME</code>	Name of the collation.
<code>CHARACTER_SET_NAME</code>	Associated character set.
<code>ID</code>	Collation id.
<code>IS_DEFAULT</code>	Whether the collation is the character set's default.
<code>IS_COMPILED</code>	Whether the collation is compiled into the server.
<code>SORTLEN</code>	Sort length, used for determining the memory used to sort strings in this collation.

The [SHOW COLLATION](#) statement returns the same results and both can be reduced in a similar way.

For example, in MariaDB Server 10.6, the following two statements return the same results:

```
SHOW COLLATION WHERE Charset LIKE 'utf8mb3';
```

and

```
SELECT * FROM information_schema.COLLATIONS
WHERE CHARACTER_SET_NAME LIKE 'utf8mb3';
```

In MariaDB Server 10.5 and before,

utf8  
should be specified instead of  
utf8mb3

## NO PAD collations

MariaDB starting with 10.2

NO PAD  
collations regard trailing spaces as normal characters. You can get a list of all  
NO PAD  
collations as follows:

```
SELECT collation_name FROM information_schema.COLLATIONS
WHERE collation_name LIKE "%noplod%";
```

collation_name
big5_chinese_noplod_ci
big5_noplod_bin
...

## Example

```
SELECT * FROM information_schema.COLLATIONS;
```

COLLATION_NAME	CHARACTER_SET_NAME	ID	IS_DEFAULT	IS_COMPILED	SORTLEN
big5_chinese_ci	big5	1	Yes	Yes	1
big5_bin	big5	84		Yes	1
big5_chinese_noplod_ci	big5	1025		Yes	1
big5_noplod_bin	big5	1108		Yes	1
dec8_swedish_ci	dec8	3	Yes	Yes	1
dec8_bin	dec8	69		Yes	1
dec8_swedish_noplod_ci	dec8	1027		Yes	1
dec8_noplod_bin	dec8	1093		Yes	1
cp850_general_ci	cp850	4	Yes	Yes	1
cp850_bin	cp850	80		Yes	1
...					

## See Also

- [Setting Character Sets and Collations](#) - specifying the character set at the server, database, table and column levels
- [Supported Character Sets and Collations](#) - full list of supported characters sets and collations.

## 1.1.2.9.1.1.11 Information Schema COLUMN\_PRIVILEGES Table

The [Information Schema](#)

COLUMN\_PRIVILEGES  
table contains column privilege information derived from the

[mysql.columns\\_priv](#)

grant table.

It has the following columns:

Column	Description
--------	-------------

GRANTEE	In the format user_name@host_name . . .
TABLE_CATALOG	Always def . . .
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
COLUMN_NAME	Column name.
PRIVILEGE_TYPE	One of SELECT , INSERT , UPDATE OR REFERENCES . . .
IS_GRANTABLE	Whether the user has the <b>GRANT OPTION</b> for this privilege.

Similar information can be accessed with the

`SHOW FULL COLUMNS`

and

`SHOW GRANTS`

statements. See the

`GRANT`

article for more about privileges.

This information is also stored in the

`columns_priv`

table, in the

`mysql`

system database.

For a description of the privileges that are shown in this table, see [column privileges](#).

## Example

In the following example, no column-level privilege has been explicitly assigned:

```
SELECT * FROM information_schema.COLUMN_PRIVILEGES;
Empty set
```

## 1.1.2.9.1.1.12 Information Schema COLUMNS Table

The [Information Schema](#)

`COLUMNS`

table provides information about columns in each table on the server.

It contains the following columns:

Column	Description	Introduced
TABLE_CATALOG	Always contains the string 'def'.	
TABLE_SCHEMA	Database name.	
TABLE_NAME	Table name.	
COLUMN_NAME	Column name.	
ORDINAL_POSITION	Column position in the table. Can be used for ordering.	
COLUMN_DEFAULT	<p>Default value for the column. From <a href="#">MariaDB 10.2.7</a>, literals are quoted to distinguish them from expressions.</p> <p>NULL means that the column has no default. In <a href="#">MariaDB 10.2.6</a> and earlier, no quotes were used for any type of default and</p> <p>NULL can either mean that there is no default, or that the default column value is</p> <p>NULL .</p>	
IS_NULLABLE	Whether the column can contain NULL s.	
DATA_TYPE	The column's <a href="#">data type</a> .	
CHARACTER_MAXIMUM_LENGTH	Maximum length.	
CHARACTER_OCTET_LENGTH	Same as the CHARACTER_MAXIMUM_LENGTH except for multi-byte <a href="#">character sets</a> .	
NUMERIC_PRECISION	For numeric types, the precision (number of significant digits) for the column. NULL if not a numeric field.	
NUMERIC_SCALE	For numeric types, the scale (significant digits to the right of the decimal point). NULL if not a numeric field.	
DATETIME_PRECISION	Fractional-seconds precision, or NULL if not a <a href="#">time data type</a> .	
CHARACTER_SET_NAME	<a href="#">Character set</a> if a non-binary <a href="#">string data type</a> , otherwise NULL.	
COLLATION_NAME	<a href="#">Collation</a> if a non-binary <a href="#">string data type</a> , otherwise NULL.	
COLUMN_TYPE	Column definition, a MySQL and MariaDB extension.	

COLUMN_KEY	Index type. PRI for primary key, UNI for unique index, MUL for multiple index. A MySQL and MariaDB extension.	
EXTRA	Additional information about a column, for example whether the column is an <a href="#">invisible column</a> , or, from <a href="#">MariaDB 10.3.6</a> , WITHOUT SYSTEM VERSIONING if the table is not a <a href="#">system-versioned table</a> . A MySQL and MariaDB extension.	
PRIVILEGES	Which privileges you have for the column. A MySQL and MariaDB extension.	
COLUMN_COMMENT	Column comments.	
IS_GENERATED	Indicates whether the column value is <a href="#">generated (virtual, or computed)</a> . Can be ALWAYS or NEVER	MariaDB 10.2.5
GENERATION_EXPRESSION	The expression used for computing the column value in a <a href="#">generated (virtual, or computed)</a> column.	MariaDB 10.2.5

It provides information similar to, but more complete, than

`SHOW COLUMNS`

and

`mysqlshow`

## Examples

```
SELECT * FROM information_schema.COLUMNS\G
...
***** 9. row *****
TABLE_CATALOG: def
TABLE_SCHEMA: test
TABLE_NAME: t2
COLUMN_NAME: j
ORDINAL_POSITION: 1
COLUMN_DEFAULT: NULL
COLUMN_NULLABLE: YES
DATA_TYPE: longtext
CHARACTER_MAXIMUM_LENGTH: 4294967295
CHARACTER_OCTET_LENGTH: 4294967295
NUMERIC_PRECISION: NULL
NUMERIC_SCALE: NULL
DATETIME_PRECISION: NULL
CHARACTER_SET_NAME: utf8mb4
COLLATION_NAME: utf8mb4_bin
COLUMN_TYPE: longtext
COLUMN_KEY:
EXTRA:
PRIVILEGES: select,insert,update,references
COLUMN_COMMENT:
IS_GENERATED: NEVER
GENERATION_EXPRESSION: NULL
...
```

```

CREATE TABLE t (
    s1 VARCHAR(20) DEFAULT 'ABC',
    s2 VARCHAR(20) DEFAULT (concat('A','B')),
    s3 VARCHAR(20) DEFAULT ("concat('A','B')"),
    s4 VARCHAR(20),
    s5 VARCHAR(20) DEFAULT NULL,
    s6 VARCHAR(20) NOT NULL,
    s7 VARCHAR(20) DEFAULT 'NULL' NULL,
    s8 VARCHAR(20) DEFAULT 'NULL' NOT NULL
);

SELECT
    table_name,
    column_name,
    ordinal_position,
    column_default,
    column_default IS NULL
FROM information_schema.COLUMNS
WHERE table_schema=DATABASE()
AND TABLE_NAME='t';

```

From MariaDB 10.2.7 :

table_name	column_name	ordinal_position	column_default	column_default IS NULL
t	s1	1	'ABC'	0
t	s2	2	concat('A','B')	0
t	s3	3	"concat('A','B')"	0
t	s4	4	NULL	0
t	s5	5	NULL	0
t	s6	6	NULL	1
t	s7	7	'NULL'	0
t	s8	8	'NULL'	0

In the results above, the two single quotes in

concat(''A'', ''B'')

indicate an escaped single quote - see [string-literals](#). Note that while [mysql-command-line-client](#) appears to show the same default value for columns

s5  
and  
s6

, the first is a 4-character string "NULL", while the second is the SQL NULL value.

MariaDB 10.2.6 and before:

table_name	column_name	ordinal_position	column_default	column_default IS NULL
t	s1	1	ABC	0
t	s2	2	concat('A','B')	0
t	s3	3	concat('A','B')	0
t	s4	4	NULL	1
t	s5	5	NULL	1
t	s6	6	NULL	1
t	s7	7	NULL	0
t	s8	8	NULL	0

## 1.1.2.9.1.1.13 Information Schema DISKS Table

MariaDB 10.1.32

The

DISKS

table was introduced in [MariaDB 10.1.32](#) , [MariaDB 10.2.14](#) , and [MariaDB 10.3.6](#) as part of the

DISKS

plugin.

## Contents

1. [Description](#)
2. [Example](#)
3. [See Also](#)

## Description

The

DISKS

table is created when the [DISKS](#) plugin is enabled, and shows metadata about disks on the system.

Before [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#) and [MariaDB 10.1.41](#) , this plugin did **not** check [user privileges](#) . When it is enabled, **any** user can query the

`INFORMATION_SCHEMA.DISKS`

table and see all the information it provides.

Since [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#) and [MariaDB 10.1.41](#) , it requires the [FILE privilege](#) .

The plugin only works on Linux.

The table contains the following columns:

Column	Description
DISK	Name of the disk itself.
PATH	Mount point of the disk.
TOTAL	Total space in KiB.
USED	Used amount of space in KiB.
AVAILABLE	Amount of space in KiB available to non-root users.

Note that as the amount of space available to root (OS user) may be more than what is available to non-root users, 'available' + 'used' may be less than 'total'.

All paths to which a particular disk has been mounted are reported. The rationale is that someone might want to take different action e.g. depending on which disk is relevant for a particular path. This leads to the same disk being reported multiple times.

## Example

```
SELECT * FROM information_schema.DISKS;
```

```
+-----+-----+-----+-----+
| Disk   | Path  | Total   | Used    | Available |
+-----+-----+-----+-----+
| /dev/vda1 | /   | 26203116 | 2178424 | 24024692 |
| /dev/vda1 | /boot | 26203116 | 2178424 | 24024692 |
| /dev/vda1 | /etc  | 26203116 | 2178424 | 24024692 |
+-----+-----+-----+-----+
```

## See Also

- [Disks Plugin](#) for details on installing, options
- [Plugin Overview](#) for details on managing plugins.

## 1.1.2.9.1.1.14 Information Schema ENABLED\_ROLES Table

The [Information Schema](#)

`ENABLED_ROLES`

table shows the enabled [roles](#) for the current session.

It contains the following column:

Column	Description
ROLE_NAME	The enabled role name, or NULL .

This table lists all roles that are currently enabled, one role per row — the current role, roles granted to the current role, roles granted to these roles and so on. If no role is set, the row contains a

NULL  
value.

The roles that the current user can enable are listed in the [APPLICABLE\\_ROLES](#) Information Schema table.

See also [CURRENT\\_ROLE\(\)](#).

## Examples

```
SELECT * FROM information_schema.ENABLED_ROLES;
+-----+
| ROLE_NAME |
+-----+
| NULL      |
+-----+
SET ROLE staff;

SELECT * FROM information_schema.ENABLED_ROLES;
+-----+
| ROLE_NAME |
+-----+
| staff     |
+-----+
```

## 1.1.2.9.1.1.15 Information Schema ENGINES Table

The [Information Schema](#)

ENGINES  
table displays status information about the server's [storage engines](#).

It contains the following columns:

Column	Description
ENGINE	Name of the storage engine.
SUPPORT	Whether the engine is the default, or is supported or not.
COMMENT	Storage engine comments.
TRANSACTIONS	Whether or not the engine supports <a href="#">transactions</a> .
XA	Whether or not the engine supports <a href="#">XA transactions</a> .
SAVEPOINTS	Whether or not <a href="#">savepoints</a> are supported.

It provides identical information to the

[SHOW ENGINES](#)

statement. Since storage engines are plugins, different information about them is also shown in the

```
information_schema.PLUGINS
```

table and by the

```
SHOW PLUGINS
```

statement.

The table is not a standard Information Schema table, and is a MySQL and MariaDB extension.

Note that both MySQL's InnoDB and Percona's XtraDB replacement are labeled as

```
InnoDB
```

. However, if XtraDB is in use, it will be specified in the

```
COMMENT
```

field. See [XtraDB and InnoDB](#) . The same applies to [FederatedX](#) .

## Example

```

SELECT * FROM information_schema.ENGINES\G;
***** 1. row *****
ENGINE: InnoDB
SUPPORT: DEFAULT
COMMENT: Supports transactions, row-level locking, and foreign keys
TRANSACTIONS: YES
XA: YES
SAVEPOINTS: YES
***** 2. row *****
ENGINE: CSV
SUPPORT: YES
COMMENT: CSV storage engine
TRANSACTIONS: NO
XA: NO
SAVEPOINTS: NO
***** 3. row *****
ENGINE: MyISAM
SUPPORT: YES
COMMENT: MyISAM storage engine
TRANSACTIONS: NO
XA: NO
SAVEPOINTS: NO
***** 4. row *****
ENGINE: BLACKHOLE
SUPPORT: YES
COMMENT: /dev/null storage engine (anything you write to it disappears)
TRANSACTIONS: NO
XA: NO
SAVEPOINTS: NO
***** 5. row *****
ENGINE: FEDERATED
SUPPORT: YES
COMMENT: FederatedX pluggable storage engine
TRANSACTIONS: YES
XA: NO
SAVEPOINTS: YES
***** 6. row *****
ENGINE: MRG_MyISAM
SUPPORT: YES
COMMENT: Collection of identical MyISAM tables
TRANSACTIONS: NO
XA: NO
SAVEPOINTS: NO
***** 7. row *****
ENGINE: ARCHIVE
SUPPORT: YES
COMMENT: Archive storage engine
TRANSACTIONS: NO
XA: NO
SAVEPOINTS: NO
***** 8. row *****
ENGINE: MEMORY
SUPPORT: YES
COMMENT: Hash based, stored in memory, useful for temporary tables
TRANSACTIONS: NO
XA: NO
SAVEPOINTS: NO
***** 9. row *****
ENGINE: PERFORMANCE_SCHEMA
SUPPORT: YES
COMMENT: Performance Schema
TRANSACTIONS: NO
XA: NO
SAVEPOINTS: NO
***** 10. row *****
ENGINE: Aria
SUPPORT: YES
COMMENT: Crash-safe tables with MyISAM heritage
TRANSACTIONS: NO
XA: NO
SAVEPOINTS: NO
10 rows in set (0.00 sec)

```

Check if a given storage engine is available:

```
SELECT SUPPORT FROM information_schema.ENGINES WHERE ENGINE LIKE 'tokudb';
Empty set
```

Check which storage engine supports XA transactions:

```
SELECT ENGINE FROM information_schema.ENGINES WHERE XA = 'YES';
+-----+
| ENGINE |
+-----+
| InnoDB |
+-----+
```

## 1.1.2.9.1.1.16 Information Schema EVENTS Table

The [Information Schema](#)

EVENTS

table stores information about [Events](#) on the server.

It contains the following columns:

Column	Description
EVENT_CATALOG	Always def
EVENT_SCHEMA	Database where the event was defined.
EVENT_NAME	Event name.
DEFINER	Event definer.
TIME_ZONE	Time zone used for the event's scheduling and execution, by default SYSTEM
EVENT_BODY	SQL
EVENT_DEFINITION	The SQL defining the event.
EVENT_TYPE	Either ONE TIME or RECURRING
EXECUTE_AT	DATETIME when the event is set to execute, or NULL if recurring.
INTERVAL_VALUE	Numeric interval between event executions for a recurring event, or NULL if not recurring.
INTERVAL_FIELD	Interval unit (e.g., HOUR )

SQL_MODE	The  <code>SQL_MODE</code>  at the time the event was created.
STARTS	Start  <code>DATETIME</code>  for a recurring event, NULL if not defined or not recurring.
ENDS	End  <code>DATETIME</code>  for a recurring event, NULL if not defined or not recurring.
STATUS	One of  ENABLED , DISABLED or / SLAVESIDE_DISABLED  .
ON_COMPLETION	The  ON COMPLETION clause, either PRESERVE or NOT PRESERVE  .
CREATED	When the event was created.
LAST_ALTERED	When the event was last changed.
LAST_EXECUTED	When the event was last run.
EVENT_COMMENT	The comment provided in the  <code>CREATE EVENT</code>  statement, or an empty string if none.
ORIGINATOR	MariaDB server ID on which the event was created.
CHARACTER_SET_CLIENT	 <code>character_set_client</code>  system variable session value at the time the event was created.
COLLATION_CONNECTION	 <code>collation_connection</code> system variable session value at the time the event was created.
DATABASE_COLLATION	Database <code>collation</code> with which the event is linked.

The

```
SHOW EVENTS
```

and

```
SHOW CREATE EVENT
```

statements provide similar information.

## 1.1.2.9.1.1.17 Information Schema FEEDBACK Table

The [Information Schema](#)

FEEDBACK

table is created when the [Feedback Plugin](#) is enabled, and contains the complete contents submitted by the plugin.

It contains two columns:

Column	Description
VARIABLE_NAME	Name of the item of information being collected.
VARIABLE_VALUE	Contents of the item of information being collected.

It is possible to disable automatic collection, by setting the

```
feedback_url
```

variable to an empty string, and to submit the contents manually, as follows:

```
$ mysql -e 'SELECT * FROM information_schema.FEEDBACK' > report.txt
```

Then you can send it by opening

[https://mariadb.org/feedback\\_plugin/post](https://mariadb.org/feedback_plugin/post)

in your browser, and uploading your generated

report.txt

. Or you can do it from the command line with (for example):

```
$ curl -F data=@report.txt https://mariadb.org/feedback_plugin/post
```

Manual uploading allows you to be absolutely sure that we receive only the data shown in the

information\_schema.FEEDBACK

table and that no private or sensitive information is being sent.

## Example

```

SELECT * FROM information_schema.FEEDBACK\G
...
***** 906. row *****
VARIABLE_NAME: Uname_sysname
VARIABLE_VALUE: Linux
***** 907. row *****
VARIABLE_NAME: Uname_release
VARIABLE_VALUE: 3.13.0-53-generic
***** 908. row *****
VARIABLE_NAME: Uname_version
VARIABLE_VALUE: #89-Ubuntu SMP Wed May 20 10:34:39 UTC 2015
***** 909. row *****
VARIABLE_NAME: Uname_machine
VARIABLE_VALUE: x86_64
***** 910. row *****
VARIABLE_NAME: Uname_distribution
VARIABLE_VALUE: lsb: Ubuntu 14.04.2 LTS
***** 911. row *****
VARIABLE_NAME: Collation used latin1_german1_ci
VARIABLE_VALUE: 1
***** 912. row *****
VARIABLE_NAME: Collation used latin1_swedish_ci
VARIABLE_VALUE: 18
***** 913. row *****
VARIABLE_NAME: Collation used utf8_general_ci
VARIABLE_VALUE: 567
***** 914. row *****
VARIABLE_NAME: Collation used latin1_bin
VARIABLE_VALUE: 1
***** 915. row *****
VARIABLE_NAME: Collation used binary
VARIABLE_VALUE: 16
***** 916. row *****
VARIABLE_NAME: Collation used utf8_bin
VARIABLE_VALUE: 4044

```

## 1.1.2.9.1.1.18 Information Schema FILES Table

The

**FILES**

tables is unused in MariaDB. See [MDEV-11426](#).

## 1.1.2.9.1.1.19 Information Schema GEOMETRY\_COLUMNS Table

### Description

The [Information Schema](#)

**GEOMETRY\_COLUMNS**

table provides support for Spatial Reference systems for GIS data.

It contains the following columns:

Column	Type	Null	Description
F_TABLE_CATALOG	VARCHAR(512)	NO	Together with F_TABLE_SCHEMA and F_TABLE_NAME , the fully qualified name of the featured table containing the geometry column.
F_TABLE_SCHEMA	VARCHAR(64)	NO	Together with F_TABLE_CATALOG and F_TABLE_NAME , the fully qualified name of the featured table containing the geometry column.

F_TABLE_NAME	VARCHAR(64)	NO	Together with F_TABLE_CATALOG and F_TABLE_SCHEMA , the fully qualified name of the featured table containing the geometry column.
F_GEOMETRY_COLUMN	VARCHAR(64)	NO	Name of the column in the featured table that is the geometry column.
G_TABLE_CATALOG	VARCHAR(512)	NO	
G_TABLE_SCHEMA	VARCHAR(64)	NO	Database name of the table implementing the geometry column.
G_TABLE_NAME	VARCHAR(64)	NO	Table name that is implementing the geometry column.
G_GEOMETRY_COLUMN	VARCHAR(64)	NO	
STORAGE_TYPE	TINYINT(2)	NO	Binary geometry implementation. Always 1 in MariaDB.
GEOMETRY_TYPE	INT(7)	NO	Integer reflecting the type of geometry stored in this column (see table below).
COORD_DIMENSION	TINYINT(2)	NO	Number of dimensions in the spatial reference system. Always 2 in MariaDB.
MAX_PPR	TINYINT(2)	NO	Always 0 in MariaDB.
SRID	SMALLINT(5)	NO	ID of the Spatial Reference System used for the coordinate geometry in this table. It is a foreign key reference to the  <a href="#">SPATIAL_REF_SYS table</a>

## Storage\_type

The integers in the  
storage\_type  
field match the geometry types as follows:

Integer	Type
0	GEOMETRY
1	POINT

3	LINestring
5	POLYGON
7	MULTIPOINT
9	MULTILINESTRING
11	MULTIPOLYGON

## Example

```
CREATE TABLE g1(g GEOMETRY(9,4) REF_SYSTEM_ID=101);

SELECT * FROM information_schema.GEOMETRY_COLUMNS\G
***** 1. row *****
F_TABLE_CATALOG: def
F_TABLE_SCHEMA: test
F_TABLE_NAME: g1
F_GEOMETRY_COLUMN:
  G_TABLE_CATALOG: def
  G_TABLE_SCHEMA: test
  G_TABLE_NAME: g1
G_GEOMETRY_COLUMN: g
  STORAGE_TYPE: 1
  GEOMETRY_TYPE: 0
COORD_DIMENSION: 2
  MAX_PPR: 0
  SRID: 101
```

## See also

- The

[SPATIAL\\_REF\\_SYS](#)

table.

## 1.1.2.9.1.1.20 Information Schema GLOBAL\_STATUS and SESSION\_STATUS Tables

The [Information Schema](#)

GLOBAL\_STATUS

and

SESSION\_STATUS

tables store a record of all [status variables](#) and their global and session values respectively. This is the same information as displayed by the

[SHOW STATUS](#)

commands

`SHOW GLOBAL STATUS`

and

`SHOW SESSION STATUS`

They contain the following columns:

Column	Description
--------	-------------

VARIABLE_NAME	Status variable name.
VARIABLE_VALUE	Global or session value.

## Example

```
SELECT * FROM information_schema.GLOBAL_STATUS;
+-----+-----+
| VARIABLE_NAME          | VARIABLE_VALUE   |
+-----+-----+
...
| BINLOG_SNAPSHOT_FILE  | mariadb-bin.000208 |
| BINLOG_SNAPSHOT_POSITION | 369               |
...
| THREADS_CONNECTED      | 1                 |
| THREADS_CREATED        | 1                 |
| THREADS_RUNNING         | 1                 |
| UPTIME                  | 57358             |
| UPTIME_SINCE_FLUSH_STATUS | 57358             |
+-----+-----+
```

## 1.1.2.9.1.1.21 Information Schema GLOBAL\_VARIABLES and SESSION\_VARIABLES Tables

The [Information Schema](#)

GLOBAL\_VARIABLES

and

SESSION\_VARIABLES

tables stores a record of all [system variables](#) and their global and session values respectively. This is the same information as displayed by the

[SHOW VARIABLES](#)

commands

`SHOW GLOBAL VARIABLES`

and

`SHOW SESSION VARIABLES`

It contains the following columns:

Column	Description
VARIABLE_NAME	System variable name.
VARIABLE_VALUE	Global or session value.

## Example

```

SELECT * FROM information_schema.GLOBAL_VARIABLES ORDER BY VARIABLE_NAME\G
***** 1. row *****
VARIABLE_NAME: ARIA_BLOCK_SIZE
VARIABLE_VALUE: 8192
***** 2. row *****
VARIABLE_NAME: ARIA_CHECKPOINT_LOG_ACTIVITY
VARIABLE_VALUE: 1048576
***** 3. row *****
VARIABLE_NAME: ARIA_CHECKPOINT_INTERVAL
VARIABLE_VALUE: 30
...
***** 455. row *****
VARIABLE_NAME: VERSION_COMPILE_MACHINE
VARIABLE_VALUE: x86_64
***** 456. row *****
VARIABLE_NAME: VERSION_COMPILE_OS
VARIABLE_VALUE: debian-linux-gnu
***** 457. row *****
VARIABLE_NAME: WAIT_TIMEOUT
VARIABLE_VALUE: 600

```

## 1.1.2.9.1.1.22 Information Schema INDEX\_STATISTICS Table

The [Information Schema](#)

### INDEX\_STATISTICS

table shows statistics on index usage and makes it possible to do such things as locating unused indexes and generating the commands to remove them.

This is part of the [User Statistics](#) feature, which is not enabled by default.

It contains the following columns:

Field	Type	Notes
TABLE_SCHEMA	VARCHAR(192)	The schema (database) name.
TABLE_NAME	VARCHAR(192)	The table name.
INDEX_NAME	VARCHAR(192)	The index name (as visible in <a href="#">SHOW CREATE TABLE</a> ).
ROWS_READ	INT(21)	The number of rows read from this index.

## Example

```

SELECT * FROM information_schema.INDEX_STATISTICS
WHERE TABLE_NAME = "author";
+-----+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME | INDEX_NAME | ROWS_READ |
+-----+-----+-----+-----+
| books       | author     | by_name    |      15   |
+-----+-----+-----+-----+

```

## 1.1.2.9.1.1.23 Information Schema KEY\_CACHES Table

The [Information Schema](#)

### KEY\_CACHES

table shows statistics about the [segmented key cache](#) ,.

It contains the following columns:

Column Name	Description
-------------	-------------

KEY_CACHE_NAME	The name of the key cache
SEGMENTS	total number of segments (set to NULL for regular key caches)
SEGMENT_NUMBER	segment number (set to NULL for any regular key caches and for rows containing aggregation statistics for segmented key caches)
FULL_SIZE	memory for cache buffers/auxiliary structures
BLOCK_SIZE	size of the blocks
USED_BLOCKS	number of currently used blocks
UNUSED_BLOCKS	number of currently unused blocks
DIRTY_BLOCKS	number of currently dirty blocks
READ_REQUESTS	number of read requests
READS	number of actual reads from files into buffers
WRITE_REQUESTS	number of write requests
WRITES	number of actual writes from buffers into files

## Example

```

SELECT * FROM information_schema.KEY_CACHES \G
*****
1. row ****
KEY_CACHE_NAME: default
SEGMENTS: NULL
SEGMENT_NUMBER: NULL
  FULL_SIZE: 134217728
  BLOCK_SIZE: 1024
  USED_BLOCKS: 36
  UNUSED_BLOCKS: 107146
  DIRTY_BLOCKS: 0
  READ_REQUESTS: 40305
  READS: 21
  WRITE_REQUESTS: 19239
  WRITES: 358

```

## 1.1.2.9.1.1.24 Information Schema KEY\_COLUMN\_USAGE Table

The [Information Schema](#)

KEY\_COLUMN\_USAGE  
table shows which key columns have constraints.

It contains the following columns:

Column	Description
CONSTRAINT_CATALOG	Always def
CONSTRAINT_SCHEMA	Database name of the constraint.
CONSTRAINT_NAME	Name of the constraint ( PRIMARY for the primary key).
TABLE_CATALOG	Always #def
TABLE_SCHEMA	Database name of the column constraint.
TABLE_NAME	Table name of the column constraint.
COLUMN_NAME	Column name of the constraint.
ORDINAL_POSITION	Position of the column within the constraint.
POSITION_IN_UNIQUE_CONSTRAINT	For foreign keys , the position in the unique constraint.
REFERENCED_TABLE_SCHEMA	For foreign keys, the referenced database name.
REFERENCED_TABLE_NAME	For foreign keys, the referenced table name.
REFERENCED_COLUMN_NAME	For foreign keys, the referenced column name.

## Example

```
SELECT * FROM information_schema.KEY_COLUMN_USAGE LIMIT 1 \G
*****
1. row ****
CONSTRAINT_CATALOG: def
CONSTRAINT_SCHEMA: my_website
CONSTRAINT_NAME: PRIMARY
TABLE_CATALOG: def
TABLE_SCHEMA: users
COLUMN_NAME: user_id
ORDINAL_POSITION: 1
POSITION_IN_UNIQUE_CONSTRAINT: NULL
REFERENCED_TABLE_SCHEMA: NULL
REFERENCED_TABLE_NAME: NULL
REFERENCED_COLUMN_NAME: NULL
```

## See Also

- [Finding Tables Without Primary Keys](#)

## 1.1.2.9.1.1.25 Information Schema KEYWORDS Table

The

KEYWORDS

table was added in [MariaDB 10.6.3](#).

## Description

The [Information Schema](#)

KEYWORDS

table contains the list of MariaDB keywords.

It contains a single column:

Column	Description
WORD	Keyword

The table is not a standard Information Schema table, and is a MariaDB extension.

## Example

```
SELECT * FROM INFORMATION_SCHEMA.KEYWORDS;
```

```
+-----+  
| WORD |  
+-----+  
| && |  
| <= |  
| <> |  
| != |  
| >= |  
| << |  
| >> |  
| <=> |  
| ACCESSIBLE |  
| ACCOUNT |  
| ACTION |  
| ADD |  
| ADMIN |  
| AFTER |  
| AGAINST |  
| AGGREGATE |  
| ALL |  
| ALGORITHM |  
| ALTER |  
| ALWAYS |  
| ANALYZE |  
| AND |  
| ANY |  
| AS |  
| ASC |  
| ASCII |  
| ASENSITIVE |  
| AT |  
| ATOMIC |  
| AUTHORS |  
| AUTO_INCREMENT |  
| AUTOEXTEND_SIZE |  
| AUTO |  
| AVG |  
| AVG_ROW_LENGTH |  
| BACKUP |  
| BEFORE |  
| BEGIN |  
| BETWEEN |  
| BIGINT |  
| BINARY |  
| BINLOG |  
| BIT |  
| BLOB |  
| BLOCK |  
| BODY |  
| BOOL |  
| BOOLEAN |  
|
```

BUIH	
BTREE	
BY	
BYTE	
CACHE	
CALL	
CASCADE	
CASCADED	
CASE	
CATALOG_NAME	
CHAIN	
CHANGE	
CHANGED	
CHAR	
CHARACTER	
CHARSET	
CHECK	
CHECKPOINT	
CHECKSUM	
CIPHER	
CLASS_ORIGIN	
CLIENT	
CLOB	
CLOSE	
COALESCE	
CODE	
COLLATE	
COLLATION	
COLUMN	
COLUMN_NAME	
COLUMNS	
COLUMN_ADD	
COLUMN_CHECK	
COLUMN_CREATE	
COLUMN_DELETE	
COLUMN_GET	
COMMENT	
COMMIT	
COMMITTED	
COMPACT	
COMPLETION	
COMPRESSED	
CONCURRENT	
CONDITION	
CONNECTION	
CONSISTENT	
CONSTRAINT	
CONSTRAINT_CATALOG	
CONSTRAINT_NAME	
CONSTRAINT_SCHEMA	
CONTAINS	
CONTEXT	
CONTINUE	
CONTRIBUTORS	
CONVERT	
CPU	
CREATE	
CROSS	
CUBE	
CURRENT	
CURRENT_DATE	
CURRENT_POS	
CURRENT_ROLE	
CURRENT_TIME	
CURRENT_TIMESTAMP	
CURRENT_USER	
CURSOR	
CURSOR_NAME	
CYCLE	
DATA	
DATABASE	
DATABASES	
DATAFILE	
DATE	
DATETIME	
DAY	
DAY_HOUR	
DAY_MICROSECOND	

```
| DAY_MINUTE
| DAY_SECOND
| DEALLOCATE
| DEC
| DECIMAL
| DECLARE
| DEFAULT
| DEFINER
| DELAYED
| DELAY_KEY_WRITE
| DELETE
| DELETE_DOMAIN_ID
| DESC
| DESCRIBE
| DES_KEY_FILE
| DETERMINISTIC
| DIAGNOSTICS
| DIRECTORY
| DISABLE
| DISCARD
| DISK
| DISTINCT
| DISTINCTROW
| DIV
| DO
| DOUBLE
| DO_DOMAIN_IDS
| DROP
| DUAL
| DUMPFILE
| DUPLICATE
| DYNAMIC
| EACH
| ELSE
| ELSEIF
| ELSIF
| EMPTY
| ENABLE
| ENCLOSED
| END
| ENDS
| ENGINE
| ENGINES
| ENUM
| ERROR
| ERRORS
| ESCAPE
| ESCAPED
| EVENT
| EVENTS
| EVERY
| EXAMINED
| EXCEPT
| EXCHANGE
| EXCLUDE
| EXECUTE
| EXCEPTION
| EXISTS
| EXIT
| EXPANSION
| EXPIRE
| EXPORT
| EXPLAIN
| EXTENDED
| EXTENT_SIZE
| FALSE
| FAST
| FAULTS
| FEDERATED
| FETCH
| FIELDS
| FILE
| FIRST
| FIXED
| FLOAT
| FLOAT4
| FLOAT8
| FLUSH
```

```
| FLUSH
| FOLLOWING
| FOLLOWS
| FOR
| FORCE
| FOREIGN
| FORMAT
| FOUND
| FROM
| FULL
| FULLTEXT
| FUNCTION
| GENERAL
| GENERATED
| GET_FORMAT
| GET
| GLOBAL
| GOTO
| GRANT
| GRANTS
| GROUP
| HANDLER
| HARD
| HASH
| HAVING
| HELP
| HIGH_PRIORITY
| HISTORY
| HOST
| HOSTS
| HOUR
| HOUR_MICROSECOND
| HOUR_MINUTE
| HOUR_SECOND
| ID
| IDENTIFIED
| IF
| IGNORE
| IGNORED
| IGNORE_DOMAIN_IDS
| IGNORE_SERVER_IDS
| IMMEDIATE
| IMPORT
| INTERSECT
| IN
| INCREMENT
| INDEX
| INDEXES
| INFILE
| INITIAL_SIZE
| INNER
| INOUT
| INSENSITIVE
| INSERT
| INSERT_METHOD
| INSTALL
| INT
| INT1
| INT2
| INT3
| INT4
| INT8
| INTEGER
| INTERVAL
| INVISIBLE
| INTO
| IO
| IO_THREAD
| IPC
| IS
| ISOLATION
| ISOPEN
| ISSUER
| ITERATE
| INVOKER
| JOIN
| JSON
| JSON_TABLE
```

KEY	
KEYS	
KEY_BLOCK_SIZE	
KILL	
LANGUAGE	
LAST	
LAST_VALUE	
LASTVAL	
LEADING	
LEAVE	
LEAVES	
LEFT	
LESS	
LEVEL	
LIKE	
LIMIT	
LINEAR	
LINES	
LIST	
LOAD	
LOCAL	
LOCALTIME	
LOCALTIMESTAMP	
LOCK	
LOCKED	
LOCKS	
LOGFILE	
LOGS	
LONG	
LONGBLOB	
LONGTEXT	
LOOP	
LOW_PRIORITY	
MASTER	
MASTER_CONNECT_RETRY	
MASTER_DELAY	
MASTER_GTID_POS	
MASTER_HOST	
MASTER_LOG_FILE	
MASTER_LOG_POS	
MASTER_PASSWORD	
MASTER_PORT	
MASTER_SERVER_ID	
MASTER_SSL	
MASTER_SSL_CA	
MASTER_SSL_CAPATH	
MASTER_SSL_CERT	
MASTER_SSL_CIPHER	
MASTER_SSL_CRL	
MASTER_SSL_CRLPATH	
MASTER_SSL_KEY	
MASTER_SSL_VERIFY_SERVER_CERT	
MASTER_USER	
MASTER_USE_GTID	
MASTER_HEARTBEAT_PERIOD	
MATCH	
MAX_CONNECTIONS_PER_HOUR	
MAX_QUERIES_PER_HOUR	
MAX_ROWS	
MAX_SIZE	
MAX_STATEMENT_TIME	
MAX_UPDATES_PER_HOUR	
MAX_USER_CONNECTIONS	
MAXVALUE	
MEDIUM	
MEDIUMBLOB	
MEDIUMINT	
MEDIUMTEXT	
MEMORY	
MERGE	
MESSAGE_TEXT	
MICROSECOND	
MIDDLEINT	
MIGRATE	
MINUS	
MINUTE	
MINUTE_MICROSECOND	
MINUTE_SECOND	

```
| MINVALUE
| MIN_ROWS
| MOD
| MODE
| MODIFIES
| MODIFY
| MONITOR
| MONTH
| MUTEX
| MYSQL
| MYSQL_ERRNO
| NAME
| NAMES
| NATIONAL
| NATURAL
| NCHAR
| NESTED
| NEVER
| NEW
| NEXT
| NEXTVAL
| NO
| NOMAXVALUE
| NOMINVALUE
| NOCACHE
| NOCYCLE
| NO_WAIT
| NOWAIT
| NODEGROUP
| NONE
| NOT
| NOTFOUND
| NO_WRITE_TO_BINLOG
| NULL
| NUMBER
| NUMERIC
| NVARCHAR
| OF
| OFFSET
| OLD_PASSWORD
| ON
| ONE
| ONLINE
| ONLY
| OPEN
| OPTIMIZE
| OPTIONS
| OPTION
| OPTIONALY
| OR
| ORDER
| ORDINALITY
| OTHERS
| OUT
| OUTER
| OUTFILE
| OVER
| OVERLAPS
| OWNER
| PACKAGE
| PACK_KEYS
| PAGE
| PAGE_CHECKSUM
| PARSER
| PARSE_VCOL_EXPR
| PATH
| PERIOD
| PARTIAL
| PARTITION
| PARTITIONING
| PARTITIONS
| PASSWORD
| PERSISTENT
| PHASE
| PLUGIN
| PLUGINS
| PORT
```

PORTION	
PRECEDES	
PRECEDING	
PRECISION	
PREPARE	
PRESERVE	
PREV	
PREVIOUS	
PRIMARY	
PRIVILEGES	
PROCEDURE	
PROCESS	
PROCESSLIST	
PROFILE	
PROFILES	
PROXY	
PURGE	
QUARTER	
QUERY	
QUICK	
RAISE	
RANGE	
RAW	
READ	
READ_ONLY	
READ_WRITE	
READS	
REAL	
REBUILD	
RECOVER	
RECURSIVE	
REDO_BUFFER_SIZE	
REDOFILE	
REDUNDANT	
REFERENCES	
REGEXP	
RELAY	
RELAYLOG	
RELAY_LOG_FILE	
RELAY_LOG_POS	
RELAY_THREAD	
RELEASE	
RELOAD	
REMOVE	
RENAME	
REORGANIZE	
REPAIR	
REPEATABLE	
REPLACE	
REPLAY	
REPLICA	
REPLICAS	
REPLICA_POS	
REPLICATION	
REPEAT	
REQUIRE	
RESET	
RESIGNAL	
RESTART	
RESTORE	
RESTRICT	
RESUME	
RETURNED_SQLSTATE	
RETURN	
RETURNING	
RETURNS	
REUSE	
REVERSE	
REVOKE	
RIGHT	
RLIKE	
ROLE	
ROLLBACK	
ROLLUP	
ROUTINE	
ROW	
ROWCOUNT	
ROWNUM	

| ROWS  
| ROWTYPE  
| ROW\_COUNT  
| ROW\_FORMAT  
| RTREE  
| SAVEPOINT  
| SCHEDULE  
| SCHEMA  
| SCHEMA\_NAME  
| SCHEMAS  
| SECOND  
| SECOND\_MICROSECOND  
| SECURITY  
| SELECT  
| SENSITIVE  
| SEPARATOR  
| SEQUENCE  
| SERIAL  
| SERIALIZABLE  
| SESSION  
| SERVER  
| SET  
| SETVAL  
| SHARE  
| SHOW  
| SHUTDOWN  
| SIGNAL  
| SIGNED  
| SIMPLE  
| SKIP  
| SLAVE  
| SLAVES  
| SLAVE\_POS  
| SLOW  
| SNAPSHOT  
| SMALLINT  
| SOCKET  
| SOFT  
| SOME  
| SONAME  
| SOUNDS  
| SOURCE  
| STAGE  
| STORED  
| SPATIAL  
| SPECIFIC  
| REF\_SYSTEM\_ID  
| SQL  
| SQLEXCEPTION  
| SQLSTATE  
| SQLWARNING  
| SQL\_BIG\_RESULT  
| SQL\_BUFFER\_RESULT  
| SQL\_CACHE  
| SQL\_CALC\_FOUND\_ROWS  
| SQL\_NO\_CACHE  
| SQL\_SMALL\_RESULT  
| SQL\_THREAD  
| SQL\_TSI\_SECOND  
| SQL\_TSI\_MINUTE  
| SQL\_TSI\_HOUR  
| SQL\_TSI\_DAY  
| SQL\_TSI\_WEEK  
| SQL\_TSI\_MONTH  
| SQL\_TSI\_QUARTER  
| SQL\_TSI\_YEAR  
| SSL  
| START  
| STARTING  
| STARTS  
| STATEMENT  
| STATS\_AUTO\_RECALC  
| STATS\_PERSISTENT  
| STATS\_SAMPLE\_PAGES  
| STATUS  
| STOP  
| STORAGE  
| CTDATACUT JOIN

```
| STRAIGHT_JOIN
| STRING
| SUBCLASS_ORIGIN
| SUBJECT
| SUBPARTITION
| SUBPARTITIONS
| SUPER
| SUSPEND
| SWAPS
| SWITCHES
| SYSDATE
| SYSTEM
| SYSTEM_TIME
| TABLE
| TABLE_NAME
| TABLES
| TABLESPACE
| TABLE_CHECKSUM
| TEMPORARY
| TEMPTABLE
| TERMINATED
| TEXT
| THAN
| THEN
| TIES
| TIME
| TIMESTAMP
| TIMESTAMPADD
| TIMESTAMPDIFF
| TINYBLOB
| TINYINT
| TINYTEXT
| TO
| TRAILING
| TRANSACTION
| TRANSACTIONAL
| THREADS
| TRIGGER
| TRIGGERS
| TRUE
| TRUNCATE
| TYPE
| TYPES
| UNBOUNDED
| UNCOMMITTED
| UNDEFINED
| UNDO_BUFFER_SIZE
| UNDOFILE
| UNDO
| UNICODE
| UNION
| UNIQUE
| UNKNOWN
| UNLOCK
| UNINSTALL
| UNSIGNED
| UNTIL
| UPDATE
| UPGRADE
| USAGE
| USE
| USER
| USER_RESOURCES
| USE_FRM
| USING
| UTC_DATE
| UTC_TIME
| UTC_TIMESTAMP
| VALUE
| VALUES
| VARBINARY
| VARCHAR
| VARCHARACTER
| VARCHAR2
| VARIABLES
| VARYING
| VIA
| VIEW
```

```

| VIRTUAL
| VISIBLE
| VERSIONING
| WAIT
| WARNINGS
| WEEK
| WEIGHT_STRING
| WHEN
| WHERE
| WHILE
| WINDOW
| WITH
| WITHIN
| WITHOUT
| WORK
| WRAPPER
| WRITE
| X509
| XOR
| XA
| XML
| YEAR
| YEAR_MONTH
| ZEROFILL
| ||
+-----+
694 rows in set (0.000 sec)

```

## See Also

- [Reserved Words](#)

## 1.1.2.9.1.1.26 Information Schema LOCALES Table

### Description

The [Information Schema](#)

#### LOCALES

table contains a list of all compiled-in locales. It is only available if the [LOCALES plugin](#) has been installed.

It contains the following columns:

Column	Description
ID	Row ID.
NAME	Locale name, for example en_GB
DESCRIPTION	Locale description, for example English - United Kingdom
MAX_MONTH_NAME_LENGTH	Numeric length of the longest month in the locale
MAX_DAY_NAME_LENGTH	Numeric length of the longest day name in the locale.
DECIMAL_POINT	Decimal point character (some locales use a comma).
THOUSAND_SEP	Thousand's character separator,
ERROR_MESSAGE_LANGUAGE	Error message language.

The table is not a standard Information Schema table, and is a MariaDB extension.

The

[SHOW LOCALES](#)

statement returns a subset of the information.

## Example

SELECT * FROM information_schema.LOCALES;						
ID	NAME	DESCRIPTION	MAX_MONTH_NAME_LENGTH	MAX_DAY_NAME_LENGTH	DECIMAL_POINT	THOUSAND_SEP
0	en_US	English - United States	9	9	.	,
1	en_GB	English - United Kingdom	9	9	.	,
2	ja_JP	Japanese - Japan	3	3	.	,
3	sv_SE	Swedish - Sweden	9	7	,	
4	de_DE	German - Germany	9	10	,	.
5	fr_FR	French - France	9	8	,	
6	ar_AE	Arabic - United Arab Emirates	6	8	.	,
7	ar_BH	Arabic - Bahrain	6	8	.	,
8	ar_JO	Arabic - Jordan	12	8	.	,
...						
106	no_NO	Norwegian - Norway	9	7	,	.
107	sv_FI	Swedish - Finland	9	7	,	
108	zh_HK	Chinese - Hong Kong SAR	3	3	.	,
109	el_GR	Greek - Greece	11	9	,	.

## 1.1.2.9.1.1.27 Information Schema METADATA\_LOCK\_INFO Table

The [Information Schema](#)

[METADATA\\_LOCK\\_INFO](#)

table is created by the [metadata\\_lock\\_info](#) plugin. It shows active [metadata locks](#) and user locks (the locks acquired with [GET\\_LOCK](#) ).

It has the following columns:

Column	Description
THREAD_ID	
LOCK_MODE	One of MDL_INTENTION_EXCLUSIVE , MDL_SHARED , MDL_SHARED_HIGH_PRIO , MDL_SHARED_READ , MDL_SHARED_READ_ONLY , MDL_SHARED_WRITE , MDL_SHARED_NO_WRITE , MDL_SHARED_NO_READ_WRITE , MDL_SHARED_UPGRADABLE or MDL_EXCLUSIVE .

LOCK_DURATION	One of MDL_STATEMENT ,MDL_TRANSACTION or MDL_EXPLICIT
LOCK_TYPE	One of Global read lock ,Schema metadata lock ,Table metadata lock ,Stored function metadata lock ,Stored procedure metadata lock ,Trigger metadata lock ,Event metadata lock ,Commit lock or User lock
TABLE_SCHEMA	
TABLE_NAME	

## "LOCK\_MODE" Descriptions

The

LOCK\_MODE

column can have the following values:

Value	Description
MDL_INTENTION_EXCLUSIVE	An intention exclusive metadata lock (IX). Used only for scoped locks. Owner of this type of lock can acquire upgradable exclusive locks on individual objects. Compatible with other IX locks, but is incompatible with scoped S and X locks. IX lock is taken in SCHEMA namespace when we intend to modify object metadata. Object may refer table, stored procedure, trigger, view/etc.
MDL_SHARED	A shared metadata lock (S). To be used in cases when we are interested in object metadata only and there is no intention to access object data (e.g. for stored routines or during preparing prepared statements). We also mis-use this type of lock for open HANDLERs, since lock acquired by this statement has to be compatible with lock acquired by LOCK TABLES ... WRITE statement, i.e. SNRW (We can't get by acquiring S lock at HANDLER ... OPEN time and upgrading it to SR lock for HANDLER ... READ as it doesn't solve problem with need to abort DML statements which wait on table level lock while having open HANDLER in the same connection). To avoid deadlock which may occur when SNRW lock is being upgraded to X lock for table on which there is an active S lock which is owned by thread which waits in its turn for table-level lock owned by thread performing upgrade we have to use <code>thr_abort_locks_for_thread()</code> facility in such situation. This problem does not arise for locks on stored routines as we don't use SNRW locks for them. It also does not arise when S locks are used during PREPARE calls as table-level locks are not acquired in this case. This lock is taken for global read lock, when caching a stored procedure in memory for the duration of the transaction and for tables used by prepared statements.
MDL_SHARED_HIGH_PRIO	A high priority shared metadata lock. Used for cases when there is no intention to access object data (i.e. data in the table). "High priority" means that, unlike other shared locks, it is granted ignoring pending requests for exclusive locks. Intended for use in cases when we only need to access metadata and not data, e.g. when filling an INFORMATION_SCHEMA table. Since SH lock is compatible with SNRW lock, the connection that holds SH lock lock should not try to acquire any kind of table-level or row-level lock, as this can lead to a deadlock. Moreover, after acquiring SH lock, the connection should not wait for any other resource, as it might cause starvation for X locks and a potential deadlock during upgrade of SNW or SNRW to X lock (e.g. if the upgrading connection holds the resource that is being waited for).
MDL_SHARED_READ	A shared metadata lock (SR) for cases when there is an intention to read data from table. A connection holding this kind of lock can read table metadata and read table data (after acquiring appropriate table and row-level locks). This means that one can only acquire TL_READ, TL_READ_NO_INSERT, and similar table-level locks on table if one holds SR MDL lock on it. To be used for tables in SELECTs, subqueries, and LOCK TABLE ... READ statements.

MDL_SHARED_WRITE	A shared metadata lock (SW) for cases when there is an intention to modify (and not just read) data in the table. A connection holding SW lock can read table metadata and modify or read table data (after acquiring appropriate table and row-level locks). To be used for tables to be modified by INSERT, UPDATE, DELETE statements, but not LOCK TABLE ... WRITE or DDL. Also taken by SELECT ... FOR UPDATE.
MDL_SHARED_UPGRADABLE	An upgradable shared metadata lock for cases when there is an intention to modify (and not just read) data in the table. Can be upgraded to MDL_SHARED_NO_WRITE and MDL_EXCLUSIVE. A connection holding SU lock can read table metadata and modify or read table data (after acquiring appropriate table and row-level locks). To be used for the first phase of ALTER TABLE.
MDL_SHARED_READ_ONLY	A shared metadata lock for cases when we need to read data from table and block all concurrent modifications to it (for both data and metadata). Used by LOCK TABLES READ statement.
MDL_SHARED_NO_WRITE	An upgradable shared metadata lock which blocks all attempts to update table data, allowing reads. A connection holding this kind of lock can read table metadata and read table data. Can be upgraded to X metadata lock. Note, that since this type of lock is not compatible with SNRW or SW lock types, acquiring appropriate engine-level locks for reading (TL_READ* for MyISAM, shared row locks in InnoDB) should be contention-free. To be used for the first phase of ALTER TABLE, when copying data between tables, to allow concurrent SELECTs from the table, but not UPDATEs.
MDL_SHARED_NO_READ_WRITE	An upgradable shared metadata lock which allows other connections to access table metadata, but not data. It blocks all attempts to read or update table data, while allowing INFORMATION_SCHEMA and SHOW queries. A connection holding this kind of lock can read table metadata modify and read table data. Can be upgraded to X metadata lock. To be used for LOCK TABLES WRITE statement. Not compatible with any other lock type except S and SH.
MDL_EXCLUSIVE	An exclusive metadata lock (X). A connection holding this lock can modify both table's metadata and data. No other type of metadata lock can be granted while this lock is held. To be used for CREATE/DROP/RENAME TABLE statements and for execution of certain phases of other DDL statements.

## Examples

User lock:

```
SELECT GET_LOCK('abc',1000);
+-----+
| GET_LOCK('abc',1000) |
+-----+
|           1 |
+-----+

SELECT * FROM information_schema.METADATA_LOCK_INFO;
+-----+-----+-----+-----+-----+
| THREAD_ID | LOCK_MODE          | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----+-----+-----+-----+-----+
|       61 | MDL_SHARED_NO_READ_WRITE | MDL_EXPLICIT   | User lock  | abc          |          |
+-----+-----+-----+-----+-----+
```

Table metadata lock:

```
START TRANSACTION;

INSERT INTO t VALUES (1,2);

SELECT * FROM information_schema.METADATA_LOCK_INFO \G
***** 1. row *****
THREAD_ID: 4
LOCK_MODE: MDL_SHARED_WRITE
LOCK_DURATION: MDL_TRANSACTION
LOCK_TYPE: Table metadata lock
TABLE_SCHEMA: test
TABLE_NAME: t
```

```
SELECT * FROM information_schema.METADATA_LOCK_INFO;
+-----+-----+-----+-----+-----+
| THREAD_ID | LOCK_MODE | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----+-----+-----+-----+-----+
| 31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT | Global read lock |  | |
| 31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT | Commit lock |  |
| 31 | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT | Schema metadata lock | dbname |
| 31 | MDL_SHARED_NO_READ_WRITE | MDL_EXPLICIT | Table metadata lock | dbname | exotics |
+-----+-----+-----+-----+-----+
```

## See also

- [metadata locks](#)
- [Performance Schema metadata\\_locks table](#)
- [GET\\_LOCK](#) ).

## 1.1.2.9.1.1.28 Information Schema MROONGA\_STATS Table

The [Information Schema](#)

MROONGA\_STATS

table only exists if the [Mroonga](#) storage engine is installed, and contains information about its activities.

Column	Description
VERSION	Mroonga version.
rows_written	Number of rows written into Mroonga tables.
rows_read	Number of rows read from all Mroonga tables.

This table always contains 1 row.

## 1.1.2.9.1.1.29 Information Schema OPTIMIZER\_TRACE Table

MariaDB starting with [10.4.3](#)

Optimizer Trace was introduced in [MariaDB 10.4.3](#).

## Description

The [Information Schema](#)

OPTIMIZER\_TRACE

table contains [Optimizer Trace](#) information.

It contains the following columns:

Column	Description
QUERY	Displays the query that was asked to be traced.
TRACE	A JSON document displaying the stats we collected when the query was run.
MISSING_BYTES_BEYOND_MAX_MEM_SIZE	For huge trace, where the trace is truncated due to the optimizer_trace_max_mem_size limit being reached, displays the bytes that are missing in the trace
INSUFFICIENT_PRIVILEGES	Set to 1 if the user running the trace does not have the privileges to see the trace.

Structure:

```

SHOW CREATE TABLE INFORMATION_SCHEMA.OPTIMIZER_TRACE \G
*****
1. row *****

Table: OPTIMIZER_TRACE
Create Table: CREATE TEMPORARY TABLE `OPTIMIZER_TRACE` (
  `QUERY` longtext NOT NULL DEFAULT '',
  `TRACE` longtext NOT NULL DEFAULT '',
  `MISSING_BYTES_BEYOND_MAX_MEM_SIZE` int(20) NOT NULL DEFAULT 0,
  `INSUFFICIENT_PRIVILEGES` tinyint(1) NOT NULL DEFAULT 0
) ENGINE=Aria DEFAULT CHARSET=utf8 PAGE_CHECKSUM=0

```

## 1.1.2.9.1.1.30 Information Schema PARAMETERS Table

The [Information Schema](#)

PARAMETERS

table stores information about [stored procedures](#) and [stored functions](#) parameters.

It contains the following columns:

Column	Description
SPECIFIC_CATALOG	Always def . . .
SPECIFIC_SCHEMA	Database name containing the stored routine parameter.
SPECIFIC_NAME	Stored routine name.
ORDINAL_POSITION	Ordinal position of the parameter, starting at 1 . . . 0 for a function RETURNS clause.
PARAMETER_MODE	One of IN , OUT , INOUT or NULL for RETURNS.
PARAMETER_NAME	Name of the parameter, or NULL for RETURNS.
DATA_TYPE	The column's <a href="#">data type</a> .
CHARACTER_MAXIMUM_LENGTH	Maximum length.
CHARACTER_OCTET_LENGTH	Same as the CHARACTER_MAXIMUM_LENGTH except for multi-byte <a href="#">character sets</a> .
NUMERIC_PRECISION	For numeric types, the precision (number of significant digits) for the column. NULL if not a numeric field.
NUMERIC_SCALE	For numeric types, the scale (significant digits to the right of the decimal point). NULL if not a numeric field.
DATETIME_PRECISION	Fractional-seconds precision, or NULL if not a <a href="#">time data type</a> .

CHARACTER_SET_NAME	Character set if a non-binary string data type , otherwise NULL
COLLATION_NAME	Collation if a non-binary string data type , otherwise NULL
DTD_IDENTIFIER	Description of the data type.
ROUTINE_TYPE	PROCEDURE or FUNCTION

Information from this table is similar to that found in the

`param_list`  
column in the `mysql.proc` table, and the output of the

`SHOW CREATE PROCEDURE`

and

`SHOW CREATE FUNCTION`

statements.

To obtain information about the routine itself, you can query the [Information Schema ROUTINES table](#).

## Example

```
SELECT * FROM information_schema.PARAMETERS
LIMIT 1 \G
*****
1. row *****
SPECIFIC_CATALOG: def
SPECIFIC_SCHEMA: accounts
SPECIFIC_NAME: user_counts
ORDINAL_POSITION: 1
PARAMETER_MODE: IN
PARAMETER_NAME: user_order
DATA_TYPE: varchar
CHARACTER_MAXIMUM_LENGTH: 255
CHARACTER_OCTET_LENGTH: 765
NUMERIC_PRECISION: NULL
NUMERIC_SCALE: NULL
DATETIME_PRECISION: NULL
CHARACTER_SET_NAME: utf8
COLLATION_NAME: utf8_general_ci
DTD_IDENTIFIER: varchar(255)
ROUTINE_TYPE: PROCEDURE
```

## 1.1.2.9.1.1.31 Information Schema PARTITIONS Table

The [Information Schema](#)

`PARTITIONS`

contains information about table partitions, with each record corresponding to a single partition or subpartition of a partitioned table. Each non-partitioned table also has a record in the

`PARTITIONS`

table, but most of the values are

`NULL`

It contains the following columns:

Column	Description
TABLE_CATALOG	Always def

TABLE_SCHEMA	Database name.
TABLE_NAME	Table name containing the partition.
PARTITION_NAME	Partition name.
SUBPARTITION_NAME	Subpartition name, or NULL if not a subpartition.
PARTITION_ORDINAL_POSITION	Order of the partition starting from 1.
SUBPARTITION_ORDINAL_POSITION	Order of the subpartition starting from 1.
PARTITION_METHOD	The partitioning type; one of RANGE ,, LIST ,, HASH ,, LINEAR HASH ,, KEY or LINEAR KEY ..
SUBPARTITION_METHOD	Subpartition type; one of HASH ,, LINEAR HASH ,, KEY or LINEAR KEY ,, NULL if not a subpartition.
PARTITION_EXPRESSION	Expression used to create the partition by the  <code>CREATE TABLE</code>  or  <code>ALTER TABLE</code>  statement.
SUBPARTITION_EXPRESSION	Expression used to create the subpartition by the  <code>CREATE TABLE</code>  or  <code>ALTER TABLE</code>  statement, or NULL if not a subpartition.

PARTITION_DESCRIPTION	For a RANGE partition, contains either MAXINTEGER or an integer, as set in the VALUES LESS THAN clause. For a LIST partition, contains a comma-separated list of integers, as set in the VALUES IN . NULL if another type of partition.
TABLE_ROWS	Number of rows in the table (may be an estimate for some storage engines).
AVG_ROW_LENGTH	Average row length, that is DATA_LENGTH divided by TABLE_ROWS
DATA_LENGTH	Total number of bytes stored in all rows of the partition.
MAX_DATA_LENGTH	Maximum bytes that could be stored in the partition.
INDEX_LENGTH	Size in bytes of the partition index file.
DATA_FREE	Unused bytes allocated to the partition.
CREATE_TIME	Time the partition was created
UPDATE_TIME	Time the partition was last modified.
CHECK_TIME	Time the partition was last checked, or NULL for storage engines that don't record this information.
CHECKSUM	Checksum value, or NULL if none.
PARTITION_COMMENT	Partition comment, truncated to 80 characters, or an empty string if no comment.
NODEGROUP	Node group, only used for MySQL Cluster, defaults to 0 .
TABLESPACE_NAME	Always default .

## 1.1.2.9.1.1.32 Information Schema PLUGINS Table

The [Information Schema](#)

PLUGINS

table contains information about [server plugins](#).

It contains the following columns:

Column	Description
--------	-------------

PLUGIN_NAME	Name of the plugin.
PLUGIN_VERSION	Version from the plugin's general type descriptor.
PLUGIN_STATUS	Plugin status, one of ACTIVE , INACTIVE , DISABLED or DELETED .
PLUGIN_TYPE	Plugin type; STORAGE ENGINE , INFORMATION_SCHEMA , AUTHENTICATION , REPLICATION , DAEMON or AUDIT .
PLUGIN_TYPE_VERSION	Version from the plugin's type-specific descriptor.
PLUGIN_LIBRARY	Plugin's shared object file name, located in the directory specified by the <code>plugin_dir</code> system variable, and used by the <code>INSTALL PLUGIN</code> and <code>UNINSTALL PLUGIN</code> statements. NULL if the plugin is compiled in and cannot be uninstalled.
PLUGIN_LIBRARY_VERSION	Version from the plugin's API interface.
PLUGIN_AUTHOR	Author of the plugin.
PLUGIN_DESCRIPTION	Description.
PLUGIN_LICENSE	Plugin's licence.
LOAD_OPTION	How the plugin was loaded; one of OFF , ON , FORCE or FORCE_PLUS_PERMANENT . See <a href="#">Installing Plugins</a> .

PLUGIN_MATURITY	Plugin's maturity level; one of Unknown ,
	Experimental ,
	Alpha ,
	Beta ,
	'Gamma , and Stable

PLUGIN_AUTH_VERSION	Plugin's version as determined by the plugin author. An example would be '0.99 beta 1'.
---------------------	---

It provides a superset of the information shown by the [SHOW PLUGINS](#) statement. For specific information about storage engines (a particular type of plugins), see the [information\\_schema.ENGINES](#) table and the [SHOW ENGINES](#) statement.

This table provides a subset of the Information Schema [information\\_schema.ALL\\_PLUGINS](#) table, which contains all available plugins, installed or not.

The table is not a standard Information Schema table, and is a MariaDB extension.

## Examples

The easiest way to get basic information on plugins is with [SHOW PLUGINS](#) :

```
SHOW PLUGINS;
```

Name	Status	Type	Library	License
binlog	ACTIVE	STORAGE ENGINE	NULL	GPL
mysql_native_password	ACTIVE	AUTHENTICATION	NULL	GPL
mysql_old_password	ACTIVE	AUTHENTICATION	NULL	GPL
MRG_MyISAM	ACTIVE	STORAGE ENGINE	NULL	GPL
MyISAM	ACTIVE	STORAGE ENGINE	NULL	GPL
CSV	ACTIVE	STORAGE ENGINE	NULL	GPL
MEMORY	ACTIVE	STORAGE ENGINE	NULL	GPL
FEDERATED	ACTIVE	STORAGE ENGINE	NULL	GPL
PERFORMANCE_SCHEMA	ACTIVE	STORAGE ENGINE	NULL	GPL
Aria	ACTIVE	STORAGE ENGINE	NULL	GPL
InnoDB	ACTIVE	STORAGE ENGINE	NULL	GPL
INNODB_TRX	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_LOCKS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_LOCK_WAITS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_CMP	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_CMP_RESET	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_CMPMEM	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_CMPMEM_RESET	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_BUFFER_PAGE	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_BUFFER_PAGE_LRU	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_BUFFER_POOL_STATS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_METRICS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_FT_DEFAULT_STOPWORD	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_FT_INSERTED	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_FT_DELETED	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_FT_BEING_DELETED	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_FT_CONFIG	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_FT_INDEX_CACHE	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_FT_INDEX_TABLE	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_SYS_TABLES	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_SYS_TABLESTATS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_SYS_INDEXES	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_SYS_COLUMNS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_SYS_FIELDS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_SYS_FOREIGN	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_SYS_FOREIGN_COLS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
SPHINX	ACTIVE	STORAGE ENGINE	NULL	GPL
ARCHIVE	ACTIVE	STORAGE ENGINE	NULL	GPL
BLACKHOLE	ACTIVE	STORAGE ENGINE	NULL	GPL
FEEDBACK	DISABLED	INFORMATION SCHEMA	NULL	GPL
partition	ACTIVE	STORAGE ENGINE	NULL	GPL
pam	ACTIVE	AUTHENTICATION	auth_pam.so	GPL

```
SELECT LOAD_OPTION
FROM INFORMATION_SCHEMA.PLUGINS
WHERE PLUGIN_NAME LIKE 'tokudb';
Empty set
```

The equivalent `SELECT` query would be:

```
SELECT PLUGIN_NAME, PLUGIN_STATUS,
PLUGIN_TYPE, PLUGIN_LIBRARY, PLUGIN_LICENSE
FROM INFORMATION_SCHEMA.PLUGINS;
```

Other `SELECT` queries can be used to see additional information. For example:

```

SELECT PLUGIN_NAME, PLUGIN_DESCRIPTION,
PLUGIN_MATURITY, PLUGIN_AUTH_VERSION
FROM INFORMATION_SCHEMA.PLUGINS
WHERE PLUGIN_TYPE='STORAGE ENGINE'
ORDER BY PLUGIN_MATURITY [G]

***** 1. row *****
PLUGIN_NAME: FEDERATED
PLUGIN_DESCRIPTION: FederatedX pluggable storage engine
    PLUGIN_MATURITY: Beta
PLUGIN_AUTH_VERSION: 2.1
***** 2. row *****
PLUGIN_NAME: Aria
PLUGIN_DESCRIPTION: Crash-safe tables with MyISAM heritage
    PLUGIN_MATURITY: Gamma
PLUGIN_AUTH_VERSION: 1.5
***** 3. row *****
PLUGIN_NAME: PERFORMANCE_SCHEMA
PLUGIN_DESCRIPTION: Performance Schema
    PLUGIN_MATURITY: Gamma
PLUGIN_AUTH_VERSION: 0.1
***** 4. row *****
PLUGIN_NAME: binlog
PLUGIN_DESCRIPTION: This is a pseudo storage engine to represent the binlog in a transaction
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
***** 5. row *****
PLUGIN_NAME: MEMORY
PLUGIN_DESCRIPTION: Hash based, stored in memory, useful for temporary tables
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
***** 6. row *****
PLUGIN_NAME: MyISAM
PLUGIN_DESCRIPTION: MyISAM storage engine
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
***** 7. row *****
PLUGIN_NAME: MRG_MyISAM
PLUGIN_DESCRIPTION: Collection of identical MyISAM tables
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
***** 8. row *****
PLUGIN_NAME: CSV
PLUGIN_DESCRIPTION: CSV storage engine
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
***** 9. row *****
PLUGIN_NAME: InnoDB
PLUGIN_DESCRIPTION: Supports transactions, row-level locking, and foreign keys
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.2.5
***** 10. row *****
PLUGIN_NAME: BLACKHOLE
PLUGIN_DESCRIPTION: /dev/null storage engine (anything you write to it disappears)
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
***** 11. row *****
PLUGIN_NAME: ARCHIVE
PLUGIN_DESCRIPTION: Archive storage engine
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0
***** 12. row *****
PLUGIN_NAME: partition
PLUGIN_DESCRIPTION: Partition Storage Engine Helper
    PLUGIN_MATURITY: Stable
PLUGIN_AUTH_VERSION: 1.0

```

Check if a given plugin is available:

```

SELECT LOAD_OPTION
FROM INFORMATION_SCHEMA.PLUGINS
WHERE PLUGIN_NAME LIKE 'tokudb';
Empty set

```

Show authentication plugins:

```

SELECT PLUGIN_NAME, LOAD_OPTION
FROM INFORMATION_SCHEMA.PLUGINS
WHERE PLUGIN_TYPE LIKE 'authentication' NG

***** 1. row *****
PLUGIN_NAME: mysql_native_password
LOAD_OPTION: FORCE
***** 2. row *****
PLUGIN_NAME: mysql_old_password
LOAD_OPTION: FORCE

```

## See Also

- [List of Plugins](#)
- [Plugin Overview](#)
- [SHOW PLUGINS](#)
- [INSTALL PLUGIN](#)
- [INSTALL SONAME](#)
- [UNINSTALL PLUGIN](#)
- [UNINSTALL SONAME](#)

## 1.1.2.9.1.1.33 Information Schema PROCESSLIST Table

### Contents

1. [Example](#)
2. [See Also](#)

The [Information Schema](#)

PROCESSLIST  
table contains information about running threads.

Similar information can also be returned with the

`SHOW [FULL] PROCESSLIST`

statement, or the

`mysqladmin processlist`

command.

It contains the following columns:

Column	Description
ID	Connection identifier.
USER	MariaDB User.
HOST	Connecting host.
DB	Default database, or NULL if none.
COMMAND	Type of command running, corresponding to the <code>Com_</code> <code>status variables</code> . See <a href="#">Thread Command Values</a> .
TIME	Seconds that the thread has spent on the current COMMAND so far.
STATE	Current state of the thread. See <a href="#">Thread States</a> .

INFO	Statement the thread is executing, or NULL if none.
TIME_MS	Time in milliseconds with microsecond precision that the thread has spent on the current COMMAND so far ( <a href="#">see more</a> ).
STAGE	The stage the process is currently in.
MAX_STAGE	The maximum number of stages.
PROGRESS	The progress of the process within the current stage (0-100%).
MEMORY_USED	Memory in bytes used by the thread.
EXAMINED_ROWS	Rows examined by the thread. Only updated by UPDATE, DELETE, and similar statements. For SELECT and other statements, the value remains zero.
QUERY_ID	Query ID.
INFO_BINARY	Binary data information
TID	Thread ID ( <a href="#">MDEV-6756</a> )

Note that as a difference to MySQL, in MariaDB the

`TIME`  
column (and also the  
`TIME_MS`  
column) are not affected by any setting of

`@TIMESTAMP`

. This means that it can be reliably used also for threads that change  
`@TIMESTAMP`  
(such as the `replication` SQL thread). See also [MySQL Bug #22047](#) .

As a consequence of this, the

`TIME`  
column of  
`SHOW FULL PROCESSLIST`  
and  
`INFORMATION_SCHEMA.PROCESSLIST`  
can not be used to determine if a slave is lagging behind. For this, use instead the  
`Seconds_Behind_Master`  
column in the output of [SHOW SLAVE STATUS](#) .

Note that the

`PROGRESS`  
field from the information schema, and the  
`PROGRESS`  
field from  
`SHOW PROCESSLIST`  
display different results.  
`SHOW PROCESSLIST`  
shows the total progress, while the information schema shows the progress for the current stage only.. To retrieve a similar "total" Progress value from

```
information_schema.PROCESSLIST
```

```
as the one from
```

```
SHOW PROCESSLIST
```

```
, use
```

```
SELECT CASE WHEN Max_Stage < 2 THEN Progress ELSE (Stage-1)/Max_Stage*100+Progress/Max_Stage END  
AS Progress FROM INFORMATION_SCHEMA.PROCESSLIST;
```

## Example

```
SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST\G  
***** 1. row *****  
    ID: 9  
  USER: msandbox  
  HOST: localhost  
    DB: NULL  
COMMAND: Query  
   TIME: 0  
  STATE: Filling schema table  
  INFO: SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST  
TIME_MS: 0.351  
 STAGE: 0  
MAX_STAGE: 0  
 PROGRESS: 0.000  
MEMORY_USED: 85392  
EXAMINED_ROWS: 0  
 QUERY_ID: 15  
INFO_BINARY: SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST  
      TID: 11838  
***** 2. row *****  
    ID: 5  
  USER: system user  
  HOST:  
    DB: NULL  
COMMAND: Daemon  
   TIME: 0  
  STATE: InnoDB shutdown handler  
  INFO: NULL  
TIME_MS: 0.000  
 STAGE: 0  
MAX_STAGE: 0  
 PROGRESS: 0.000  
MEMORY_USED: 24160  
EXAMINED_ROWS: 0  
 QUERY_ID: 0  
INFO_BINARY: NULL  
      TID: 3856  
...  
...
```

## See Also

- [TIME\\_MS column in Information Schema SHOW PROCESSLIST](#)

## 1.1.2.9.1.1.34 Information Schema PROFILING Table

The [Information Schema](#)

PROFILING

table contains information about statement resource usage. Profiling information is only recorded if the

[profiling](#)

session variable is set to 1.

It contains the following columns:

Column Name	Description
QUERY_ID	Query_ID.

SEQ	Sequence number showing the display order for rows with the same QUERY_ID
STATE	Profiling state.
DURATION	Time in seconds that the statement has been in the current state.
CPU_USER	User CPU usage in seconds.
CPU_SYSTEM	System CPU usage in seconds.
CONTEXT_VOLUNTARY	Number of voluntary context switches.
CONTEXT_INVOLUNTARY	Number of involuntary context switches.
BLOCK_OPS_IN	Number of block input operations.
BLOCK_OPS_OUT	Number of block output operations.
MESSAGES_SENT	Number of communications sent.
MESSAGES RECEIVED	Number of communications received.
PAGE_FAULTS_MAJOR	Number of major page faults.
PAGE_FAULTS_MINOR	Number of minor page faults.
SWAPS	Number of swaps.
SOURCE_FUNCTION	Function in the source code executed by the profiled state.
SOURCE_FILE	File in the source code executed by the profiled state.
SOURCE_LINE	Line in the source code executed by the profiled state.

It contains similar information to the

[SHOW PROFILE](#)

and

[SHOW PROFILES](#)

statements.

## 1.1.2.9.1.1.35 Information Schema QUERY\_CACHE\_INFO Table

### Description

The table is not a standard Information Schema table, and is a MariaDB extension.

The

QUERY\_CACHE\_INFO

table is created by the [QUERY\\_CACHE\\_INFO](#) plugin, and allows you to see the contents of the [query cache](#). It creates a table in the [information\\_schema](#) database that shows all queries that are in the cache. You must have the [PROCESS](#) privilege (see [GRANT](#)) to use this table.

It contains the following columns:

Column	Description
STATEMENT_SCHEMA	Database used when query was included
STATEMENT_TEXT	Query text
RESULT_BLOCKS_COUNT	Number of result blocks
RESULT_BLOCKS_SIZE	Size in bytes of result blocks
RESULT_BLOCKS_SIZE_USED	Size in bytes of used blocks
LIMIT	Added <a href="#">MariaDB 10.1.8</a> .
MAX_SORT_LENGTH	Added <a href="#">MariaDB 10.1.8</a> .
GROUP_CONCAT_MAX_LENGTH	Added <a href="#">MariaDB 10.1.8</a> .
CHARACTER_SET_CLIENT	Added <a href="#">MariaDB 10.1.8</a> .
CHARACTER_SET_RESULT	Added <a href="#">MariaDB 10.1.8</a> .
COLLATION	Added <a href="#">MariaDB 10.1.8</a> .
TIMEZONE	Added <a href="#">MariaDB 10.1.8</a> .
DEFAULT_WEEK_FORMAT	Added <a href="#">MariaDB 10.1.8</a> .
DIV_PRECISION_INCREMENT	Added <a href="#">MariaDB 10.1.8</a> .

SQL_MODE	Added <a href="#">MariaDB 10.1.8</a> .
LC_TIME_NAMES	Added <a href="#">MariaDB 10.1.8</a> .
CLIENT_LONG_FLAG	Added <a href="#">MariaDB 10.1.8</a> .
CLIENT_PROTOCOL_41	Added <a href="#">MariaDB 10.1.8</a> .
PROTOCOL_TYPE	Added <a href="#">MariaDB 10.1.8</a> .
MORE_RESULTS_EXISTS	Added <a href="#">MariaDB 10.1.8</a> .
IN_TRANS	Added <a href="#">MariaDB 10.1.8</a> .
AUTOCOMMIT	Added <a href="#">MariaDB 10.1.8</a> .
PACKET_NUMBER	Added <a href="#">MariaDB 10.1.8</a> .
HITS	Incremented each time the query cache is hit. Added <a href="#">MariaDB 10.3.2</a> .

For example:

```
SELECT * FROM information_schema.QUERY_CACHE_INFO;
+-----+-----+-----+-----+-----+
| STATEMENT_SCHEMA | STATEMENT_TEXT | RESULT_BLOCKS_COUNT | RESULT_BLOCKS_SIZE | RESULT_BLOCKS_SIZE_USED |
+-----+-----+-----+-----+-----+
...
| test           | SELECT * FROM a |          1 |        512 |         143 |
| test           | select * FROM a |          1 |        512 |         143 |
...
+-----+-----+-----+-----+
```

## 1.1.2.9.1.1.36 Information Schema QUERY\_RESPONSE\_TIME Table

### Description

The [Information Schema](#)

`QUERY_RESPONSE_TIME`

table contains information about queries that take a long time to execute . It is only available if the `QUERY_RESPONSE_TIME` plugin has been installed.

It contains the following columns:

Column	Description
TIME	Time interval
COUNT	Count of queries falling into the time interval

TOTAL	Total execution time of all queries for this interval
-------	---

See [QUERY\\_RESPONSE\\_TIME](#) plugin for a full description.

The table is not a standard Information Schema table, and is a MariaDB extension.

```
SHOW QUERY_RESPONSE_TIME
is available from MariaDB 10.1.1 as an alternative for retrieving the data.
```

## Example

```
SELECT * FROM information_schema.QUERY_RESPONSE_TIME;
+-----+-----+-----+
| TIME | COUNT | TOTAL |
+-----+-----+-----+
| 0.000001 | 0 | 0.000000 |
| 0.000010 | 17 | 0.000094 |
| 0.000100 | 4301 | 0.236555 |
| 0.001000 | 1499 | 0.824450 |
| 0.010000 | 14851 | 81.680502 |
| 0.100000 | 8066 | 443.635693 |
| 1.000000 | 0 | 0.000000 |
| 10.000000 | 0 | 0.000000 |
| 100.000000 | 1 | 55.937094 |
| 1000.000000 | 0 | 0.000000 |
| 10000.000000 | 0 | 0.000000 |
| 100000.000000 | 0 | 0.000000 |
| 1000000.000000 | 0 | 0.000000 |
| TOO LONG | 0 | TOO LONG |
+-----+-----+-----+
```

## 1.1.2.9.1.1.37 Information Schema REFERENTIAL\_CONSTRAINTS Table

The [Information Schema](#)

`REFERENTIAL_CONSTRAINTS`  
table contains information about [foreign keys](#). The single columns are listed in the

[KEY\\_COLUMN\\_USAGE](#)

table.

It has the following columns:

Column	Description
CONSTRAINT_CATALOG	Always def .
CONSTRAINT_SCHEMA	Database name, together with CONSTRAINT_NAME identifies the foreign key.
CONSTRAINT_NAME	Foreign key name, together with CONSTRAINT_SCHEMA identifies the foreign key.
UNIQUE_CONSTRAINT_CATALOG	Always def .
UNIQUE_CONSTRAINT_SCHEMA	Database name, together with UNIQUE_CONSTRAINT_NAME and REFERENCED_TABLE_NAME identifies the referenced key.

UNIQUE_CONSTRAINT_NAME	Referenced key name, together with UNIQUE_CONSTRAINT_SCHEMA and REFERENCED_TABLE_NAME identifies the referenced key.
MATCH_OPTION	Always NONE
UPDATE_RULE	The Update Rule; one of CASCADED , SET NULL , SET DEFAULT , RESTRICT , NO ACTION
DELETE_RULE	The Delete Rule; one of CASCADED , SET NULL , SET DEFAULT , RESTRICT , NO ACTION
TABLE_NAME	Table name from the TABLE_CONSTRAINTS table.
REFERENCED_TABLE_NAME	Referenced key table name, together with UNIQUE_CONSTRAINT_SCHEMA and UNIQUE_CONSTRAINT_NAME identifies the referenced key.

## 1.1.2.9.1.1.38 Information Schema ROUTINES Table

The [Information Schema](#)

[ROUTINES](#)  
table stores information about [stored procedures](#) and [stored functions](#).

It contains the following columns:

Column	Description
SPECIFIC_NAME	
ROUTINE_CATALOG	Always def
ROUTINE_SCHEMA	Database name associated with the routine.
ROUTINE_NAME	Name of the routine.
ROUTINE_TYPE	Whether the routine is a PROCEDURE or a FUNCTION

DATA_TYPE	The return value's <a href="#">data type</a> (for stored functions).
CHARACTER_MAXIMUM_LENGTH	Maximum length.
CHARACTER_OCTET_LENGTH	Same as the CHARACTER_MAXIMUM_LENGTH except for multi-byte <a href="#">character sets</a> .
NUMERIC_PRECISION	For numeric types, the precision (number of significant digits) for the column. NULL if not a numeric field.
NUMERIC_SCALE	For numeric types, the scale (significant digits to the right of the decimal point). NULL if not a numeric field.
DATETIME_PRECISION	Fractional-seconds precision, or NULL if not a <a href="#">time data type</a> .
CHARACTER_SET_NAME	<a href="#">Character set</a> if a non-binary <a href="#">string data type</a> , otherwise NULL
COLLATION_NAME	<a href="#">Collation</a> if a non-binary <a href="#">string data type</a> , otherwise NULL.
DATA_TYPE	The column's <a href="#">data type</a> .
ROUTINE_BODY	Always SQL
ROUTINE_DEFINITION	Definition of the routine.
EXTERNAL_NAME	Always NULL
EXTERNAL_LANGUAGE	Always SQL
PARAMETER_STYLE	Always SQL
IS_DETERMINISTIC	Whether the routine is deterministic (can produce only one result for a given list of parameters) or not.
SQL_DATA_ACCESS	One of READS SQL DATA ,MODIFIES SQL DATA ,CONTAINS SQL , OR NO SQL
SQL_PATH	Always NULL

SECURITY_TYPE	INVOKER or DEFINER . Indicates which user's privileges apply to this routine.
CREATED	Date and time the routine was created.
LAST_ALTERED	Date and time the routine was last changed.
SQL_MODE	The  <a href="#">SQL_MODE</a>  at the time the routine was created.
ROUTINE_COMMENT	Comment associated with the routine.
DEFINER	If the  SECURITY_TYPE is DEFINER , this value indicates which user defined this routine.
CHARACTER_SET_CLIENT	The <a href="#">character set</a> used by the client that created the routine.
COLLATION_CONNECTION	The <a href="#">collation</a> (and character set) used by the connection that created the routine.
DATABASE_COLLATION	The default <a href="#">collation</a> (and character set) for the database, at the time the routine was created.

It provides information similar to, but more complete, than the

[SHOW PROCEDURE STATUS](#)

and

[SHOW FUNCTION STATUS](#)

statements.

For information about the parameters accepted by the routine, you can query the

[information\\_schema.PARAMETERS](#)

table.

## See also

- [Stored Function Overview](#)
- [Stored Procedure Overview](#)

## 1.1.2.9.1.1.39 Information Schema SCHEMA\_PRIVILEGES Table

The [Information Schema](#)

[SCHEMA\\_PRIVILEGES](#)

table contains information about [database privileges](#).

It contains the following columns:

Column	Description
--------	-------------

GRANTEE	Account granted the privilege in the format user_name@host_name
TABLE_CATALOG	Always def
TABLE_SCHEMA	Database name.
PRIVILEGE_TYPE	The granted privilege.
IS_GRANTABLE	Whether the privilege can be granted.

The same information in a different format can be found in the

[mysql.db](#)

table.

## 1.1.2.9.1.1.40 Information Schema SCHEMATA Table

The [Information Schema](#)

SCHEMATA

table stores information about databases on the server.

It contains the following columns:

Column	Description
CATALOG_NAME	Always def
SCHEMA_NAME	Database name.
DEFAULT_CHARACTER_SET_NAME	Default <a href="#">character set</a> for the database.
DEFAULT_COLLATION_NAME	Default <a href="#">collation</a> .
SQL_PATH	Always NULL
SCHEMA_COMMENT	Database comment. From <a href="#">MariaDB 10.5.0</a> .

## Example

```

SELECT * FROM INFORMATION_SCHEMA.SCHEMATA\G
*****
1. row *****
  CATALOG_NAME: def
    SCHEMA_NAME: information_schema
DEFAULT_CHARACTER_SET_NAME: utf8
  DEFAULT_COLLATION_NAME: utf8_general_ci
    SQL_PATH: NULL
*****
2. row *****
  CATALOG_NAME: def
    SCHEMA_NAME: mysql
DEFAULT_CHARACTER_SET_NAME: latin1
  DEFAULT_COLLATION_NAME: latin1_swedish_ci
    SQL_PATH: NULL
*****
3. row *****
  CATALOG_NAME: def
    SCHEMA_NAME: performance_schema
DEFAULT_CHARACTER_SET_NAME: utf8
  DEFAULT_COLLATION_NAME: utf8_general_ci
    SQL_PATH: NULL
*****
4. row *****
  CATALOG_NAME: def
    SCHEMA_NAME: test
DEFAULT_CHARACTER_SET_NAME: latin1
  DEFAULT_COLLATION_NAME: latin1_swedish_ci
    SQL_PATH: NULL
...

```

From MariaDB 10.5.0 :

```

SELECT * FROM INFORMATION_SCHEMA.SCHEMATA\G
...
*****
2. row *****
  CATALOG_NAME: def
    SCHEMA_NAME: presentations
DEFAULT_CHARACTER_SET_NAME: latin1
  DEFAULT_COLLATION_NAME: latin1_swedish_ci
    SQL_PATH: NULL
    SCHEMA_COMMENT: Presentations for conferences
...

```

## See Also

- [CREATE DATABASE](#)
- [ALTER DATABASE](#)
- [DROP DATABASE](#)
- [SHOW CREATE DATABASE](#)
- [SHOW DATABASES](#)
- [Character Sets and Collations](#)

## 1.1.2.9.1.1.41 Information Schema SPATIAL\_REF\_SYS Table

MariaDB starting with [10.1.2](#)

The

SPATIAL\_REF\_SYS  
table was introduced in [MariaDB 10.1.2](#)

## Description

The [Information Schema](#)

SPATIAL\_REF\_SYS  
table stores information on each spatial reference system used in the database.

It contains the following columns:

Column	Type	Null	Description
--------	------	------	-------------

SRID	smallint(5)	NO	An integer value that uniquely identifies each Spatial Reference System within a database.
AUTH_NAME	varchar(512)	NO	The name of the standard or standards body that is being cited for this reference system.
AUTH_SRID	smallint(5)	NO	The numeric ID of the coordinate system in the above authority's catalog.
SRTEXT	varchar(2048)	NO	The <a href="#">Well-known Text Representation</a> of the Spatial Reference System.

Note: See [MDEV-7540](#).

## See also

- [information\\_schema.GEOMETRY\\_COLUMNS](#) table.

## 1.1.2.9.1.1.42 Information Schema SPIDER\_ALLOC\_MEM Table

The [Information Schema](#)

`SPIDER_ALLOC_MEM`

table is installed along with the [Spider](#) storage engine. It shows information about Spider's memory usage.

It contains the following columns:

Column	Description
ID	
FUNC_NAME	
FILE_NAME	
LINE_NO	
TOTAL_ALLOC_MEM	
CURRENT_ALLOC_MEM	
ALLOC_MEM_COUNT	
FREE_MEM_COUNT	

## 1.1.2.9.1.1.43 Information Schema

## SPIDER\_WRAPPER\_PROTOCOLS Table

MariaDB starting with [10.5.4](#)

The [Information Schema](#)

SPIDER\_WRAPPER\_PROTOCOLS  
table is installed along with the [Spider](#) storage engine from [MariaDB 10.5.4](#).

It contains the following columns:

Column	Type	Description
WRAPPER_NAME	varchar(64)	
WRAPPER_VERSION	varchar(20)	
WRAPPER_DESCRIPTION	longtext	
WRAPPER_MATURITY	varchar(12)	

## 1.1.2.9.1.1.44 Information Schema SQL\_FUNCTIONS Table

MariaDB starting with [10.6.3](#)

The

[SQL\\_FUNCTIONS](#)  
table was added in [MariaDB 10.6.3](#).

### Description

The [Information Schema](#)

[SQL\\_FUNCTIONS](#)  
table contains the list of [MariaDB functions](#).

It contains a single column:

Column	Description
FUNCTION	Function name

The table is not a standard Information Schema table, and is a MariaDB extension.

### Example

```
SELECT * FROM INFORMATION_SCHEMA.SQL_FUNCTIONS;
+-----+
| FUNCTION |
+-----+
| ADDDATE |
| ADD_MONTHS |
| BIT_AND |
| BIT_OR |
| BIT_XOR |
| CAST |
| COUNT |
| CUME_DIST |
| CURDATE |
| CURTIME |
| DATE_ADD |
```

```
| DATE_SUB  
| DATE_FORMAT  
| DECODE  
| DENSE_RANK  
| EXTRACT  
| FIRST_VALUE  
| GROUP_CONCAT  
| JSON_ARRAYAGG  
| JSON_OBJECTAGG  
| LAG  
| LEAD  
| MAX  
| MEDIAN  
| MID  
| MIN  
| NOW  
| NTH_VALUE  
| NTILE  
| POSITION  
| PERCENT_RANK  
| PERCENTILE_CONT  
| PERCENTILE_DISC  
| RANK  
| ROW_NUMBER  
| SESSION_USER  
| STD  
| STDDEV  
| STDDEV_POP  
| STDDEV_SAMP  
| SUBDATE  
| SUBSTR  
| SUBSTRING  
| SUM  
| SYSTEM_USER  
| TRIM  
| TRIM_ORACLE  
| VARIANCE  
| VAR_POP  
| VAR_SAMP  
| ABS  
| ACOS  
| ADDTIME  
| AES_DECRYPT  
| AES_ENCRYPT  
| ASIN  
| ATAN  
| ATAN2  
| BENCHMARK  
| BIN  
| BINLOG_GTID_POS  
| BIT_COUNT  
| BIT_LENGTH  
| CEIL  
| CEILING  
| CHARACTER_LENGTH  
| CHAR_LENGTH  
| CHR  
| COERCIBILITY  
| COLUMN_CHECK  
| COLUMN_EXISTS  
| COLUMN_LIST  
| COLUMN_JSON  
| COMPRESS  
| CONCAT  
| CONCAT_OPERATOR_ORACLE  
| CONCAT_WS  
| CONNECTION_ID  
| CONV  
| CONVERT_TZ  
| COS  
| COT  
| CRC32  
| DATEDIFF  
| DAYNAME  
| DAYOFMONTH  
| DAYOFWEEK  
| DAYOFYEAR  
| DEGREES
```

```
| DECODES  
| DECODE_HISTOGRAM  
| DECODE_ORACLE  
| DES_DECRYPT  
| DES_ENCRYPT  
| ELT  
| ENCODE  
| ENCRYPT  
| EXP  
| EXPORT_SET  
| EXTRACTVALUE  
| FIELD  
| FIND_IN_SET  
| FLOOR  
| FORMAT  
| FOUND_ROWS  
| FROM_BASE64  
| FROM_DAYS  
| FROM_UNIXTIME  
| GET_LOCK  
| GREATEST  
| HEX  
| IFNULL  
| INSTR  
| ISNULL  
| IS_FREE_LOCK  
| IS_USED_LOCK  
| JSON_ARRAY  
| JSON_ARRAY_APPEND  
| JSON_ARRAY_INSERT  
| JSON_COMPACT  
| JSON_CONTAINS  
| JSON_CONTAINS_PATH  
| JSON_DEPTH  
| JSON_DETAILED  
| JSON_EXISTS  
| JSON_EXTRACT  
| JSON_INSERT  
| JSON_KEYS  
| JSON_LENGTH  
| JSON_LOOSE  
| JSON_MERGE  
| JSON_MERGE_PATCH  
| JSON_MERGE_PRESERVE  
| JSON_QUERY  
| JSON_QUOTE  
| JSON_OBJECT  
| JSON_REMOVE  
| JSON_REPLACE  
| JSON_SET  
| JSON_SEARCH  
| JSON_TYPE  
| JSON_UNQUOTE  
| JSON_VALID  
| JSON_VALUE  
| LAST_DAY  
| LAST_INSERT_ID  
| LCASE  
| LEAST  
| LENGTH  
| LENGTHB  
| LN  
| LOAD_FILE  
| LOCATE  
| LOG  
| LOG10  
| LOG2  
| LOWER  
| LPAD  
| LPAD_ORACLE  
| LTRIM  
| LTRIM_ORACLE  
| MAKEDATE  
| MAKETIME  
| MAKE_SET  
| MASTER_GTID_WAIT  
| MASTER_POS_WAIT  
| MD5
```

```
| MONTHNAME  
| NAME_CONST  
| NVL  
| NVL2  
| NULLIF  
| OCT  
| OCTET_LENGTH  
| ORD  
| PERIOD_ADD  
| PERIOD_DIFF  
| PI  
| POW  
| POWER  
| QUOTE  
| REGEXP_INSTR  
| REGEXP_REPLACE  
| REGEXP_SUBSTR  
| RADIANS  
| RAND  
| RELEASE_ALL_LOCKS  
| RELEASE_LOCK  
| REPLACE_ORACLE  
| REVERSE  
| ROUND  
| RPAD  
| RPAD_ORACLE  
| RTRIM  
| RTRIM_ORACLE  
| SEC_TO_TIME  
| SHA  
| SHA1  
| SHA2  
| SIGN  
| SIN  
| SLEEP  
| SOUNDEX  
| SPACE  
| SQRT  
| STRCMP  
| STR_TO_DATE  
| SUBSTR_ORACLE  
| SUBSTRING_INDEX  
| SUBTIME  
| SYS_GUID  
| TAN  
| TIMEDIFF  
| TIME_FORMAT  
| TIME_TO_SEC  
| TO_BASE64  
| TO_CHAR  
| TO_DAYS  
| TO_SECONDS  
| UCASE  
| UNCOMPRESS  
| UNCOMPRESSED_LENGTH  
| UNHEX  
| UNIX_TIMESTAMP  
| UPDATEXML  
| UPPER  
| UUID  
| UUID_SHORT  
| VERSION  
| WEEKDAY  
| WEEKOFYEAR  
| WSREP_LAST_WRITTEN_GTID  
| WSREP_LAST_SEEN_GTID  
| WSREP_SYNC_WAIT_UPTO_GTID  
| YEARWEEK  
+-----+  
234 rows in set (0.001 sec)
```

## See Also

- [Reserved Words](#)

## 1.1.2.9.1.1.45 Information Schema STATISTICS Table

The [Information Schema](#)

STATISTICS

table provides information about table indexes.

It contains the following columns:

Column	Description
TABLE_CATALOG	Always def
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
NON_UNIQUE	1 if the index can have duplicates, 0 if not.
INDEX_SCHEMA	Database name.
INDEX_NAME	Index name. The primary key is always named PRIMARY
SEQ_IN_INDEX	The column sequence number, starting at 1.
COLUMN_NAME	Column name.
COLLATION	A for sorted in ascending order, or NULL for unsorted.
CARDINALITY	Estimate of the number of unique values stored in the index based on statistics stored as integers. Higher cardinalities usually mean a greater chance of the index being used in a join. Updated by the <a href="#">ANALYZE TABLE</a> statement or <a href="#">myisamchk -a</a> .
SUB_PART	NULL if the whole column is indexed, or the number of indexed characters if partly indexed.
PACKED	NULL if not packed, otherwise how the index is packed.
NULLABLE	YES if the column may contain NULLs, empty string if not.

INDEX_TYPE	Index type, one of BTREE , RTREE , HASH or FULLTEXT . See <a href="#">Storage Engine Index Types</a> .
COMMENT	Index comments from the <code>CREATE INDEX</code> statement.
IGNORED	Whether or not an index will be ignored by the optimizer. See <a href="#">Ignored Indexes</a> . From <a href="#">MariaDB 10.6.0</a> .

The

`SHOW INDEX`

statement produces similar output.

## Example

```
SELECT * FROM INFORMATION_SCHEMA.STATISTICS\G
...
***** 85. row *****
TABLE_CATALOG: def
TABLE_SCHEMA: test
TABLE_NAME: table1
NON_UNIQUE: 1
INDEX_SCHEMA: test
INDEX_NAME: col2
SEQ_IN_INDEX: 1
COLUMN_NAME: col2
COLLATION: A
CARDINALITY: 6
SUB_PART: NULL
PACKED: NULL
NULLABLE:
INDEX_TYPE: BTREE
COMMENT:
INDEX_COMMENT:
...
```

## 1.1.2.9.1.1.46 Information Schema SYSTEM\_VARIABLES Table

MariaDB starting with [10.1.1](#)

The

`information_schema.SYSTEM_VARIABLES`  
table was introduced in [MariaDB 10.1.1](#)

The [Information Schema](#)

`SYSTEM_VARIABLES`  
table shows current values and various metadata of all [system variables](#).

It contains the following columns:

Column	Description
VARIABLE_NAME	System variable name.

SESSION_VALUE	Session value of the variable or NULL if the variable only has a global scope.
GLOBAL_VALUE	Global value of the variable or NULL if the variable only has a session scope.
GLOBAL_VALUE_ORIGIN	How the global value was set — a compile-time default, auto-configured by the server, configuration file (or a command line), with the SQL statement.
DEFAULT_VALUE	Compile-time default value of the variable.
VARIABLE_SCOPE	Global, session, or session-only.
VARIABLE_TYPE	Data type of the variable value.
VARIABLE_COMMENT	Help text, usually shown in <pre>mysqld --help --verbose</pre> .
NUMERIC_MIN_VALUE	For numeric variables — minimal allowed value.
NUMERIC_MAX_VALUE	For numeric variables — maximal allowed value.
NUMERIC_BLOCK_SIZE	For numeric variables — a valid value must be a multiple of the "block size".
ENUM_VALUE_LIST	For <pre>ENUM , SET , and FLAGSET</pre> variables — the list of recognized values.
READ_ONLY	Whether a variable can be set with the SQL statement. Note that many "read only" variables can still be set on the command line.
COMMAND_LINE_ARGUMENT	Whether an argument is required when setting the variable on the command line. <pre>NULL when a variable can not be set on the command line.</pre>
GLOBAL_VALUE_PATH	Which config file the variable got its value from. <pre>NULL if not set in any config file. Added in MariaDB 10.5.0 .</pre>

## Example

```

SELECT * FROM information_schema.SYSTEM_VARIABLES
WHERE VARIABLE_NAME='JOIN_BUFFER_SIZE'\G
***** 1. row *****
VARIABLE_NAME: JOIN_BUFFER_SIZE
SESSION_VALUE: 131072
GLOBAL_VALUE: 131072
GLOBAL_VALUE_ORIGIN: COMPILE-TIME
DEFAULT_VALUE: 131072
VARIABLE_SCOPE: SESSION
VARIABLE_TYPE: BIGINT UNSIGNED
VARIABLE_COMMENT: The size of the buffer that is used for joins
NUMERIC_MIN_VALUE: 128
NUMERIC_MAX_VALUE: 18446744073709551615
NUMERIC_BLOCK_SIZE: 128
ENUM_VALUE_LIST: NULL
READ_ONLY: NO
COMMAND_LINE_ARGUMENT: REQUIRED

```

## 1.1.2.9.1.1.47 Information Schema TABLE\_CONSTRAINTS Table

The [Information Schema](#)

TABLE\_CONSTRAINTS

table contains information about tables that have [constraints](#).

It has the following columns:

Column	Description
CONSTRAINT_CATALOG	Always def
CONSTRAINT_SCHEMA	Database name containing the constraint.
CONSTRAINT_NAME	Constraint name.
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
CONSTRAINT_TYPE	Type of constraint; one of UNIQUE , PRIMARY KEY , FOREIGN KEY or CHECK

The [REFERENTIAL\\_CONSTRAINTS](#) table has more information about foreign keys.

## 1.1.2.9.1.1.48 Information Schema TABLE\_PRIVILEGES Table

The [Information Schema](#)

TABLE\_PRIVILEGES

table contains table privilege information derived from the

[mysql.tables\\_priv](#)

grant table.

It has the following columns:

Column	Description
GRANTEE	In the format user_name@host_name
TABLE_CATALOG	Always def
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
PRIVILEGE_TYPE	One of SELECT ,, INSERT ,, UPDATE ,, REFERENCES ,, ALTER ,, INDEX ,, DROP or CREATE VIEW
IS_GRANTABLE	Whether the user has the  <a href="#">GRANT OPTION</a>  for this privilege.

Similar information can be accessed with the

[SHOW GRANTS](#)

statement. See the

[GRANT](#)

article for more about privileges.

For a description of the privileges that are shown in this table, see [table privileges](#).

## 1.1.2.9.1.1.49 Information Schema TABLE\_STATISTICS Table

The [Information Schema](#)

`TABLE_STATISTICS`

table shows statistics on table usage.

This is part of the [User Statistics](#) feature, which is not enabled by default.

It contains the following columns:

Field	Type	Notes
TABLE_SCHEMA	varchar(192)	The schema (database) name.

TABLE_NAME	varchar(192)	The table name.
ROWS_READ	int(21)	The number of rows read from the table.
ROWS_CHANGED	int(21)	The number of rows changed in the table.
ROWS_CHANGED_X_INDEXES	int(21)	The number of rows changed in the table, multiplied by the number of indexes changed.

## Example

```
SELECT * FROM INFORMATION_SCHEMA.TABLE_STATISTICS WHERE TABLE_NAME='user';
+-----+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME | ROWS_READ | ROWS_CHANGED | ROWS_CHANGED_X_INDEXES |
+-----+-----+-----+-----+
| mysql        | user       |      5 |          2 |                  2 |
+-----+-----+-----+-----+
```

## 1.1.2.9.1.1.50 Information Schema TABLES Table

### Contents

1. [Examples](#)
  1. [View Tables in Order of Size](#)
  2. [See Also](#)

The [Information Schema](#) table shows information about the various non-

TEMPORARY  
tables (except tables from the  
Information Schema  
database) and [views](#) on the server.

It contains the following columns:

Column	Description
TABLE_CATALOG	Always def .
TABLE_SCHEMA	Database name.
TABLE_NAME	Table name.
TABLE_TYPE	One of BASE TABLE for a regular table, VIEW for a <a href="#">view</a> , SYSTEM VIEW for <a href="#">Information Schema</a> tables, SYSTEM VERSIONED for <a href="#">system-versioned tables</a> or SEQUENCE for <a href="#">sequences</a> .

ENGINE	<a href="#">Storage Engine</a> .
VERSION	Version number from the table's .frm file
ROW_FORMAT	Row format (see <a href="#">InnoDB</a> , <a href="#">Aria</a> and <a href="#">MyISAM</a> row formats).
TABLE_ROWS	Number of rows in the table. Some engines, such as <a href="#">XtraDB</a> and <a href="#">InnoDB</a> may store an estimate.
AVG_ROW_LENGTH	Average row length in the table.
DATA_LENGTH	For <a href="#">InnoDB/XtraDB</a> , the index size, in pages, multiplied by the page size. For <a href="#">Aria</a> and <a href="#">MyISAM</a> , length of the data file, in bytes. For <a href="#">MEMORY</a> , the approximate allocated memory.
MAX_DATA_LENGTH	Maximum length of the data file, ie the total number of bytes that could be stored in the table. Not used in <a href="#">XtraDB</a> and <a href="#">InnoDB</a> .
INDEX_LENGTH	Length of the index file.
DATA_FREE	Bytes allocated but unused. For <a href="#">InnoDB</a> tables in a shared tablespace, the free space of the shared tablespace with small safety margin. An estimate in the case of partitioned tables - see the <a href="#">PARTITIONS</a> table.
AUTO_INCREMENT	Next <a href="#">AUTO_INCREMENT</a> value.
CREATE_TIME	Time the table was created.
UPDATE_TIME	Time the table was last updated. On Windows, the timestamp is not updated on update, so MyISAM values will be inaccurate. In <a href="#">InnoDB</a> , if shared tablespaces are used, will be NULL, while buffering can also delay the update, so the value will differ from the actual time of the last <a href="#">UPDATE</a> , <a href="#">INSERT</a> or <a href="#">DELETE</a>
CHECK_TIME	Time the table was last checked. Not kept by all storage engines, in which case will be NULL
TABLE_COLLATION	<a href="#">Character set and collation</a> .

CHECKSUM	Live checksum value, if any.
CREATE_OPTIONS	Extra CREATE TABLE options.
TABLE_COMMENT	Table comment provided when MariaDB created the table.
MAX_INDEX_LENGTH	Maximum index length (supported by MyISAM and Aria tables). Added in <a href="#">MariaDB 10.3.5</a> .
TEMPORARY	Placeholder to signal that a table is a temporary table. Currently always "N", except "Y" for generated information_schema tables and NULL for <a href="#">views</a> . Added in <a href="#">MariaDB 10.3.5</a> .

Although the table is standard in the Information Schema, all but

```
TABLE_CATALOG
,
TABLE_SCHEMA
,
TABLE_NAME
,
TABLE_TYPE
,
ENGINE
and
VERSION
are MySQL and MariaDB extensions.
```

[SHOW TABLES](#) lists all tables in a database.

## Examples

From [MariaDB 10.3.5](#):

```

SELECT * FROM information_schema.tables WHERE table_schema='test'\G
***** 1. row *****
TABLE_CATALOG: def
TABLE_SCHEMA: test
TABLE_NAME: xx5
TABLE_TYPE: BASE TABLE
ENGINE: InnoDB
VERSION: 10
ROW_FORMAT: Dynamic
TABLE_ROWS: 0
AVG_ROW_LENGTH: 0
DATA_LENGTH: 16384
MAX_DATA_LENGTH: 0
INDEX_LENGTH: 0
DATA_FREE: 0
AUTO_INCREMENT: NULL
CREATE_TIME: 2020-11-18 15:57:10
UPDATE_TIME: NULL
CHECK_TIME: NULL
TABLE_COLLATION: latin1_swedish_ci
CHECKSUM: NULL
CREATE_OPTIONS:
TABLE_COMMENT:
MAX_INDEX_LENGTH: 0
TEMPORARY: N
***** 2. row *****
TABLE_CATALOG: def
TABLE_SCHEMA: test
TABLE_NAME: xx4
TABLE_TYPE: BASE TABLE
ENGINE: MyISAM
VERSION: 10
ROW_FORMAT: Fixed
TABLE_ROWS: 0
AVG_ROW_LENGTH: 0
DATA_LENGTH: 0
MAX_DATA_LENGTH: 1970324836974591
INDEX_LENGTH: 1024
DATA_FREE: 0
AUTO_INCREMENT: NULL
CREATE_TIME: 2020-11-18 15:56:57
UPDATE_TIME: 2020-11-18 15:56:57
CHECK_TIME: NULL
TABLE_COLLATION: latin1_swedish_ci
CHECKSUM: NULL
CREATE_OPTIONS:
TABLE_COMMENT:
MAX_INDEX_LENGTH: 17179868160
TEMPORARY: N
...

```

Example with temporary = 'y', from [MariaDB 10.3.5](#) :

```

SELECT * FROM information_schema.tables WHERE temporary='y'\G
*****
1. row ****
TABLE_CATALOG: def
TABLE_SCHEMA: information_schema
TABLE_NAME: INNODB_FT_DELETED
TABLE_TYPE: SYSTEM VIEW
ENGINE: MEMORY
VERSION: 11
ROW_FORMAT: Fixed
TABLE_ROWS: NULL
AVG_ROW_LENGTH: 9
DATA_LENGTH: 0
MAX_DATA_LENGTH: 9437184
INDEX_LENGTH: 0
DATA_FREE: 0
AUTO_INCREMENT: NULL
CREATE_TIME: 2020-11-17 21:54:02
UPDATE_TIME: NULL
CHECK_TIME: NULL
TABLE_COLLATION: utf8_general_ci
CHECKSUM: NULL
CREATE_OPTIONS: max_rows=1864135
TABLE_COMMENT:
MAX_INDEX_LENGTH: 0
TEMPORARY: Y
...

```

## View Tables in Order of Size

Returns a list of all tables in the database, ordered by size:

```

SELECT table_schema AS `DB`, table_name AS `Table`,
ROUND(((data_length + index_length) / 1024 / 1024), 2) `Size (MB)`
FROM information_schema.TABLES
ORDER BY (data_length + index_length) DESC;

```

DB	Table	Size (MB)
wordpress	wp_simple_history_contexts	7.05
wordpress	wp_posts	6.59
wordpress	wp_simple_history	3.05
wordpress	wp_comments	2.73
wordpress	wp_commentmeta	2.47
wordpress	wp_simple_login_log	2.03
...		

## See Also

- [mysqlshow](#)
- [SHOW TABLE STATUS](#)
- [Finding Tables Without Primary Keys](#)

## 1.1.2.9.1.1.51 Information Schema TABLESPACES Table

The [Information Schema](#)

```

TABLESPACES
table contains information about active tablespaces..

```

The table is a MariaDB and MySQL extension, and does not include information about InnoDB tablespaces.

Column	Description
TABLESPACE_NAME	
ENGINE	

TABLESPACE_TYPE	
LOGFILE_GROUP_NAME	
EXTENT_SIZE	
AUTOEXTEND_SIZE	
MAXIMUM_SIZE	
NODEGROUP_ID	
TABLESPACE_COMMENT	

## 1.1.2.9.1.1.52 Information Schema THREAD\_POOL\_GROUPS Table

MariaDB starting with 10.5

The [Information Schema](#)

THREAD\_POOL\_GROUPS  
table was introduced in [MariaDB 10.5.0](#).

The table provides information about [thread pool](#) groups, and contains the following columns:

Column	Description
GROUP_ID	
CONNECTIONS	
THREADS	
ACTIVE_THREADS	
STANDBY_THREADS	
QUEUE_LENGTH	
HAS_LISTENER	
IS_STALLED	

Setting [thread\\_pool\\_dedicated\\_listener](#) will give each group its own dedicated listener, and the listener thread will not pick up work items. As a result, the actual queue size in the table will be more exact, since IO requests are immediately dequeued from poll, without delay.

## 1.1.2.9.1.1.53 Information Schema THREAD\_POOL\_STATS Table

MariaDB starting with [10.5](#)

The [Information Schema](#)

THREAD\_POOL\_STATS

table was introduced in [MariaDB 10.5.0](#).

The table provides performance counter information for the [thread pool](#), and contains the following columns:

Column	Description
GROUP_ID	
THREAD_CREATATIONS	
THREAD_CREATATIONS_DUE_TO_STALL	
WAKES	
WAKES_DUE_TO_STALL	
THROTTLES	
STALLS	
POLLS_BY_LISTENER	
POLLS_BY_WORKER	
DEQUEUES_BY_LISTENER	
DEQUEUES_BY_WORKER	

## 1.1.2.9.1.1.54 Information Schema THREAD\_POOL\_WAITS Table

MariaDB starting with [10.5](#)

The [Information Schema](#)

THREAD\_POOL\_WAITS

table was introduced in [MariaDB 10.5.0](#).

The table provides wait counters for the [thread pool](#), and contains the following columns:

Column	Description
REASON	

COUNT	
-------	--

## 1.1.2.9.1.1.55 Information Schema TRIGGERS Table

### Contents

1. See also

The [Information Schema](#)

TRIGGERS  
table contains information about [triggers](#).

It has the following columns:

Column	Description
TRIGGER_CATALOG	Always def
TRIGGER_SCHEMA	Database name in which the trigger occurs.
TRIGGER_NAME	Name of the trigger.
EVENT_MANIPULATION	The event that activates the trigger. One of INSERT , UPDATE or 'DELETE'
EVENT_OBJECT_CATALOG	Always def
EVENT_OBJECT_SCHEMA	Database name on which the trigger acts.
EVENT_OBJECT_TABLE	Table name on which the trigger acts.
ACTION_ORDER	Indicates the order that the action will be performed in (of the list of a table's triggers with identical EVENT_MANIPULATION and ACTION_TIMING values). Before MariaDB 10.2.3 introduced the <a href="#">FOLLOWS</a> and <a href="#">PRECEDES clauses</a> , always 0
ACTION_CONDITION	NULL
ACTION_STATEMENT	Trigger body, UTF-8 encoded.
ACTION_ORIENTATION	Always ROW
ACTION_TIMING	Whether the trigger acts BEFORE or AFTER the event that triggers it.

ACTION_REFERENCE_OLD_TABLE	Always NULL
ACTION_REFERENCE_NEW_TABLE	Always NULL
ACTION_REFERENCE_OLD_ROW	Always OLD
ACTION_REFERENCE_NEW_ROW	Always NEW
CREATED	Always NULL
SQL_MODE	The <code>SQL_MODE</code> when the trigger was created, and which it uses for execution.
DEFINER	The account that created the trigger, in the form <code>user_name@host_name</code>
CHARACTER_SET_CLIENT	The client <code>character set</code> when the trigger was created, from the session value of the <code>character_set_client</code> system variable.
COLLATION_CONNECTION	The client <code>collation</code> when the trigger was created, from the session value of the <code>collation_connection</code> system variable.
DATABASE_COLLATION	<code>Collation</code> of the associated database.

Queries to the

`TRIGGERS`  
table will return information only for databases and tables for which you have the  
`TRIGGER` privilege. Similar information is returned by the

`SHOW TRIGGERS`

statement.

## See also

- [Trigger Overview](#)
- 

[CREATE TRIGGER](#)

•

[DROP TRIGGER](#)

•

[SHOW TRIGGERS](#)

```
SHOW CREATE TRIGGER
```

- Trigger Limitations

## 1.1.2.9.1.1.56 Information Schema USER\_PRIVILEGES Table

The [Information Schema](#)

USER\_PRIVILEGES

table contains global user privilege information derived from the

`mysql.user`

grant table.

It contains the following columns:

Column	Description
GRANTEE	In the format <code>user_name@host_name</code>
TABLE_CATALOG	Always def
PRIVILEGE_TYPE	The specific privilege, for example SELECT , INSERT , UPDATE or REFERENCES
IS_GRANTABLE	Whether the user has the <code>GRANT OPTION</code> for this privilege.

Similar information can be accessed with the

```
SHOW GRANTS
```

statement. See the

[GRANT](#)

article for more about privileges.

This information is also stored in the `user` table, in the

`mysql`

system database.

## 1.1.2.9.1.1.57 Information Schema USER\_STATISTICS Table

The [Information Schema](#)

USER\_STATISTICS

table holds statistics about user activity. This is part of the [User Statistics](#) feature, which is not enabled by default.

You can use this table to find out such things as which user is causing the most load and which users are being abusive. You can also use this table to measure how close to capacity the server may be.

It contains the following columns:

Field	Type	Notes
-------	------	-------

USER	varchar(48)	The username. The value '#mysql_system_user#' appears when there is no username (such as for the slave SQL thread).
TOTAL_CONNECTIONS	int(21)	The number of connections created for this user.
CONCURRENT_CONNECTIONS	int(21)	The number of concurrent connections for this user.
CONNECTED_TIME	int(21)	The cumulative number of seconds elapsed while there were connections from this user.
BUSY_TIME	double	The cumulative number of seconds there was activity on connections from this user.
CPU_TIME	double	The cumulative CPU time elapsed while servicing this user's connections.
BYTES_RECEIVED	int(21)	The number of bytes received from this user's connections.
BYTES_SENT	int(21)	The number of bytes sent to this user's connections.
BINLOG_BYTES_WRITTEN	int(21)	The number of bytes written to the <a href="#">binary log</a> from this user's connections.
ROWS_READ	int(21)	The number of rows read by this user's connections.
ROWS_SENT	int(21)	The number of rows sent by this user's connections.
ROWS_DELETED	int(21)	The number of rows deleted by this user's connections.
ROWS_INSERTED	int(21)	The number of rows inserted by this user's connections.
ROWS_UPDATED	int(21)	The number of rows updated by this user's connections.
SELECT_COMMANDS	int(21)	The number of <a href="#">SELECT</a> commands executed from this user's connections.
UPDATE_COMMANDS	int(21)	The number of <a href="#">UPDATE</a> commands executed from this user's connections.
OTHER_COMMANDS	int(21)	The number of other commands executed from this user's connections.
COMMIT_TRANSACTIONS	int(21)	The number of <a href="#">COMMIT</a> commands issued by this user's connections.

ROLLBACK_TRANSACTIONS	int(21)	The number of <a href="#">ROLLBACK</a> commands issued by this user's connections.
DENIED_CONNECTIONS	int(21)	The number of connections denied to this user.
LOST_CONNECTIONS	int(21)	The number of this user's connections that were terminated uncleanly.
ACCESS_DENIED	int(21)	The number of times this user's connections issued commands that were denied.
EMPTY_QUERIES	int(21)	The number of times this user's connections sent empty queries to the server.
TOTAL_SSL_CONNECTIONS	int(21)	The number of <a href="#">TLS connections</a> created for this user. (>= <a href="#">MariaDB 10.1.1</a> )
MAX_STATEMENT_TIME_EXCEEDED	int(21)	The number of times a statement was aborted, because it was executed longer than its <a href="#">MAX_STATEMENT_TIME</a> threshold. (>= <a href="#">MariaDB 10.1.1</a> )

## Example

```
SELECT * FROM information_schema.USER_STATISTICS\G
*****
1. row *****
    USER: root
    TOTAL_CONNECTIONS: 1
    CONCURRENT_CONNECTIONS: 0
    CONNECTED_TIME: 297
        BUSY_TIME: 0.001725
        CPU_TIME: 0.001982
    BYTES RECEIVED: 388
    BYTES SENT: 2327
    BINLOG_BYTES_WRITTEN: 0
        ROWS READ: 0
        ROWS SENT: 12
        ROWS_DELETED: 0
        ROWS_INSERTED: 13
        ROWS_UPDATED: 0
    SELECT_COMMANDS: 4
    UPDATE_COMMANDS: 0
    OTHER_COMMANDS: 3
    COMMIT_TRANSACTIONS: 0
    ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
    LOST_CONNECTIONS: 0
    ACCESS_DENIED: 0
    EMPTY_QUERIES: 1
```

## 1.1.2.9.1.1.58 Information Schema USER\_VARIABLES Table

MariaDB 10.2.0

The

USER\_VARIABLES  
table was introduced in [MariaDB 10.2.0](#) as part of the

[user\\_variables](#)

plugin.

## Description

The

USER\_VARIABLES  
table is created when the

[user\\_variables](#)

plugin is enabled, and contains information about [user-defined variables](#).

The table contains the following columns:

Column	Description
VARIABLE_NAME	Variable name.
VARIABLE_VALUE	Variable value.
VARIABLE_TYPE	Variable <a href="#">type</a> .
CHARACTER_SET_NAME	<a href="#">Character set</a> .

## Example

```
SELECT * FROM information_schema.USER_VARIABLES ORDER BY VARIABLE_NAME;
+-----+-----+-----+-----+
| VARIABLE_NAME | VARIABLE_VALUE | VARIABLE_TYPE | CHARACTER_SET_NAME |
+-----+-----+-----+-----+
| var          | 0             | INT           | utf8            |
| var2         | abc           | VARCHAR       | utf8            |
+-----+-----+-----+-----+
```

## 1.1.2.9.1.1.59 Information Schema VIEWS Table

The [Information Schema](#)

VIEWS  
table contains information about [views](#). The  
SHOW VIEW  
[privilege](#) is required to view the table.

It has the following columns:

Column	Description	Added
TABLE_CATALOG	Aways def .	
TABLE_SCHEMA	Database name containing the view.	
TABLE_NAME	View table name.	
VIEW_DEFINITION	Definition of the view.	

CHECK_OPTION	YES if the WITH CHECK_OPTION clause has been specified, NO otherwise.	
IS_UPDATABLE	Whether the view is updatable or not.	
DEFINER	Account specified in the DEFINER clause (or the default when created).	
SECURITY_TYPE	SQL SECURITY characteristic, either DEFINER or INVOKER	
CHARACTER_SET_CLIENT	The client <a href="#">character set</a> when the view was created, from the session value of the <a href="#">character_set_client</a> system variable.	
COLLATION_CONNECTION	The client <a href="#">collation</a> when the view was created, from the session value of the <a href="#">collation_connection</a> system variable.	
ALGORITHM	The algorithm used in the view. See <a href="#">View Algorithms</a> .	MariaDB 10.1.3

## Example

```
SELECT * FROM information_schema.VIEWS\G
*****
1. row ****
  TABLE_CATALOG: def
  TABLE_SCHEMA: test
  TABLE_NAME: v
  VIEW_DEFINITION: select `test`.`t`.`qty` AS `qty`, `test`.`t`.`price` AS `price`,(`test`.`t`.`qty` * `test`.`t`.`price`) AS
  CHECK_OPTION: NONE
  IS_UPDATABLE: YES
  DEFINER: root@localhost
  SECURITY_TYPE: DEFINER
  CHARACTER_SET_CLIENT: utf8
  COLLATION_CONNECTION: utf8_general_ci
  ALGORITHM: UNDEFINED
```

## See also

- [CREATE VIEW](#)
- [ALTER VIEW](#)
- [DROP VIEW](#)

SHOW CREATE VIEWS

## 1.1.2.9.1.1.60 Information Schema WSREP\_MEMBERSHIP Table

The

WSREP\_STATUS

table makes Galera node cluster membership information available through the [Information Schema](#). The same information can be returned using the `SHOW WSREP_MEMBERSHIP` statement. Only users with the `SUPER` can access information from this table.

The

WSREP\_MEMBERSHIP

table is part of the [WSREP\\_INFO plugin](#).

### Example

```
SELECT * FROM information_schema.WSREP_MEMBERSHIP;
+-----+-----+-----+
| INDEX | UUID           | NAME   | ADDRESS      |
+-----+-----+-----+
| 0    | 46da96e3-6e9e-11e4-95a2-f609aa5444b3 | node1  | 10.0.2.15:16000 |
| 1    | 5f6bc72a-6e9e-11e4-84ed-57ec6780a3d3 | node2  | 10.0.2.15:16001 |
| 2    | 7473fd75-6e9e-11e4-91de-0b541ad91bd0 | node3  | 10.0.2.15:16002 |
+-----+-----+-----+
```

## 1.1.2.9.1.1.61 Information Schema WSREP\_STATUS Table

The WSREP\_STATUS table makes Galera node cluster status information available through the [Information Schema](#). The same information can be returned using the `SHOW WSREP_STATUS` statement. Only users with the `SUPER` privilege can access information from this table.

The

WSREP\_STATUS

table is part of the [WSREP\\_INFO plugin](#).

### Example

```
SELECT * FROM information_schema.WSREP_STATUS\G
***** 1. row *****
  NODE_INDEX: 0
  NODE_STATUS: Synced
  CLUSTER_STATUS: Primary
  CLUSTER_SIZE: 3
  CLUSTER_STATE_UUID: 00b0fbad-6e84-11e4-8a8b-376f19ce8ee7
  CLUSTER_STATE_SEQNO: 2
  CLUSTER_CONF_ID: 3
  GAP: NO
  PROTOCOL_VERSION: 3
```

## 1.1.2.9.1.2 Extended SHOW

### 1.1.2.9.1.3 TIME\_MS column in INFORMATION\_SCHEMA.PROCESSLIST

In MariaDB, an extra column

TIME\_MS

has been added to the [INFORMATION\\_SCHEMA.PROCESSLIST](#) table. This column shows the same information as the column 'TIME', but in units of milliseconds with microsecond precision (the unit and precision of the TIME column is one second).

For details about microseconds support in MariaDB, see [microseconds in MariaDB](#).

The value displayed in the

TIME

and

TIME\_MS

columns is the period of time that the given thread has been in its current state. Thus it can be used to check for example how long a thread has been executing the current query, or for how long it has been idle.

```
select id, time, time_ms, command, state from
    information_schema.processlist, (select sleep(2)) t;
+----+----+----+----+
| id | time | time_ms | command | state   |
+----+----+----+----+
| 37 | 2   | 2000.493 | Query   | executing |
+----+----+----+----+
```

Note that as a difference to MySQL, in MariaDB the

TIME

column (and also the

TIME\_MS

column) are not affected by any setting of [@TIMESTAMP](#). This means that it can be reliably used also for threads that change [@TIMESTAMP](#) (such as the [replication](#) SQL thread). See also [MySQL Bug #22047](#).

As a consequence of this, the

TIME

column of

SHOW FULL PROCESSLIST

and

INFORMATION\_SCHEMA.PROCESSLIST

can not be used to determine if a slave is lagging behind. For this, use instead the

Seconds\_Behind\_Master

column in the output of [SHOW SLAVE STATUS](#).

The addition of the TIME\_MS column is based on the microsec\_process patch, developed by [Percona](#).

## 1.1.2.9.2 Performance Schema

### 1.1.2.9.2.1 Performance Schema Tables

#### 1.1.2.9.2.1.1 List of Performance Schema Tables

Below is a list of all [Performance Schema](#) tables as well as a brief description of each of them.

Table	Description
accounts	Client account connection statistics.
cond_instances	Synchronization object instances.
events_stages_current	Current stage events.
events_stages_history	Ten most recent stage events per thread.

<code>events_stages_history_long</code>	Ten thousand most recent stage events.
<code>events_stages_summary_by_account_by_event_name</code>	Summarized stage events per account and event name.
<code>events_stages_summary_by_host_by_event_name</code>	Summarized stage events per host and event name.
<code>events_stages_summary_by_thread_by_event_name</code>	Summarized stage events per thread and event name.
<code>events_stages_summary_by_user_by_event_name</code>	Summarized stage events per user name and event name.
<code>events_stages_summary_global_by_event_name</code>	Summarized stage events per event name.
<code>events_statements_current</code>	Current statement events.
<code>events_statements_history</code>	Ten most recent events per thread.
<code>events_statements_history_long</code>	Ten thousand most recent stage events.
<code>events_statements_summary_by_account_by_event_name</code>	Summarized statement events per account and event name.
<code>events_statements_summary_by_digest</code>	Summarized statement events by scheme and digest.
<code>events_statements_summary_by_host_by_event_name</code>	Summarized statement events by host and event name.

<code>events_statements_summary_by_program</code>	
<code>events_statements_summary_by_thread_by_event_name</code>	Summarized statement events by thread and event name.
<code>events_statements_summary_by_user_by_event_name</code>	Summarized statement events by user and event name.
<code>events_statements_summary_global_by_event_name</code>	Summarized statement events by event name.
<code>events_transactions_current</code>	Current transaction events for each thread.
<code>events_transactions_history</code>	Most recent completed transaction events for each thread.
<code>events_transactions_history_long</code>	Most recent completed transaction events that have ended globally.
<code>events_transactions_summary_by_account_by_event_name</code>	Transaction events aggregated by account and event.
<code>events_transactions_summary_by_host_by_event_name</code>	Transaction events aggregated by host and event..
<code>events_transactions_summary_by_thread_by_event_name</code>	Transaction events aggregated by thread and event..
<code>events_transactions_summary_by_user_by_event_name</code>	Transaction events aggregated by user and event..
<code>events_transactions_summary_global_by_event_name</code>	Transaction events aggregated by event name.

<code>events_waits_current</code>	Current wait events.
<code>events_waits_history</code>	Ten most recent wait events per thread.
<code>events_waits_history_long</code>	Ten thousand most recent wait events per thread.
<code>events_waits_summary_by_account_by_event_name</code>	Summarized wait events by account and event name.
<code>events_waits_summary_by_host_by_event_name</code>	Summarized wait events by host and event name.
<code>events_waits_summary_by_instance</code>	Summarized wait events by instance.
<code>events_waits_summary_by_thread_by_event_name</code>	Summarized wait events by thread and event name.
<code>events_waits_summary_by_user_by_event_name</code>	Summarized wait events by user and event name.
<code>events_waits_summary_global_by_event_name</code>	Summarized wait events by event name.
<code>file_instances</code>	Seen files.
<code>file_summary_by_event_name</code>	File events summarized by event name.
<code>file_summary_by_instance</code>	File events summarized by instance.

<code>global_status</code>	Global status variables and values.
<code>host_cache</code>	Host and IP information.
<code>hosts</code>	Connections by host.
<code>memory_summary_by_account_by_event_name</code>	Memory usage statistics aggregated by account and event.
<code>memory_summary_by_host_by_event_name</code>	Memory usage statistics aggregated by host. and event.
<code>memory_summary_by_thread_by_event_name</code>	Memory usage statistics aggregated by thread and event..
<code>memory_summary_by_user_by_event_name</code>	Memory usage statistics aggregated by user and event..
<code>memory_summary_global_by_event_name</code>	Memory usage statistics aggregated by event.
<code>metadata_locks</code>	<code>Metadata locks</code> .
<code>mutex_instances</code>	Seen mutexes.
<code>objects_summary_global_by_type</code>	Object wait events.
<code>performance_timers</code>	Available event timers.

<code>prepared_statements_instances</code>	Aggregate statistics of prepared statements.
<code>replication_applier_configuration</code>	Configuration settings affecting replica transactions.
<code>replication_applier_status</code>	General transaction execution status on the replica.
<code>replication_applier_status_by_coordinator</code>	Coordinator thread specific information.
<code>replication_applier_status_by_worker</code>	Replica worker thread specific information.
<code>replication_connection_configuration</code>	Replica's configuration settings used for connecting to the primary.
<code>rwlock_instances</code>	Seen read-write locks.
<code>session_account_connect_attrs</code>	Current session connection attributes.
<code>session_connect_attrs</code>	All session connection attributes.
<code>session_status</code>	Session status variables and values.
<code>setup_actors</code>	Details on foreground thread monitoring.
<code>setup_consumers</code>	Consumers for which event information is stored.

<code>setup_instruments</code>	Instrumented objects for which events are collected.
<code>setup_objects</code>	Objects to be monitored.
<code>setup_timers</code>	Currently selected event timers.
<code>socket_instances</code>	Active connections.
<code>socket_summary_by_event_name</code>	Timer and byte count statistics by socket instrument.
<code>socket_summary_by_instance</code>	Timer and byte count statistics by socket instance.
<code>status_by_thread</code>	Status variable info about active foreground threads.
<code>table_io_waits_summary_by_index_usage</code>	Aggregate table I/O wait events by index.
<code>table_io_waits_summary_by_table</code>	Aggregate table I/O wait events by table.
<code>table_lock_waits_summary_by_table</code>	Aggregate table lock wait events by table.
<code>threads</code>	Server thread information.
<code>users</code>	Connection statistics by user.

## 1.1.2.9.2.1.2 Performance Schema accounts Table

## Description

Each account that connects to the server is stored as a row in the accounts table, along with current and total connections.

The table size is determined at startup by the value of the [performance\\_schema\\_accounts\\_size](#) system variable. If this is set to 0, account statistics will be disabled.

Column	Description
USER	The connection's client user name for the connection, or NULL if an internal thread.
HOST	The connection client's host name, or NULL if an internal thread.
CURRENT_CONNECTIONS	Current connections for the account.
TOTAL_CONNECTIONS	Total connections for the account.

The

USER  
and  
HOST

values shown here are the username and host used for user connections, not the patterns used to check permissions.

## Example

```
SELECT * FROM performance_schema.accounts;
+-----+-----+-----+-----+
| USER      | HOST     | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+-----+-----+-----+-----+
| root      | localhost |                1 |                 2 |
| NULL      | NULL      |               20 |                23 |
| debian-sys-maint | localhost |                0 |                35 |
+-----+-----+-----+-----+
```

## 1.1.2.9.2.1.3 Performance Schema cond\_instances Table

### Description

The

cond\_instances

table lists all conditions while the server is executing. A condition, or instrumented condition object, is an internal code mechanism used for signalling that a specific event has occurred so that any threads waiting for this condition can continue.

The maximum number of conditions stored in the performance schema is determined by the [performance\\_schema\\_max\\_cond\\_instances](#) system variable.

Column	Description
NAME	Client user name for the connection, or NULL if an internal thread.
OBJECT_INSTANCE_BEGIN	Address in memory of the instrumented condition.

## 1.1.2.9.2.1.4 Performance Schema events\_stages\_current Table

The

events\_stages\_current

table contains current stage events, with each row being a record of a thread and its most recent stage event.

The table contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if the event has not ended or timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if the event has not ended or timing is not collected.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. One of transaction , statement , stage or wait .

It is possible to empty this table with a

```
TRUNCATE TABLE
statement.
```

The related tables, [events\\_stages\\_history](#) and [events\\_stages\\_history\\_long](#) derive their values from the current events.

## 1.1.2.9.2.1.5 Performance Schema events\_stages\_history Table

The

```
events_stages_history
```

table by default contains the ten most recent completed stage events per thread. This number can be adjusted by setting the [performance\\_schema\\_events\\_stages\\_history\\_size](#) system variable when the server starts up.

The table structure is identical to the [events\\_stage\\_current](#) table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.

END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. One of transaction , statement , stage or wait .

It is possible to empty this table with a

```
TRUNCATE TABLE
statement.
```

[events\\_stages\\_current](#) and [events\\_stages\\_history\\_long](#) are related tables.

## 1.1.2.9.2.1.6 Performance Schema [events\\_stages\\_history\\_long](#) Table

The

[events\\_stages\\_history\\_long](#)

table by default contains the ten thousand most recent completed stage events. This number can be adjusted by setting the [performance\\_schema\\_events\\_stages\\_history\\_long\\_size](#) system variable when the server starts up.

The table structure is identical to the

[events\\_stage\\_current](#)

table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table

SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. One of transaction ,
	statement ,
	stage or wait .

It is possible to empty this table with a

```
TRUNCATE TABLE
statement.
```

[events\\_stages\\_current](#) and [events\\_stages\\_history](#) are related tables.

## 1.1.2.9.2.1.7 Performance Schema `events_stages_summary_by_account_by_event_name` Table

The table lists stage events, summarized by account and event name.

It contains the following columns:

Column	Description
USER	User. Used together with HOST and EVENT_NAME for grouping events.
HOST	Host. Used together with USER and EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER and HOST for grouping events.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.

AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

## Example

```

SELECT * FROM events_stages_summary_by_account_by_event_name\G
...
***** 325. row *****
    USER: NULL
    HOST: NULL
  EVENT_NAME: stage/sql/Waiting for event metadata lock
  COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 326. row *****
    USER: NULL
    HOST: NULL
  EVENT_NAME: stage/sql/Waiting for commit lock
  COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 327. row *****
    USER: NULL
    HOST: NULL
  EVENT_NAME: stage/aria/Waiting for a resource
  COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0

```

### 1.1.2.9.2.1.8 Performance Schema events\_stages\_summary\_by\_host\_by\_event\_name Table

The table lists stage events, summarized by host and event name.

It contains the following columns:

Column	Description
HOST	Host. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with HOST for grouping events.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.

MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.
----------------	---

## Example

```

SELECT * FROM events_stages_summary_by_host_by_event_name\G
...
***** 216. row *****
HOST: NULL
EVENT_NAME: stage/sql/Waiting for event metadata lock
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 217. row *****
HOST: NULL
EVENT_NAME: stage/sql/Waiting for commit lock
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 218. row *****
HOST: NULL
EVENT_NAME: stage/aria/Waiting for a resource
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0

```

### 1.1.2.9.2.1.9 Performance Schema events\_stages\_summary\_by\_thread\_by\_event\_name Table

The table lists stage events, summarized by thread and event name.

It contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_NAME uniquely identifies the row.
EVENT_NAME	Event name. Used together with THREAD_ID for grouping events.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

## Example

```
SELECT * FROM events_stages_summary_by_thread_by_event_name\G
...
***** 2287. row *****
    THREAD_ID: 64
    EVENT_NAME: stage/sql/Waiting for event metadata lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 2288. row *****
    THREAD_ID: 64
    EVENT_NAME: stage/sql/Waiting for commit lock
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 2289. row *****
    THREAD_ID: 64
    EVENT_NAME: stage/aria/Waiting for a resource
    COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

### 1.1.2.9.2.1.10 Performance Schema events\_stages\_summary\_by\_user\_by\_event\_name Table

The table lists stage events, summarized by user and event name.

It contains the following columns:

Column	Description
USER	User. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER for grouping events.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

## Example

```

SELECT * FROM events_stages_summary_by_user_by_event_name\G
...
***** 325. row *****
USER: NULL
EVENT_NAME: stage/sql/Waiting for event metadata lock
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 326. row *****
USER: NULL
EVENT_NAME: stage/sql/Waiting for commit lock
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 327. row *****
USER: NULL
EVENT_NAME: stage/aria/Waiting for a resource
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0

```

## 1.1.2.9.2.1.11 Performance Schema events\_stages\_summary\_global\_by\_event\_name Table

The table lists stage events, summarized by thread and event name.

It contains the following columns:

Column	Description
EVENT_NAME	Event name.
COUNT_STAR	Number of summarized events, which includes all timed and untimed events.
SUM_TIMER_WAIT	Total wait time of the timed summarized events.
MIN_TIMER_WAIT	Minimum wait time of the timed summarized events.
AVG_TIMER_WAIT	Average wait time of the timed summarized events.
MAX_TIMER_WAIT	Maximum wait time of the timed summarized events.

### Example

```

SELECT * FROM events_stages_summary_global_by_event_name\G
...
***** 106. row *****
EVENT_NAME: stage/sql/Waiting for trigger metadata lock
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 107. row *****
EVENT_NAME: stage/sql/Waiting for event metadata lock
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 108. row *****
EVENT_NAME: stage/sql/Waiting for commit lock
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 109. row *****
EVENT_NAME: stage/aria/Waiting for a resource
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0

```

## 1.1.2.9.2.1.12 Performance Schema events\_statements\_history Table

The

`events_statements_history`  
table by default contains the ten most recent completed statement events per thread. This number can be adjusted by setting the `performance_schema_events_statements_history_size` system variable when the server starts up.

The table structure is identical to the `events_statements_current` table structure, and contains the following columns:

The table contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected.

TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
LOCK_TIME	Time in picoseconds spent waiting for locks. The time is calculated in microseconds but stored in picoseconds for compatibility with other timings.
SQL_TEXT	The SQL statement, or NULL if the command is not associated with an SQL statement.
DIGEST	<a href="#">Statement digest</a> .
DIGEST_TEXT	<a href="#">Statement digest text</a> .
CURRENT_SCHEMA	Statement's default database for the statement, or NULL if there was none.
OBJECT_SCHEMA	Reserved, currently NULL
OBJECT_NAME	Reserved, currently NULL
OBJECT_TYPE	Reserved, currently NULL
OBJECT_INSTANCE_BEGIN	Address in memory of the statement object.
MYSQL_ERRNO	Error code. See <a href="#">MariaDB Error Codes</a> for a full list.
RETURNED_SQLSTATE	The <a href="#">SQLSTATE</a> value.
MESSAGE_TEXT	Statement error message. See <a href="#">MariaDB Error Codes</a> .
ERRORS	0 if SQLSTATE signifies completion (starting with 00) or warning (01), otherwise 1 .
WARNINGS	Number of warnings from the diagnostics area.
ROWS_AFFECTED	Number of rows affected the statement affected.
ROWS_SENT	Number of rows returned.
ROWS_EXAMINED	Number of rows read during the statement's execution.

CREATED_TMP_DISK_TABLES	Number of on-disk temp tables created by the statement.
CREATED_TMP_TABLES	Number of temp tables created by the statement.
SELECT_FULL_JOIN	Number of joins performed by the statement which did not use an index.
SELECT_FULL_RANGE_JOIN	Number of joins performed by the statement which used a range search of the first table.
SELECT_RANGE	Number of joins performed by the statement which used a range of the first table.
SELECT_RANGE_CHECK	Number of joins without keys performed by the statement that check for key usage after each row.
SELECT_SCAN	Number of joins performed by the statement which used a full scan of the first table.
SORT_MERGE_PASSES	Number of merge passes by the sort algorithm performed by the statement. If too high, you may need to increase the <code>sort_buffer_size</code> .
SORT_RANGE	Number of sorts performed by the statement which used a range.
SORT_ROWS	Number of rows sorted by the statement.
SORT_SCAN	Number of sorts performed by the statement which used a full table scan.
NO_INDEX_USED	<ul style="list-style-type: none"> <li>0 if the statement performed a table scan with an index,</li> <li>1 if without an index.</li> </ul>
NO_GOOD_INDEX_USED	<ul style="list-style-type: none"> <li>0 if a good index was found for the statement,</li> <li>1 if no good index was found. See the Range checked for each record description in the <a href="#">EXPLAIN</a> article.</li> </ul>
NESTING_EVENT_ID	Reserved, currently NULL .
NESTING_EVENT_TYPE	Reserved, currently NULL .

It is possible to empty this table with a

```
TRUNCATE TABLE
statement.
```

[events\\_statements\\_current](#) and [events\\_statements\\_history\\_long](#) are related tables.

## 1.1.2.9.2.1.13 Performance Schema events\_statements\_history\_long Table

The

`events_statements_history_long`

table by default contains the ten thousand most recent completed statement events. This number can be adjusted by setting the `performance_schema_events_statements_history_long_size` system variable when the server starts up.

The table structure is identical to the `events_statements_current` table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
LOCK_TIME	Time in picoseconds spent waiting for locks. The time is calculated in microseconds but stored in picoseconds for compatibility with other timings.
SQL_TEXT	The SQL statement, or NULL if the command is not associated with an SQL statement.
DIGEST	<code>Statement digest</code> .
DIGEST_TEXT	<code>Statement digest text</code> .
CURRENT_SCHEMA	Statement's default database for the statement, or NULL if there was none.
OBJECT_SCHEMA	Reserved, currently NULL
OBJECT_NAME	Reserved, currently NULL
OBJECT_TYPE	Reserved, currently NULL

OBJECT_INSTANCE_BEGIN	Address in memory of the statement object.
MYSQL_ERRNO	Error code. See <a href="#">MariaDB Error Codes</a> for a full list.
RETURNED_SQLSTATE	The <a href="#">SQLSTATE</a> value.
MESSAGE_TEXT	Statement error message. See <a href="#">MariaDB Error Codes</a> .
ERRORS	<pre> 0 if SQLSTATE signifies completion (starting with 00) or warning (01), otherwise 1 . </pre>
WARNINGS	Number of warnings from the diagnostics area.
ROWS_AFFECTED	Number of rows affected the statement affected.
ROWS_SENT	Number of rows returned.
ROWS_EXAMINED	Number of rows read during the statement's execution.
CREATED_TMP_DISK_TABLES	Number of on-disk temp tables created by the statement.
CREATED_TMP_TABLES	Number of temp tables created by the statement.
SELECT_FULL_JOIN	Number of joins performed by the statement which did not use an index.
SELECT_FULL_RANGE_JOIN	Number of joins performed by the statement which used a range search of the first table.
SELECT_RANGE	Number of joins performed by the statement which used a range of the first table.
SELECT_RANGE_CHECK	Number of joins without keys performed by the statement that check for key usage after each row.
SELECT_SCAN	Number of joins performed by the statement which used a full scan of the first table.
SORT_MERGE_PASSES	Number of merge passes by the sort algorithm performed by the statement. If too high, you may need to increase the <a href="#">sort_buffer_size</a> .

SORT_RANGE	Number of sorts performed by the statement which used a range.
SORT_ROWS	Number of rows sorted by the statement.
SORT_SCAN	Number of sorts performed by the statement which used a full table scan.
NO_INDEX_USED	0 if the statement performed a table scan with an index, 1 if without an index.
NO_GOOD_INDEX_USED	0 if a good index was found for the statement, 1 if no good index was found. See the Range checked for each record description in the <a href="#">EXPLAIN</a> article.
NESTING_EVENT_ID	Reserved, currently NULL .
NESTING_EVENT_TYPE	Reserved, currently NULL .

It is possible to empty this table with a

```
TRUNCATE TABLE
statement.
```

[events\\_statements\\_current](#) and [events\\_statements\\_history](#) are related tables.

## 1.1.2.9.2.1.14 Performance Schema events\_statements\_summary\_by\_account\_by\_event\_name Table

The [Performance Schema](#) events\_statements\_summary\_by\_account\_by\_event\_name table contains statement events summarized by account and event name. It contains the following columns:

Column	Description
USER	User. Used together with HOST and EVENT_NAME for grouping events.
HOST	Host. Used together with USER and EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER and HOST for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.

MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.

SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.

The

\*\_TIMER\_WAIT  
columns only calculate results for timed events, as non-timed events have a  
NULL  
wait time.

## Example

```

SELECT * FROM events_statements_summary_by_account_by_event_name\G
...
***** 521. row *****
    USER: NULL
    HOST: NULL
    EVENT_NAME: statement/com/Error
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
    SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0
***** 522. row *****
    USER: NULL
    HOST: NULL
    EVENT_NAME: statement/com/
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
    SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0

```

## 1.1.2.9.2.1.15 Performance Schema events\_statements\_summary\_by\_digest Table

The [Performance Schema digest](#) is a hashed, normalized form of a statement with the specific data values removed. It allows statistics to be gathered for similar kinds of statements.

The [Performance Schema](#)

`events_statements_summary_by_digest`

table records statement events summarized by schema and digest. It contains the following columns:

Column	Description
SCHEMA NAME	Database name. Records are summarised together with DIGEST

DIGEST	Performance Schema digest . Records are summarised together with SCHEMA NAME .
DIGEST_TEXT	The unhashed form of the digest.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.

SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.
FIRST_SEEN	Time at which the digest was first seen.
LAST_SEEN	Time at which the digest was most recently seen.

The

\*\_TIMER\_WAIT  
columns only calculate results for timed events, as non-timed events have a

NULL  
wait time.

The

events\_statements\_summary\_by\_digest

table is limited in size by the [performance\\_schema\\_digests\\_size](#) system variable. Once the limit has been reached and the table is full, all entries are aggregated in a row with a

NULL

digest. The

COUNT\_STAR

value of this

NULL

row indicates how many digests are recorded in the row and therefore gives an indication of whether

[performance\\_schema\\_digests\\_size](#)

should be increased to provide more accurate statistics.

## 1.1.2.9.2.1.16 Performance Schema events\_statements\_summary\_by\_host\_by\_event\_name Table

The [Performance Schema](#) events\_statements\_summary\_by\_host\_by\_event\_name table contains statement events summarized by host and event name. It contains the following columns:

Column	Description
HOST	Host. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with HOST for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.

SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.

SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.

The

```
* _TIMER_WAIT
columns only calculate results for timed events, as non-timed events have a
NULL
wait time.
```

## Example

```

SELECT * FROM events_statements_summary_by_host_by_event_name\G
...
***** row *****
    HOST: NULL
    EVENT_NAME: statement/com/Error
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0
***** row *****
    HOST: NULL
    EVENT_NAME: statement/com/
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0

```

## 1.1.2.9.2.1.17 Performance Schema events\_statements\_summary\_by\_program Table

MariaDB starting with 10.5.2

The

`events_statements_summary_by_program`  
table, along with many other new [Performance Schema tables](#), was added in [MariaDB 10.5.2](#).

Each row in the the `Performance Schema events_statements_summary_by_program` table summarizes events for a particular stored program (stored procedure, stored function, trigger or event).

It contains the following fields.

Column	Type	Null	Description
--------	------	------	-------------

OBJECT_TYPE	enum('EVENT', 'FUNCTION', 'PROCEDURE', 'TABLE', 'TRIGGER')	YES	Object type for which the summary is generated.
OBJECT_SCHEMA	varchar(64)	NO	The schema of the object for which the summary is generated.
OBJECT_NAME	varchar(64)	NO	The name of the object for which the summary is generated.
COUNT_STAR	bigint(20) unsigned	NO	The number of summarized events (from events_statements_current). This value includes all events, whether timed or nontimed.
SUM_TIMER_WAIT	bigint(20) unsigned	NO	The number of summarized events (from events_statements_current). This value includes all events, whether timed or nontimed.
MIN_TIMER_WAIT	bigint(20) unsigned	NO	The minimum wait time of the summarized timed events.
AVG_TIMER_WAIT	bigint(20) unsigned	NO	The average wait time of the summarized timed events.
MAX_TIMER_WAIT	bigint(20) unsigned	NO	The maximum wait time of the summarized timed events.
COUNT_STATEMENTS	bigint(20) unsigned	NO	Total number of nested statements invoked during stored program execution.
SUM_STATEMENTS_WAIT	bigint(20) unsigned	NO	The total wait time of the summarized timed statements. This value is calculated only for timed statements because nontimed statements have a wait time of NULL. The same is true for the other xxx_STATEMENT_WAIT values.
MIN_STATEMENTS_WAIT	bigint(20) unsigned	NO	The minimum wait time of the summarized timed statements.
AVG_STATEMENTS_WAIT	bigint(20) unsigned	NO	The average wait time of the summarized timed statements.
MAX_STATEMENTS_WAIT	bigint(20) unsigned	NO	The maximum wait time of the summarized timed statements.
SUM_LOCK_TIME	bigint(20) unsigned	NO	The total time spent (in picoseconds) waiting for table locks for the summarized statements.
SUM_ERRORS	bigint(20) unsigned	NO	The total number of errors that occurred for the summarized statements.
SUM_WARNINGS	bigint(20) unsigned	NO	The total number of warnings that occurred for the summarized statements.
SUM_ROWS_AFFECTED	bigint(20) unsigned	NO	The total number of affected rows by the summarized statements.
SUM_ROWS_SENT	bigint(20) unsigned	NO	The total number of rows returned by the summarized statements.
SUM_ROWS_EXAMINED	bigint(20) unsigned	NO	The total number of rows examined by the summarized statements. The total number of affected rows by the summarized statements.
SUM_CREATED_TMP_DISK_TABLES	bigint(20) unsigned	NO	The total number of on-disk temporary tables created by the summarized statements.
SUM_CREATED_TMP_TABLES	bigint(20) unsigned	NO	The total number of in-memory temporary tables created by the summarized statements.
SUM_SELECT_FULL_JOIN	bigint(20) unsigned	NO	The total number of full joins executed by the summarized statements.
SUM_SELECT_FULL_RANGE_JOIN	bigint(20) unsigned	NO	The total number of range search joins executed by the summarized statements.
SUM_SELECT_RANGE	bigint(20) unsigned	NO	The total number of joins that used ranges on the first table executed by the summarized statements.
SUM_SELECT_RANGE_CHECK	bigint(20) unsigned	NO	The total number of joins that check for key usage after each row executed by the summarized statements.
SUM_SELECT_SCAN	bigint(20) unsigned	NO	The total number of joins that did a full scan of the first table executed by the summarized statements.
SUM_SORT_MERGE_PASSES	bigint(20) unsigned	NO	The total number of merge passes that the sort algorithm has had to do for the summarized statements.
SUM_SORT_RANGE	bigint(20) unsigned	NO	The total number of sorts that were done using ranges for the summarized statements.
SUM_SORT_ROWS	bigint(20) unsigned	NO	The total number of sorted rows that were sorted by the summarized statements.
SUM_SORT_SCAN	bigint(20) unsigned	NO	The total number of sorts that were done by scanning the table by the summarized statements.
SUM_NO_INDEX_USED	bigint(20) unsigned	NO	The total number of statements that performed a table scan without using an index.
SUM_NO_GOOD_INDEX_USED	bigint(20) unsigned	NO	The total number of statements where no good index was found.

## 1.1.2.9.2.1.18 Performance Schema events\_statements\_summary\_by\_thread\_by\_event\_name Table

The [Performance Schema](#)

events\_statements\_summary\_by\_thread\_by\_event\_name

table contains statement events summarized by thread and event name. It contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_NAME uniquely identifies the row.
EVENT_NAME	Event name. Used together with THREAD_ID for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.

SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.

<pre>SUM_NO_GOOD_INDEX_USED</pre>	<p>Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.</p>
-----------------------------------	---

The

```
*_TIMER_WAIT
columns only calculate results for timed events, as non-timed events have a
NULL
wait time.
```

## Example

```
SELECT * FROM events_statements_summary_by_thread_by_event_name\G
...
***** 3653. row *****
    THREAD_ID: 64
    EVENT_NAME: statement/com/Error
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
    SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0
***** 3654. row *****
    THREAD_ID: 64
    EVENT_NAME: statement/com/
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
    SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0
```

## 1.1.2.9.2.1.19 Performance Schema

# events\_statements\_summary\_by\_user\_by\_event\_name Table

The [Performance Schema](#)

`events_statements_summary_by_user_by_event_name`

table contains statement events summarized by user and event name. It contains the following columns:

Column	Description
USER	User. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.

SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.
SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.

<pre>SUM_NO_GOOD_INDEX_USED</pre>	<p>Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.</p>
-----------------------------------	---

The

```
*_TIMER_WAIT
columns only calculate results for timed events, as non-timed events have a
NULL
wait time.
```

## Example

```
SELECT * FROM events_statements_summary_by_user_by_event_name\G
...
***** 521. row *****
    USER: NULL
    EVENT_NAME: statement/com/Error
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
    SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0
***** 522. row *****
    USER: NULL
    EVENT_NAME: statement/com/
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
    SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0
```

## 1.1.2.9.2.1.20 Performance Schema

# events\_statements\_summary\_global\_by\_event\_name Table

The [Performance Schema](#)

events\_statements\_summary\_global\_by\_event\_name

table contains statement events summarized by event name. It contains the following columns:

Column	Description
EVENT_NAME	Event name.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
SUM_LOCK_TIME	Sum of the LOCK_TIME column in the events_statements_current table.
SUM_ERRORS	Sum of the ERRORS column in the events_statements_current table.
SUM_WARNINGS	Sum of the WARNINGS column in the events_statements_current table.
SUM_ROWS_AFFECTED	Sum of the ROWS_AFFECTED column in the events_statements_current table.
SUM_ROWS_SENT	Sum of the ROWS_SENT column in the events_statements_current table.
SUM_ROWS_EXAMINED	Sum of the ROWS_EXAMINED column in the events_statements_current table.
SUM_CREATED_TMP_DISK_TABLES	Sum of the CREATED_TMP_DISK_TABLES column in the events_statements_current table.

SUM_CREATED_TMP_TABLES	Sum of the CREATED_TMP_TABLES column in the events_statements_current table.
SUM_SELECT_FULL_JOIN	Sum of the SELECT_FULL_JOIN column in the events_statements_current table.
SUM_SELECT_FULL_RANGE_JOIN	Sum of the SELECT_FULL_RANGE_JOIN column in the events_statements_current table.
SUM_SELECT_RANGE	Sum of the SELECT_RANGE column in the events_statements_current table.
SUM_SELECT_RANGE_CHECK	Sum of the SELECT_RANGE_CHECK column in the events_statements_current table.
SUM_SELECT_SCAN	Sum of the SELECT_SCAN column in the events_statements_current table.
SUM_SORT_MERGE_PASSES	Sum of the SORT_MERGE_PASSES column in the events_statements_current table.
SUM_SORT_RANGE	Sum of the SORT_RANGE column in the events_statements_current table.
SUM_SORT_ROWS	Sum of the SORT_ROWS column in the events_statements_current table.
SUM_SORT_SCAN	Sum of the SORT_SCAN column in the events_statements_current table.
SUM_NO_INDEX_USED	Sum of the NO_INDEX_USED column in the events_statements_current table.
SUM_NO_GOOD_INDEX_USED	Sum of the NO_GOOD_INDEX_USED column in the events_statements_current table.

The

\*\_TIMER\_WAIT  
columns only calculate results for timed events, as non-timed events have a  
NULL

wait time.

## Example

```
SELECT * FROM events_statements_summary_global_by_event_name\G
...
***** 173. row *****
    EVENT_NAME: statement/com/Error
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0
***** 174. row *****
    EVENT_NAME: statement/com/
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
    SUM_LOCK_TIME: 0
    SUM_ERRORS: 0
    SUM_WARNINGS: 0
    SUM_ROWS_AFFECTED: 0
    SUM_ROWS_SENT: 0
    SUM_ROWS_EXAMINED: 0
SUM_CREATED_TMP_DISK_TABLES: 0
    SUM_CREATED_TMP_TABLES: 0
    SUM_SELECT_FULL_JOIN: 0
SUM_SELECT_FULL_RANGE_JOIN: 0
    SUM_SELECT_RANGE: 0
    SUM_SELECT_RANGE_CHECK: 0
    SUM_SELECT_SCAN: 0
SUM_SORT_MERGE_PASSES: 0
    SUM_SORT_RANGE: 0
    SUM_SORT_ROWS: 0
    SUM_SORT_SCAN: 0
    SUM_NO_INDEX_USED: 0
    SUM_NO_GOOD_INDEX_USED: 0
```

### 1.1.2.9.2.1.21 Performance Schema events\_transactions\_current Table

MariaDB starting with [10.5.2](#)

The events\_transactions\_current table was introduced in [MariaDB 10.5.2](#).

The

events\_transactions\_current  
table contains current transaction events for each thread.

The table size cannot be figured, and always stores one row for each thread, showing the current status of the thread's most recent monitored transaction event.

The table contains the following columns:

Column	Type	Description
THREAD_ID	bigint(20) unsigned	The thread associated with the event.
EVENT_ID	bigint(20) unsigned	The event id associated with the event.
END_EVENT_ID	bigint(20) unsigned	This column is set to NULL when the event starts and updated to the thread current event number when the event ends.
EVENT_NAME	varchar(128)	The name of the instrument from which the event was collected. This is a NAME value from the setup_instruments table.
STATE	enum('ACTIVE', 'COMMITTED', 'ROLLED BACK')	The current transaction state. The value is ACTIVE (after START TRANSACTION or BEGIN), COMMITTED (after COMMIT), or ROLLED BACK (after ROLLBACK).
TRX_ID	bigint(20) unsigned	Unused.
GTID	varchar(64)	Transaction GTID , using the format DOMAIN-SERVER_ID-SEQUENCE_NO.
XID_FORMAT_ID	int(11)	XA transaction format ID for GTRID and BQUAL values.
XID_GTRID	varchar(130)	XA global transaction ID.
XID_BQUAL	varchar(130)	XA transaction branch qualifier.
XA_STATE	varchar(64)	The state of the XA transaction. The value is ACTIVE (after XA START), IDLE (after XA END), PREPARED (after XA PREPARE), ROLLED BACK (after XA ROLLBACK), or COMMITTED (after XA COMMIT).
SOURCE	varchar(64)	The name of the source file containing the instrumented code that produced the event and the line number in the file at which the instrumentation occurs.
TIMER_START	bigint(20) unsigned	The unit is picoseconds. When event timing started. NULL if event has no timing information.
TIMER_END	bigint(20) unsigned	The unit is picoseconds. When event timing ended. NULL if event has no timing information.
TIMER_WAIT	bigint(20) unsigned	The unit is picoseconds. Event duration. NULL if event has not timing information.
ACCESS_MODE	enum('READ ONLY', 'READ WRITE')	Transaction access mode.
ISOLATION_LEVEL	varchar(64)	Transaction isolation level. One of: REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED, or SERIALIZABLE.
AUTOCOMMIT	enum('YES','NO')	Whether autocommit mode was enabled when the transaction started.
NUMBER_OF_SAVEPOINTS	bigint(20) unsigned	The number of SAVEPOINT statements issued during the transaction.
NUMBER_OF_ROLLBACK_TO_SAVEPOINT	bigint(20) unsigned	The number of ROLLBACK_TO_SAVEPOINT statements issued during the transaction.
NUMBER_OF_RELEASE_SAVEPOINT	bigint(20) unsigned	The number of RELEASE_SAVEPOINT statements issued during the transaction.
OBJECT_INSTANCE_BEGIN	bigint(20) unsigned	Unused.
NESTING_EVENT_ID	bigint(20) unsigned	The EVENT_ID value of the event within which this event is nested.
NESTING_EVENT_TYPE	enum('TRANSACTION', 'STATEMENT', 'STAGE', 'WAIT')	The nesting event type.

## 1.1.2.9.2.1.22 Performance Schema events\_transactions\_history Table

MariaDB starting with [10.5.2](#)

The events\_transactions\_history table was introduced in [MariaDB 10.5.2](#).

The

events\_transactions\_history

table contains the most recent completed transaction events for each thread.

The number of records stored per thread in the table is determined by the [performance\\_schema\\_events\\_transactions\\_history\\_size](#) system variable,

which is autosized on startup.

If adding a completed transaction event would cause the table to exceed this limit, the oldest thread row is discarded.

All of a thread's rows are discarded when the thread ends.

The table contains the following columns:

Column	Type	Description
THREAD_ID	bigint(20) unsigned	The thread associated with the event.
EVENT_ID	bigint(20) unsigned	The event id associated with the event.
END_EVENT_ID	bigint(20) unsigned	This column is set to NULL when the event starts and updated to the thread current event number when the event ends.
EVENT_NAME	varchar(128)	The name of the instrument from which the event was collected. This is a NAME value from the setup_instruments table.
STATE	enum('ACTIVE', 'COMMITTED', 'ROLLED BACK')	The current transaction state. The value is ACTIVE (after START TRANSACTION or BEGIN), COMMITTED (after COMMIT), or ROLLED BACK (after ROLLBACK).
TRX_ID	bigint(20) unsigned	Unused.
GTID	varchar(64)	Transaction <a href="#">GTID</a> , using the format DOMAIN-SERVER_ID-SEQUENCE_NO.
XID_FORMAT_ID	int(11)	XA transaction format ID for GTRID and BQUAL values.
XID_GTRID	varchar(130)	XA global transaction ID.
XID_BQUAL	varchar(130)	XA transaction branch qualifier.
XA_STATE	varchar(64)	The state of the XA transaction. The value is ACTIVE (after XA START), IDLE (after XA END), PREPARED (after XA PREPARE), ROLLED BACK (after XA ROLLBACK), or COMMITTED (after XA COMMIT).
SOURCE	varchar(64)	The name of the source file containing the instrumented code that produced the event and the line number in the file at which the instrumentation occurs.
TIMER_START	bigint(20) unsigned	The unit is picoseconds. When event timing started. NULL if event has no timing information.
TIMER_END	bigint(20) unsigned	The unit is picoseconds. When event timing ended. NULL if event has no timing information.
TIMER_WAIT	bigint(20) unsigned	The unit is picoseconds. Event duration. NULL if event has not timing information.
ACCESS_MODE	enum('READ ONLY', 'READ WRITE')	Transaction access mode.
ISOLATION_LEVEL	varchar(64)	Transaction isolation level. One of: REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED, or SERIALIZABLE.
AUTOCOMMIT	enum('YES', 'NO')	NO
NUMBER_OF_SAVEPOINTS	bigint(20) unsigned	The number of SAVEPOINT statements issued during the transaction.
NUMBER_OF_ROLLBACK_TO_SAVEPOINT	bigint(20) unsigned	The number of ROLLBACK_TO_SAVEPOINT statements issued during the transaction.
NUMBER_OF_RELEASE_SAVEPOINT	bigint(20) unsigned	The number of RELEASE_SAVEPOINT statements issued during the transaction.
OBJECT_INSTANCE_BEGIN	bigint(20) unsigned	Unused.
NESTING_EVENT_ID	bigint(20) unsigned	The EVENT_ID value of the event within which this event is nested.
NESTING_EVENT_TYPE	enum('TRANSACTION', 'STATEMENT', 'STAGE', 'WAIT')	The nesting event type.

## 1.1.2.9.2.1.23 Performance Schema events\_transactions\_history\_long Table

MariaDB starting with [10.5.2](#)

The events\_transactions\_history\_long table was introduced in [MariaDB 10.5.2](#).

The

`events_transactions_history_long`

table contains the most recent completed transaction events that have ended globally, across all threads.

The number of records stored in the table is determined by the `performance_schema_events_transactions_history_long_size` system variable, which is autosized on startup.

If adding a completed transaction would cause the table to exceed this limit, the oldest row, regardless of thread, is discarded.

The table contains the following columns:

Column	Type	Description
THREAD_ID	bigint(20) unsigned	The thread associated with the event.
EVENT_ID	bigint(20) unsigned	The event id associated with the event.
END_EVENT_ID	bigint(20) unsigned	This column is set to NULL when the event starts and updated to the thread current event number when the event ends.
EVENT_NAME	varchar(128)	The name of the instrument from which the event was collected. This is a NAME value from the setup_instruments table.
STATE	enum('ACTIVE', 'COMMITTED', 'ROLLED BACK')	The current transaction state. The value is ACTIVE (after START TRANSACTION or BEGIN), COMMITTED (after COMMIT), or ROLLED BACK (after ROLLBACK).
TRX_ID	bigint(20) unsigned	Unused.
GTID	varchar(64)	Transaction GTID , using the format DOMAIN-SERVER_ID-SEQUENCE_NO.
XID_FORMAT_ID	int(11)	XA transaction format ID for GTRID and BQUAL values.
XID_GTRID	varchar(130)	XA global transaction ID.
XID_BQUAL	varchar(130)	XA transaction branch qualifier.
XA_STATE	varchar(64)	The state of the XA transaction. The value is ACTIVE (after XA START), IDLE (after XA END), PREPARED (after XA PREPARE), ROLLED BACK (after XA ROLLBACK), or COMMITTED (after XA COMMIT).
SOURCE	varchar(64)	The name of the source file containing the instrumented code that produced the event and the line number in the file at which the instrumentation occurs.
TIMER_START	bigint(20) unsigned	The unit is picoseconds. When event timing started. NULL if event has no timing information.
TIMER_END	bigint(20) unsigned	The unit is picoseconds. When event timing ended. NULL if event has no timing information.
TIMER_WAIT	bigint(20) unsigned	The unit is picoseconds. Event duration. NULL if event has not timing information.
ACCESS_MODE	enum('READ ONLY', 'READ WRITE')	Transaction access mode.
ISOLATION_LEVEL	varchar(64)	Transaction isolation level. One of: REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED, or SERIALIZABLE.
AUTOCOMMIT	enum('YES', 'NO')	NO
NUMBER_OF_SAVEPOINTS	bigint(20) unsigned	The number of SAVEPOINT statements issued during the transaction.
NUMBER_OF_ROLLBACK_TO_SAVEPOINT	bigint(20) unsigned	The number of ROLLBACK_TO_SAVEPOINT statements issued during the transaction.
NUMBER_OF_RELEASE_SAVEPOINT	bigint(20) unsigned	The number of RELEASE_SAVEPOINT statements issued during the transaction.
OBJECT_INSTANCE_BEGIN	bigint(20) unsigned	Unused.
NESTING_EVENT_ID	bigint(20) unsigned	The EVENT_ID value of the event within which this event is nested.
NESTING_EVENT_TYPE	enum('TRANSACTION', 'STATEMENT', 'STAGE', 'WAIT')	The nesting event type.

## 1.1.2.9.2.1.24 Performance Schema events\_transactions\_summary\_by\_account\_by\_event\_name Table

MariaDB starting with [10.5.2](#)

The `events_transactions_summary_by_account_by_event_name` table was introduced in [MariaDB 10.5.2](#).

The

`events_transactions_summary_by_account_by_event_name`  
table contains information on transaction events aggregated by account and event name.

The table contains the following columns:

Column	Type	Description
USER	char(32)	User for which summary is generated.
HOST	char(60)	Host for which summary is generated.
EVENT_NAME	varchar(128)	Event name for which summary is generated.
COUNT_STAR	bigint(20) unsigned	The number of summarized events. This value includes all events, whether timed or nontimed.
SUM_TIMER_WAIT	bigint(20) unsigned	The total wait time of the summarized timed events. This value is calculated only for timed events because nontimed events have a wait time of NULL. The same is true for the other <code>xxx_TIMER_WAIT</code> values.
MIN_TIMER_WAIT	bigint(20) unsigned	The minimum wait time of the summarized timed events.
AVG_TIMER_WAIT	bigint(20) unsigned	The average wait time of the summarized timed events.
MAX_TIMER_WAIT	bigint(20) unsigned	The maximum wait time of the summarized timed events.
COUNT_READ_WRITE	bigint(20) unsigned	The total number of only READ/WRITE transaction events.
SUM_TIMER_READ_WRITE	bigint(20) unsigned	The total wait time of only READ/WRITE transaction events.
MIN_TIMER_READ_WRITE	bigint(20) unsigned	The minimum wait time of only READ/WRITE transaction events.
AVG_TIMER_READ_WRITE	bigint(20) unsigned	The average wait time of only READ/WRITE transaction events.
MAX_TIMER_READ_WRITE	bigint(20) unsigned	The maximum wait time of only READ/WRITE transaction events.
COUNT_READ_ONLY	bigint(20) unsigned	The total number of only READ ONLY transaction events.
SUM_TIMER_READ_ONLY	bigint(20) unsigned	The total wait time of only READ ONLY transaction events.
MIN_TIMER_READ_ONLY	bigint(20) unsigned	The minimum wait time of only READ ONLY transaction events.
AVG_TIMER_READ_ONLY	bigint(20) unsigned	The average wait time of only READ ONLY transaction events.
MAX_TIMER_READ_ONLY	bigint(20) unsigned	The maximum wait time of only READ ONLY transaction events.

## 1.1.2.9.2.1.25 Performance Schema `events_transactions_summary_by_host_by_event_name` Table

MariaDB starting with [10.5.2](#)

The `events_transactions_summary_by_host_by_event_name` table was introduced in [MariaDB 10.5.2](#).

The

`events_transactions_summary_by_host_by_event_name`  
table contains information on transaction events aggregated by host and event name.

The table contains the following columns:

Column	Type	Description
HOST	char(60)	Host for which summary is generated.
EVENT_NAME	varchar(128)	Event name for which summary is generated.
COUNT_STAR	bigint(20) unsigned	The number of summarized events. This value includes all events, whether timed or nontimed.
SUM_TIMER_WAIT	bigint(20) unsigned	The total wait time of the summarized timed events. This value is calculated only for timed events because nontimed events have a wait time of NULL. The same is true for the other xxx_TIMER_WAIT values.
MIN_TIMER_WAIT	bigint(20) unsigned	The minimum wait time of the summarized timed events.
AVG_TIMER_WAIT	bigint(20) unsigned	The average wait time of the summarized timed events.
MAX_TIMER_WAIT	bigint(20) unsigned	The maximum wait time of the summarized timed events.
COUNT_READ_WRITE	bigint(20) unsigned	The total number of only READ/WRITE transaction events.
SUM_TIMER_READ_WRITE	bigint(20) unsigned	The total wait time of only READ/WRITE transaction events.
MIN_TIMER_READ_WRITE	bigint(20) unsigned	The minimum wait time of only READ/WRITE transaction events.
AVG_TIMER_READ_WRITE	bigint(20) unsigned	The average wait time of only READ/WRITE transaction events.
MAX_TIMER_READ_WRITE	bigint(20) unsigned	The maximum wait time of only READ/WRITE transaction events.
COUNT_READ_ONLY	bigint(20) unsigned	The total number of only READ ONLY transaction events.
SUM_TIMER_READ_ONLY	bigint(20) unsigned	The total wait time of only READ ONLY transaction events.
MIN_TIMER_READ_ONLY	bigint(20) unsigned	The minimum wait time of only READ ONLY transaction events.
AVG_TIMER_READ_ONLY	bigint(20) unsigned	The average wait time of only READ ONLY transaction events.
MAX_TIMER_READ_ONLY	bigint(20) unsigned	The maximum wait time of only READ ONLY transaction events.

## 1.1.2.9.2.1.26 Performance Schema events\_transactions\_summary\_by\_thread\_by\_event\_name Table

MariaDB starting with [10.5.2](#)

The events\_transactions\_summary\_by\_thread\_by\_event\_name table was introduced in [MariaDB 10.5.2](#).

The

events\_transactions\_summary\_by\_thread\_by\_event\_name  
table contains information on transaction events aggregated by thread and event name.

The table contains the following columns:

Field	Type	Null	Key	Default	Extra
THREAD_ID	bigint(20) unsigned	NO		NULL	
EVENT_NAME	varchar(128)	NO		NULL	
COUNT_STAR	bigint(20) unsigned	NO		NULL	
SUM_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
MIN_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
AVG_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
MAX_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
COUNT_READ_WRITE	bigint(20) unsigned	NO		NULL	
SUM_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
MIN_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
AVG_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
MAX_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
COUNT_READ_ONLY	bigint(20) unsigned	NO		NULL	
SUM_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
MIN_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
AVG_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
MAX_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	

## 1.1.2.9.2.1.27 Performance Schema events\_transactions\_summary\_by\_user\_by\_event\_name Table

MariaDB starting with [10.5.2](#)

The `events_transactions_summary_by_user_by_event_name` table was introduced in [MariaDB 10.5.2](#).

The

`events_transactions_summary_by_user_by_event_name`  
table contains information on transaction events aggregated by user and event name.

The table contains the following columns:

Field	Type	Null	Key	Default	Extra
USER	char(32)	YES		NULL	
EVENT_NAME	varchar(128)	NO		NULL	
COUNT_STAR	bigint(20) unsigned	NO		NULL	
SUM_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
MIN_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
AVG_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
MAX_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
COUNT_READ_WRITE	bigint(20) unsigned	NO		NULL	
SUM_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
MIN_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
AVG_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
MAX_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
COUNT_READ_ONLY	bigint(20) unsigned	NO		NULL	
SUM_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
MIN_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
AVG_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
MAX_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	

## 1.1.2.9.2.1.28 Performance Schema events\_transactions\_summary\_global\_by\_event\_name Table

MariaDB starting with [10.5.2](#)

The `events_transactions_summary_global_by_event_name` table was introduced in [MariaDB 10.5.2](#).

The

`events_transactions_summary_global_by_event_name`  
 table contains information on transaction events aggregated by event name.

The table contains the following columns:

Field	Type	Null	Key	Default	Extra
EVENT_NAME	varchar(128)	NO		NULL	
COUNT_STAR	bigint(20) unsigned	NO		NULL	
SUM_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
MIN_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
AVG_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
MAX_TIMER_WAIT	bigint(20) unsigned	NO		NULL	
COUNT_READ_WRITE	bigint(20) unsigned	NO		NULL	
SUM_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
MIN_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
AVG_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
MAX_TIMER_READ_WRITE	bigint(20) unsigned	NO		NULL	
COUNT_READ_ONLY	bigint(20) unsigned	NO		NULL	
SUM_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
MIN_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
AVG_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	
MAX_TIMER_READ_ONLY	bigint(20) unsigned	NO		NULL	

## 1.1.2.9.2.1.29 Performance Schema events\_waits\_current Table

The

`events_waits_current`  
 table contains the status of a thread's most recently monitored wait event, listing one event per thread.

The table contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with <code>EVENT_ID</code> uniquely identifies the row.
EVENT_ID	Thread's current event number at the start of the event. Together with <code>THREAD_ID</code> uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a <code>NAME</code> from the <code>setup_instruments</code> table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or <code>NULL</code> if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or <code>NULL</code> if the event has not ended or timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or <code>NULL</code> if the event has not ended or timing is not collected.
SPINS	Number of spin rounds for a mutex, or <code>NULL</code> if spin rounds are not used, or spinning is not instrumented.

OBJECT_SCHEMA	Name of the schema that contains the table for table I/O objects, otherwise NULL for file I/O and synchronization objects.
OBJECT_NAME	File name for file I/O objects, table name for table I/O objects, the socket's IP : PORT value for a socket object or NULL for a synchronization object.
INDEX_NAME	Name of the index, PRIMARY for the primary key, or NULL for no index used.
OBJECT_TYPE	FILE for a file object, TABLE or TEMPORARY TABLE for a table object, or NULL for a synchronization object.
OBJECT_INSTANCE_BEGIN	Address in memory of the object.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. Either statement ,, stage or wait .
OPERATION	Operation type, for example read, write or lock
NUMBER_OF_BYTES	Number of bytes that the operation read or wrote, or NULL for table I/O waits.
FLAGS	Reserved for use in the future.

It is possible to empty this table with a

```
TRUNCATE TABLE
statement.
```

The related tables, [events\\_waits\\_history](#) and [events\\_waits\\_history\\_long](#) derive their values from the current events.

## 1.1.2.9.2.1.30 Performance Schema events\_waits\_history Table

The

```
events_waits_history
table by default contains the ten most recent completed wait events per thread. This number can be adjusted by setting the
performance_schema_events_waits_history_size system variable when the server starts up.
```

The table structure is identical to the [events\\_waits\\_current](#) table structure, and contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_ID uniquely identifies the row.

EVENT_ID	Thread's current event number at the start of the event. Together with THREAD_ID uniquely identifies the row.
END_EVENT_ID	NULL when the event starts, set to the thread's current event number at the end of the event.
EVENT_NAME	Event instrument name and a NAME from the setup_instruments table
SOURCE	Name and line number of the source file containing the instrumented code that produced the event.
TIMER_START	Value in picoseconds when the event timing started or NULL if timing is not collected.
TIMER_END	Value in picoseconds when the event timing ended, or NULL if timing is not collected.
TIMER_WAIT	Value in picoseconds of the event's duration or NULL if timing is not collected.
SPINS	Number of spin rounds for a mutex, or NULL if spin rounds are not used, or spinning is not instrumented.
OBJECT_SCHEMA	Name of the schema that contains the table for table I/O objects, otherwise NULL for file I/O and synchronization objects.
OBJECT_NAME	File name for file I/O objects, table name for table I/O objects, the socket's IP:PORT value for a socket object or NULL for a synchronization object.
INDEX_NAME	Name of the index, PRIMARY for the primary key, or NULL for no index used.
OBJECT_TYPE	FILE for a file object, TABLE or TEMPORARY TABLE for a table object, or NULL for a synchronization object.
OBJECT_INSTANCE_BEGIN	Address in memory of the object.
NESTING_EVENT_ID	EVENT_ID of event within which this event nests.
NESTING_EVENT_TYPE	Nesting event type. Either statement ,, stage or wait .
OPERATION	Operation type, for example read, write or lock

NUMBER_OF_BYTES	Number of bytes that the operation read or wrote, or NULL for table I/O waits.
FLAGS	Reserved for use in the future.

It is possible to empty this table with a

```
TRUNCATE TABLE
```

[events\\_waits\\_current](#) and [events\\_waits\\_history\\_long](#) are related tables.

## 1.1.2.9.2.1.31 Performance Schema events\_waits\_summary\_by\_account\_by\_event\_name Table

The [Performance Schema](#)

[events\\_waits\\_summary\\_by\\_account\\_by\\_event\\_name](#)

table contains wait events summarized by account and event name. It contains the following columns:

Column	Description
USER	User. Used together with HOST and EVENT_NAME for grouping events.
HOST	Host. Used together with USER and EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER and HOST for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

The

```
*_TIMER_WAIT
```

columns only calculate results for timed events, as non-timed events have a  
NULL  
wait time.

## Example

```

SELECT * FROM events_waits_summary_by_account_by_event_name\G
...
***** 915. row *****
    USER: NULL
    HOST: NULL
  EVENT_NAME: wait/io/socket/sql/server_tcpip_socket
  COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 916. row *****
    USER: NULL
    HOST: NULL
  EVENT_NAME: wait/io/socket/sql/server_unix_socket
  COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 917. row *****
    USER: NULL
    HOST: NULL
  EVENT_NAME: wait/io/socket/sql/client_connection
  COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 918. row *****
    USER: NULL
    HOST: NULL
  EVENT_NAME: idle
  COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0

```

## 1.1.2.9.2.1.32 Performance Schema events\_waits\_summary\_by\_host\_by\_event\_name Table

The [Performance Schema](#)

`events_waits_summary_by_host_by_event_name`

table contains wait events summarized by host and event name. It contains the following columns:

Column	Description
HOST	Host. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER and HOST for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.

MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
----------------	--

The

```
* _TIMER_WAIT
columns only calculate results for timed events, as non-timed events have a
NULL
wait time.
```

## Example

```
SELECT * FROM events_waits_summary_by_host_by_event_name\G
...
*****
610. row ****
HOST: NULL
EVENT_NAME: wait/io/socket/sql/server_unix_socket
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*****
611. row ****
HOST: NULL
EVENT_NAME: wait/io/socket/sql/client_connection
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*****
612. row ****
HOST: NULL
EVENT_NAME: idle
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

## 1.1.2.9.2.1.33 Performance Schema events\_waits\_summary\_by\_instance Table

The [Performance Schema](#)

`events_waits_summary_by_instance`  
table contains wait events summarized by instance. It contains the following columns:

Column	Description
EVENT_NAME	Event name. Used together with <code>OBJECT_INSTANCE_BEGIN</code> for grouping events.
OBJECT_INSTANCE_BEGIN	If an instrument creates multiple instances, each instance has a unique <code>OBJECT_INSTANCE_BEGIN</code> value to allow for grouping by instance.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.

MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
----------------	--

The

```
*_TIMER_WAIT
columns only calculate results for timed events, as non-timed events have a
NULL
wait time.
```

## Example

```
SELECT * FROM events_waits_summary_by_instance\G
...
***** 202. row *****
EVENT_NAME: wait/io/file/sql/binlog
OBJECT_INSTANCE_BEGIN: 140578961969856
COUNT_STAR: 6
SUM_TIMER_WAIT: 90478331960
MIN_TIMER_WAIT: 263344
AVG_TIMER_WAIT: 15079721848
MAX_TIMER_WAIT: 67760576376
***** 203. row *****
EVENT_NAME: wait/io/file/sql/dbopt
OBJECT_INSTANCE_BEGIN: 140578961970560
COUNT_STAR: 6
SUM_TIMER_WAIT: 39891428472
MIN_TIMER_WAIT: 387168
AVG_TIMER_WAIT: 6648571412
MAX_TIMER_WAIT: 24503293304
***** 204. row *****
EVENT_NAME: wait/io/file/sql/dbopt
OBJECT_INSTANCE_BEGIN: 140578961971264
COUNT_STAR: 6
SUM_TIMER_WAIT: 39902495024
MIN_TIMER_WAIT: 177888
AVG_TIMER_WAIT: 6650415692
MAX_TIMER_WAIT: 21026400404
```

### 1.1.2.9.2.1.34 Performance Schema events\_waits\_summary\_by\_thread\_by\_event\_name Table

The [Performance Schema](#)

`events_waits_summary_by_thread_by_event_name`  
table contains wait events summarized by thread and event name. It contains the following columns:

Column	Description
THREAD_ID	Thread associated with the event. Together with EVENT_NAME uniquely identifies the row.
EVENT_NAME	Event name. Used together with THREAD_ID for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.

MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
----------------	--

The

```
* _TIMER_WAIT
columns only calculate results for timed events, as non-timed events have a
NULL
wait time.
```

## Example

```
SELECT * FROM events_waits_summary_by_thread_by_event_name\G
...
*****6424. row *****
THREAD_ID: 64
EVENT_NAME: wait/io/socket/sql/server_unix_socket
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*****6425. row *****
THREAD_ID: 64
EVENT_NAME: wait/io/socket/sql/client_connection
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*****6426. row *****
THREAD_ID: 64
EVENT_NAME: idle
COUNT_STAR: 73
SUM_TIMER_WAIT: 2200525216200000
MIN_TIMER_WAIT: 300000
AVG_TIMER_WAIT: 30144181000000
MAX_TIMER_WAIT: 491241757300000
```

### 1.1.2.9.2.1.35 Performance Schema events\_waits\_summary\_by\_user\_by\_event\_name Table

The [Performance Schema](#)

`events_waits_summary_by_user_by_event_name`  
table contains wait events summarized by user and event name. It contains the following columns:

Column	Description
USER	User. Used together with EVENT_NAME for grouping events.
EVENT_NAME	Event name. Used together with USER for grouping events.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.

MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
----------------	--

The

```
* _TIMER_WAIT
columns only calculate results for timed events, as non-timed events have a
NULL
wait time.
```

## Example

```
SELECT * FROM events_waits_summary_by_user_by_event_name\G
...
*****916. row *****
USER: NULL
EVENT_NAME: wait/io/socket/sql/server_unix_socket
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*****917. row *****
USER: NULL
EVENT_NAME: wait/io/socket/sql/client_connection
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
*****918. row *****
USER: NULL
EVENT_NAME: idle
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
```

## 1.1.2.9.2.1.36 Performance Schema events\_waits\_summary\_global\_by\_event\_name Table

The [Performance Schema](#)

`events_waits_summary_global_by_event_name`  
table contains wait events summarized by event name. It contains the following columns:

Column	Description
EVENT_NAME	Event name.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

The

```
* _TIMER_WAIT
columns only calculate results for timed events, as non-timed events have a
NULL
wait time.
```

## Example

```
SELECT * FROM events_waits_summary_global_by_event_name\G
...
***** 303. row *****
EVENT_NAME: wait/io/socket/sql/server_tcpip_socket
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 304. row *****
EVENT_NAME: wait/io/socket/sql/server_unix_socket
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 305. row *****
EVENT_NAME: wait/io/socket/sql/client_connection
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 306. row *****
EVENT_NAME: idle
COUNT_STAR: 265
SUM_TIMER_WAIT: 4686112518100000
MIN_TIMER_WAIT: 1000000
AVG_TIMER_WAIT: 17683443400000
MAX_TIMER_WAIT: 491241757300000
```

## 1.1.2.9.2.1.37 Performance Schema file\_instances Table

### Description

The

`file_instances`

table lists instances of instruments seen by the Performance Schema when executing file I/O instrumentation, and the associated files. Only files that have been opened, and that have not been deleted, will be listed in the table.

The `performance_schema_max_file_instances` system variable specifies the maximum number of instrumented file objects.

Column	Description
<code>FILE_NAME</code>	File name.
<code>EVENT_NAME</code>	Instrument name associated with the file.
<code>OPEN_COUNT</code>	Open handles on the file. A value of greater than zero means that the file is currently open.

## Example

```
SELECT * FROM performance_schema.file_instances WHERE OPEN_COUNT>0;
```

FILE_NAME	EVENT_NAME	OPEN_COUNT
/var/log/mysql/mariadb-bin.index	wait/io/file/sql/binlog_index	1
/var/lib/mysql/ibdata1	wait/io/file/innodb/innodb_data_file	2
/var/lib/mysql/ib_logfile0	wait/io/file/innodb/innodb_log_file	2
/var/lib/mysql/ib_logfile1	wait/io/file/innodb/innodb_log_file	2
/var/lib/mysql/mysql/gtid_slave_pos.ibd	wait/io/file/innodb/innodb_data_file	3
/var/lib/mysql/mysql/innodb_index_stats.ibd	wait/io/file/innodb/innodb_data_file	3
/var/lib/mysql/mysql/innodb_table_stats.ibd	wait/io/file/innodb/innodb_data_file	3
...		

## 1.1.2.9.2.1.38 Performance Schema file\_summary\_by\_event\_name Table

The [Performance Schema](#)

`file_summary_by_event_name`

table contains file events summarized by event name. As of [MariaDB 10.0](#), it contains the following columns:

Column	Description
EVENT_NAME	Event name.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, including FGETS ,FGETC ,FREAD , and READ
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
SUM_NUMBER_OF_BYTES_READ	Bytes read by read operations.

COUNT_WRITE	Number of all write operations, including FPUTS ,, FPUTC ,, FPRINTF ,, VFPRINTF ,, FWRITE , and PWRITE  .
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
SUM_NUMBER_OF_BYTES_WRITE	Bytes written by write operations.
COUNT_MISC	Number of all miscellaneous operations not counted above, including CREATE ,, DELETE ,, OPEN ,, CLOSE ,, STREAM_OPEN ,, STREAM_CLOSE ,, SEEK ,, TELL ,, FLUSH ,, STAT ,, FSTAT ,, CHSIZE ,, RENAME , and SYNC  .
SUM_TIMER_MISC	Total wait time of all miscellaneous operations that are timed.
MIN_TIMER_MISC	Minimum wait time of all miscellaneous operations that are timed.
AVG_TIMER_MISC	Average wait time of all miscellaneous operations that are timed.

MAX_TIMER_MISC	Maximum wait time of all miscellaneous operations that are timed.
----------------	---

Before MariaDB 10, the table contained only the

```
EVENT_NAME
,
COUNT_READ
,
COUNT_WRITE
,
SUM_NUMBER_OF_BYTES_READ
and
SUM_NUMBER_OF_BYTES_WRITE
columns.
```

I/O operations can be avoided by caching, in which case they will not be recorded in this table.

You can [TRUNCATE](#) the table, which will reset all counters to zero.

## Example

```
SELECT * FROM file_summary_by_event_name\G
...
***** 49. row *****
    EVENT_NAME: wait/io/file/aria/MAD
    COUNT_STAR: 60
    SUM_TIMER_WAIT: 397234368
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 6620224
    MAX_TIMER_WAIT: 16808672
        COUNT_READ: 0
    SUM_TIMER_READ: 0
    MIN_TIMER_READ: 0
    AVG_TIMER_READ: 0
    MAX_TIMER_READ: 0
    SUM_NUMBER_OF_BYTES_READ: 0
        COUNT_WRITE: 0
    SUM_TIMER_WRITE: 0
    MIN_TIMER_WRITE: 0
    AVG_TIMER_WRITE: 0
    MAX_TIMER_WRITE: 0
    SUM_NUMBER_OF_BYTES_WRITE: 0
        COUNT_MISC: 60
    SUM_TIMER_MISC: 397234368
    MIN_TIMER_MISC: 0
    AVG_TIMER_MISC: 6620224
    MAX_TIMER_MISC: 16808672
***** 50. row *****
    EVENT_NAME: wait/io/file/aria/control
    COUNT_STAR: 3
    SUM_TIMER_WAIT: 24055778544
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 8018592848
    MAX_TIMER_WAIT: 24027262400
        COUNT_READ: 1
    SUM_TIMER_READ: 24027262400
    MIN_TIMER_READ: 0
    AVG_TIMER_READ: 24027262400
    MAX_TIMER_READ: 24027262400
    SUM_NUMBER_OF_BYTES_READ: 52
        COUNT_WRITE: 0
    SUM_TIMER_WRITE: 0
    MIN_TIMER_WRITE: 0
    AVG_TIMER_WRITE: 0
    MAX_TIMER_WRITE: 0
    SUM_NUMBER_OF_BYTES_WRITE: 0
        COUNT_MISC: 2
    SUM_TIMER_MISC: 28516144
    MIN_TIMER_MISC: 0
    AVG_TIMER_MISC: 14258072
    MAX_TIMER_MISC: 27262208
```

## 1.1.2.9.2.1.39 Performance Schema file\_summary\_by\_instance Table

The [Performance Schema](#)

`file_summary_by_instance`

table contains file events summarized by instance. As of [MariaDB 10.0](#), it contains the following columns:

Column	Description
<code>FILE_NAME</code>	File name.
<code>EVENT_NAME</code>	Event name.
<code>OBJECT_INSTANCE_BEGIN</code>	Address in memory. Together with <code>FILE_NAME</code> and <code>EVENT_NAME</code> uniquely identifies a row.
<code>COUNT_STAR</code>	Number of summarized events
<code>SUM_TIMER_WAIT</code>	Total wait time of the summarized events that are timed.
<code>MIN_TIMER_WAIT</code>	Minimum wait time of the summarized events that are timed.
<code>AVG_TIMER_WAIT</code>	Average wait time of the summarized events that are timed.
<code>MAX_TIMER_WAIT</code>	Maximum wait time of the summarized events that are timed.
<code>COUNT_READ</code>	Number of all read operations, including <code>FGETS</code> , <code>FGETC</code> , <code>FREAD</code> , and <code>READ</code> .
<code>SUM_TIMER_READ</code>	Total wait time of all read operations that are timed.
<code>MIN_TIMER_READ</code>	Minimum wait time of all read operations that are timed.
<code>AVG_TIMER_READ</code>	Average wait time of all read operations that are timed.
<code>MAX_TIMER_READ</code>	Maximum wait time of all read operations that are timed.
<code>SUM_NUMBER_OF_BYTES_READ</code>	Bytes read by read operations.

COUNT_WRITE	Number of all write operations, including FPUTS ,, FPUTC ,, FPRINTF ,, VFPRINTF ,, FWRITE , and PWRITE  .
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
SUM_NUMBER_OF_BYTES_WRITE	Bytes written by write operations.
COUNT_MISC	Number of all miscellaneous operations not counted above, including CREATE ,, DELETE ,, OPEN ,, CLOSE ,, STREAM_OPEN ,, STREAM_CLOSE ,, SEEK ,, TELL ,, FLUSH ,, STAT ,, FSTAT ,, CHSIZE ,, RENAME , and SYNC  .
SUM_TIMER_MISC	Total wait time of all miscellaneous operations that are timed.
MIN_TIMER_MISC	Minimum wait time of all miscellaneous operations that are timed.
AVG_TIMER_MISC	Average wait time of all miscellaneous operations that are timed.

MAX_TIMER_MISC	Maximum wait time of all miscellaneous operations that are timed.
----------------	---

Before MariaDB 10, the table contained only the

```
FILE_NAME  
,  
EVENT_NAME  
,  
COUNT_READ  
,  
COUNT_WRITE  
,  
SUM_NUMBER_OF_BYTES_READ  
and  
SUM_NUMBER_OF_BYTES_WRITE  
columns.
```

I/O operations can be avoided by caching, in which case they will not be recorded in this table.

You can [TRUNCATE](#) the table, which will reset all counters to zero.

## Example

```

SELECT * FROM file_summary_by_instance\G
...
***** 204. row *****
    FILE_NAME: /var/lib/mysql/test/db.opt
    EVENT_NAME: wait/io/file/sql/dbopt
OBJECT_INSTANCE_BEGIN: 140578961971264
    COUNT_STAR: 6
    SUM_TIMER_WAIT: 39902495024
    MIN_TIMER_WAIT: 177888
    AVG_TIMER_WAIT: 6650415692
    MAX_TIMER_WAIT: 21026400404
    COUNT_READ: 1
    SUM_TIMER_READ: 21026400404
    MIN_TIMER_READ: 21026400404
    AVG_TIMER_READ: 21026400404
    MAX_TIMER_READ: 21026400404
SUM_NUMBER_OF_BYTES_READ: 65
    COUNT_WRITE: 0
    SUM_TIMER_WRITE: 0
    MIN_TIMER_WRITE: 0
    AVG_TIMER_WRITE: 0
    MAX_TIMER_WRITE: 0
SUM_NUMBER_OF_BYTES_WRITE: 0
    COUNT_MISC: 5
    SUM_TIMER_MISC: 18876094620
    MIN_TIMER_MISC: 177888
    AVG_TIMER_MISC: 3775218924
    MAX_TIMER_MISC: 18864558060
***** 205. row *****
    FILE_NAME: /var/log/mysql/mariadb-bin.000157
    EVENT_NAME: wait/io/file/sql/binlog
OBJECT_INSTANCE_BEGIN: 140578961971968
    COUNT_STAR: 6
    SUM_TIMER_WAIT: 73985877680
    MIN_TIMER_WAIT: 251136
    AVG_TIMER_WAIT: 12330979468
    MAX_TIMER_WAIT: 73846656340
    COUNT_READ: 0
    SUM_TIMER_READ: 0
    MIN_TIMER_READ: 0
    AVG_TIMER_READ: 0
    MAX_TIMER_READ: 0
SUM_NUMBER_OF_BYTES_READ: 0
    COUNT_WRITE: 2
    SUM_TIMER_WRITE: 62583004
    MIN_TIMER_WRITE: 27630192
    AVG_TIMER_WRITE: 31291284
    MAX_TIMER_WRITE: 34952812
SUM_NUMBER_OF_BYTES_WRITE: 369
    COUNT_MISC: 4
    SUM_TIMER_MISC: 73923294676
    MIN_TIMER_MISC: 251136
    AVG_TIMER_MISC: 18480823560
    MAX_TIMER_MISC: 73846656340

```

## 1.1.2.9.2.1.40 Performance Schema global\_status Table

MariaDB starting with [10.5.2](#)

The

`global_status`  
table was added in [MariaDB 10.5.2](#).

The

`global_status`

table contains a list of status variables and their global values. The table only stores status variable statistics for threads which are instrumented, and does not collect statistics for

`Com_xxx`

variables.

The table contains the following columns:

Column	Description
--------	-------------

VARIABLE_NAME	The global status variable name.
VARIABLE_VALUE	The global status variable value.

`TRUNCATE TABLE` resets global status variables, including thread, account, host, and user status, but not those that are never reset by the server.

## 1.1.2.9.2.1.41 Performance Schema hosts Table

### Description

The  
 hosts  
 table contains a row for each host used by clients to connect to the server, containing current and total connections.  
 The size is determined by the `performance_schema_hosts_size` system variable, which, if set to zero, will disable connection statistics in the hosts table.  
 It contains the following columns:

Column	Description
HOST	Host name used by the client to connect, NULL for internal threads or user sessions that failed to authenticate.
CURRENT_CONNECTIONS	Current number of the host's connections.
TOTAL_CONNECTIONS	Total number of the host's connections

### Example

```
SELECT * FROM hosts;
+-----+-----+-----+
| HOST      | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+-----+-----+-----+
| localhost |              1 |          45 |
| NULL      |             20 |          23 |
+-----+-----+-----+
```

## 1.1.2.9.2.1.42 Performance Schema memory\_summary\_by\_account\_by\_event\_name Table

MariaDB starting with 10.5.2

The `memory_summary_by_account_by_event_name` table was introduced in MariaDB 10.5.2 .

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- `memory_summary_by_account_by_event_name`
- `memory_summary_by_host_by_event_name`
- `memory_summary_by_thread_by_event_name`
- `memory_summary_by_user_by_event_name`
- `memory_global_by_event_name`

The  
`memory_summary_by_account_by_event_name`

table contains memory usage statistics aggregated by account and event.

The table contains the following columns:

Field	Type	Null	Default	Description
USER	char(32)	YES	NULL	User portion of the account.
HOST	char(60)	YES	NULL	Host portion of the account.

EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

## 1.1.2.9.2.1.43 Performance Schema memory\_summary\_by\_host\_by\_event\_name Table

MariaDB starting with [10.5.2](#)

The `memory_summary_by_host_by_event_name` table was introduced in [MariaDB 10.5.2](#).

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- [memory\\_summary\\_by\\_account\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_host\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_thread\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_user\\_by\\_event\\_name](#)
- [memory\\_global\\_by\\_event\\_name](#)

The

```
memory_summary_by_host_by_event_name
table contains memory usage statistics aggregated by host and event.
```

The table contains the following columns:

Field	Type	Null	Default	Description
HOST	char(60)	YES	NULL	Host portion of the account.
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.

CURRENT_NUMBEROFBYTESUSED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

## 1.1.2.9.2.1.44 Performance Schema memory\_summary\_by\_thread\_by\_event\_name Table

MariaDB starting with [10.5.2](#)

The memory\_summary\_by\_thread\_by\_event\_name table was introduced in [MariaDB 10.5.2](#).

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- [memory\\_summary\\_by\\_account\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_host\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_thread\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_user\\_by\\_event\\_name](#)
- [memory\\_global\\_by\\_event\\_name](#)

The

```
memory_summary_by_thread_by_event_name
table contains memory usage statistics aggregated by thread and event.
```

The table contains the following columns:

Field	Type	Null	Default	Description
THREAD_ID	bigint(20) unsigned	NO	NULL	Thread id.
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

## 1.1.2.9.2.1.45 Performance Schema memory\_summary\_by\_user\_by\_event\_name Table

MariaDB starting with [10.5.2](#)

The memory\_summary\_by\_user\_by\_event\_name table was introduced in [MariaDB 10.5.2](#).

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- [memory\\_summary\\_by\\_account\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_host\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_thread\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_user\\_by\\_event\\_name](#)
- [memory\\_global\\_by\\_event\\_name](#)

The

### `memory_summary_by_user_by_event_name`

table contains memory usage statistics aggregated by user and event.

The table contains the following columns:

Field	Type	Null	Default	Description
USER	char(32)	YES	NULL	User portion of the account.
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).
HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

## 1.1.2.9.2.1.46 Performance Schema `memory_summary_global_by_event_name` Table

MariaDB starting with [10.5.2](#)

The `memory_summary_global_by_event_name` table was introduced in [MariaDB 10.5.2](#).

There are five memory summary tables in the Performance Schema that share a number of fields in common. These include:

- [memory\\_summary\\_by\\_account\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_host\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_thread\\_by\\_event\\_name](#)
- [memory\\_summary\\_by\\_user\\_by\\_event\\_name](#)
- [memory\\_global\\_by\\_event\\_name](#)

The

### `memory_summary_global_by_event_name`

table contains memory usage statistics aggregated by event and event.

The table contains the following columns:

Field	Type	Null	Default	Description
EVENT_NAME	varchar(128)	NO	NULL	Event name.
COUNT_ALLOC	bigint(20) unsigned	NO	NULL	Total number of allocations to memory.
COUNT_FREE	bigint(20) unsigned	NO	NULL	Total number of attempts to free the allocated memory.
SUM_NUMBER_OF_BYTES_ALLOC	bigint(20) unsigned	NO	NULL	Total number of bytes allocated.
SUM_NUMBER_OF_BYTES_FREE	bigint(20) unsigned	NO	NULL	Total number of bytes freed
LOW_COUNT_USED	bigint(20)	NO	NULL	Lowest number of allocated blocks (lowest value of CURRENT_COUNT_USED).
CURRENT_COUNT_USED	bigint(20)	NO	NULL	Currently allocated blocks that have not been freed (COUNT_ALLOC minus COUNT_FREE).

HIGH_COUNT_USED	bigint(20)	NO	NULL	Highest number of allocated blocks (highest value of CURRENT_COUNT_USED).
LOW_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Lowest number of bytes used.
CURRENT_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Current number of bytes used (total allocated minus total freed).
HIGH_NUMBER_OF_BYTES_USED	bigint(20)	NO	NULL	Highest number of bytes used.

## Example

Seeing what memory was most often allocated for:

```
SELECT * FROM memory_summary_global_by_event_name
  ORDER BY count_alloc DESC LIMIT 1\G
***** 1. row *****
  EVENT_NAME: memory/sql/QUICK_RANGE_SELECT::alloc
  COUNT_ALLOC: 147976
  COUNT_FREE: 147976
  SUM_NUMBER_OF_BYTES_ALLOC: 600190656
  SUM_NUMBER_OF_BYTES_FREE: 600190656
  LOW_COUNT_USED: 0
  CURRENT_COUNT_USED: 0
  HIGH_COUNT_USED: 68
  LOW_NUMBER_OF_BYTES_USED: 0
  CURRENT_NUMBER_OF_BYTES_USED: 0
  HIGH_NUMBER_OF_BYTES_USED: 275808
```

## 1.1.2.9.2.1.47 Performance Schema metadata\_locks Table

MariaDB starting with [10.5.2](#)

The metadata\_locks table was introduced in [MariaDB 10.5.2](#).

The

`metadata_locks`  
table contains [metadata lock](#) information.

To enable metadata lock instrumention, at runtime:

```
UPDATE performance_schema.setup_instruments SET enabled='YES', timed='YES'
  WHERE name LIKE 'wait/lock/metadata%';
```

or in the [configuration file](#):

```
performance-schema-instrument='wait/lock/metadata/sql/mdl=ON'
```

The table is by default autosized, but the size can be configured with the [performance\\_schema\\_max\\_metadata\\_locks](#) system variable.

The table is read-only, and [TRUNCATE TABLE](#) cannot be used to empty the table.

The table contains the following columns:

Field	Type	Null	Default	Description
-------	------	------	---------	-------------

OBJECT_TYPE	varchar(64)	NO	NULL	Object type. One of BACKUP , COMMIT , EVENT , FUNCTION , GLOBAL , LOCKING SERVICE , PROCEDURE , SCHEMA , TABLE , TABLESPACE , TRIGGER (unused) or USER LEVEL LOCK .
OBJECT_SCHEMA	varchar(64)	YES	NULL	Object schema.
OBJECT_NAME	varchar(64)	YES	NULL	Object name.
OBJECT_INSTANCE_BEGIN	bigint(20) unsigned	NO	NULL	Address in memory of the instrumented object.
LOCK_TYPE	varchar(32)	NO	NULL	Lock type. One of BACKUP_FTWR1 , BACKUP_START , BACKUP_TRANS_DML , EXCLUSIVE , INTENTION_EXCLUSIVE , SHARED , SHARED_HIGH_PRIO , SHARED_NO_READ_WRITE , SHARED_NO_WRITE , SHARED_READ , SHARED_UPGRADABLE or SHARED_WRITE .
LOCK_DURATION	varchar(32)	NO	NULL	Lock duration. One of EXPLICIT (locks released by explicit action, for example a global lock acquired with <a href="#">FLUSH TABLES WITH READ LOCK</a> ), STATEMENT (locks implicitly released at statement end) or TRANSACTION (locks implicitly released at transaction end).

LOCK_STATUS	varchar(32)	NO	NULL	Lock status. One of GRANTED , KILLED , PENDING , POST_RELEASE_NOTIFY , PRE_ACQUIRE_NOTIFY , TIMEOUT or VICTIM .
SOURCE	varchar(64)	YES	NULL	Source file containing the instrumented code that produced the event, as well as the line number where the instrumentation occurred. This allows one to examine the source code involved.
OWNER_THREAD_ID	bigint(20) unsigned	YES	NULL	Thread that requested the lock.
OWNER_EVENT_ID	bigint(20) unsigned	YES	NULL	Event that requested the lock.

## 1.1.2.9.2.1.48 Performance Schema mutex\_instances Table

### Description

The

```
mutex_instances
```

table lists all mutexes that the Performance Schema seeing while the server is executing.

A mutex is a code mechanism for ensuring that threads can only access resources one at a time. A second thread attempting to access a resource will find it protected by a mutex, and will wait for it to be unlocked.

The [performance\\_schema\\_max\\_mutex\\_instances](#) system variable specifies the maximum number of instrumented mutex instances.

Column	Description
NAME	Instrument name associated with the mutex.
OBJECT_INSTANCE_BEGIN	Memory address of the instrumented mutex.
LOCKED_BY_THREAD_ID	The THREAD_ID of the locking thread if a thread has a mutex locked, otherwise NULL .

## 1.1.2.9.2.1.49 Performance Schema objects\_summary\_global\_by\_type Table

It aggregates object wait events, and contains the following columns:

Column	Description
OBJECT_TYPE	Groups records together with OBJECT_SCHEMA and OBJECT_NAME .

OBJECT_SCHEMA	Groups records together with OBJECT_TYPE and OBJECT_NAME
OBJECT_NAME	Groups records together with OBJECT_SCHEMA and OBJECT_TYPE
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

You can [TRUNCATE](#) the table, which will reset all counters to zero.

## Example

```
SELECT * FROM objects_summary_global_by_type\G
...
***** 101. row *****
OBJECT_TYPE: TABLE
OBJECT_SCHEMA: test
OBJECT_NAME: v
COUNT_STAR: 0
SUM_TIMER_WAIT: 0
MIN_TIMER_WAIT: 0
AVG_TIMER_WAIT: 0
MAX_TIMER_WAIT: 0
***** 102. row *****
OBJECT_TYPE: TABLE
OBJECT_SCHEMA: test
OBJECT_NAME: xx2
COUNT_STAR: 2
SUM_TIMER_WAIT: 1621920
MIN_TIMER_WAIT: 481344
AVG_TIMER_WAIT: 810960
MAX_TIMER_WAIT: 1140576
```

## 1.1.2.9.2.1.50 Performance Schema performance\_timers Table

### Description

The

performance\_timers  
table lists available event timers.

It contains the following columns:

Column	Description
--------	-------------

TIMER_NAME	Time name, used in the <a href="#">setup_timers</a> table.
TIMER_FREQUENCY	Number of timer units per second. Dependent on the processor speed.
TIMER_RESOLUTION	Number of timer units by which timed values increase each time.
TIMER_OVERHEAD	Minimum timer overhead, determined during initialization by calling the timer 20 times and selecting the smallest value. Total overhead will be at least double this, as the timer is called at the beginning and end of each timed event.

Any

NULL

values indicate that that particular timer is not available on your platform. Any timer names with a non-NULL value can be used in the [setup\\_timers](#) table.

## Example

```
SELECT * FROM performance_timers;
+-----+-----+-----+-----+
| TIMER_NAME | TIMER_FREQUENCY | TIMER_RESOLUTION | TIMER_OVERHEAD |
+-----+-----+-----+-----+
| CYCLE      | 2293651741 | 1 | 28 |
| NANOSECOND | 1000000000 | 1 | 48 |
| MICROSECOND | 1000000 | 1 | 52 |
| MILLISECOND | 1000 | 1000 | 9223372036854775807 |
| TICK        | 106 | 1 | 496 |
+-----+-----+-----+-----+
```

## 1.1.2.9.2.1.51 Performance Schema prepared\_statements\_instances Table

MariaDB starting with [10.5.2](#)

The `prepared_statements_instances` table was introduced in [MariaDB 10.5.2](#).

The

```
prepared_statements_instances
table contains aggregated statistics of prepared statements.
```

The maximum number of rows in the table is determined by the [performance\\_schema\\_max\\_prepared\\_statement\\_instances](#) system variable, which is by default autosized on startup.

The table contains the following columns:

Field	Type	Null	Key	Default	Extra
OBJECT_INSTANCE_BEGIN	bigint(20) unsigned	NO		NULL	
STATEMENT_ID	bigint(20) unsigned	NO		NULL	
STATEMENT_NAME	varchar(64)	YES		NULL	
SQL_TEXT	longtext	NO		NULL	
OWNER_THREAD_ID	bigint(20) unsigned	NO		NULL	
OWNER_EVENT_ID	bigint(20) unsigned	NO		NULL	
OWNER_OBJECT_TYPE	enum('EVENT','FUNCTION','PROCEDURE','TABLE','TRIGGER')	YES		NULL	
OWNER_OBJECT_SCHEMA	varchar(64)	YES		NULL	
OWNER_OBJECT_NAME	varchar(64)	YES		NULL	
TIMER_PREPARE	bigint(20) unsigned	NO		NULL	
COUNT_REPREPARE	bigint(20) unsigned	NO		NULL	
COUNT_EXECUTE	bigint(20) unsigned	NO		NULL	
SUM_TIMER_EXECUTE	bigint(20) unsigned	NO		NULL	
MIN_TIMER_EXECUTE	bigint(20) unsigned	NO		NULL	
AVG_TIMER_EXECUTE	bigint(20) unsigned	NO		NULL	
MAX_TIMER_EXECUTE	bigint(20) unsigned	NO		NULL	
SUM_LOCK_TIME	bigint(20) unsigned	NO		NULL	
SUM_ERRORS	bigint(20) unsigned	NO		NULL	
SUM_WARNINGS	bigint(20) unsigned	NO		NULL	
SUM_ROWS_AFFECTED	bigint(20) unsigned	NO		NULL	
SUM_ROWS_SENT	bigint(20) unsigned	NO		NULL	
SUM_ROWS_EXAMINED	bigint(20) unsigned	NO		NULL	
SUM_CREATED_TMP_DISK_TABLES	bigint(20) unsigned	NO		NULL	
SUM_CREATED_TMP_TABLES	bigint(20) unsigned	NO		NULL	
SUM_SELECT_FULL_JOIN	bigint(20) unsigned	NO		NULL	
SUM_SELECT_FULL_RANGE_JOIN	bigint(20) unsigned	NO		NULL	
SUM_SELECT_RANGE	bigint(20) unsigned	NO		NULL	
SUM_SELECT_RANGE_CHECK	bigint(20) unsigned	NO		NULL	
SUM_SELECT_SCAN	bigint(20) unsigned	NO		NULL	
SUM_SORT_MERGE_PASSES	bigint(20) unsigned	NO		NULL	
SUM_SORT_RANGE	bigint(20) unsigned	NO		NULL	
SUM_SORT_ROWS	bigint(20) unsigned	NO		NULL	
SUM_SORT_SCAN	bigint(20) unsigned	NO		NULL	
SUM_NO_INDEX_USED	bigint(20) unsigned	NO		NULL	
SUM_NO_GOOD_INDEX_USED	bigint(20) unsigned	NO		NULL	

## 1.1.2.9.2.1.52 Performance Schema replication\_applier\_configuration Table

MariaDB starting with [10.5.2](#)

The

```
replication_applier_configuration
table, along with many other new Performance Schema tables, was added in MariaDB 10.5.2.
```

The [Performance Schema](#) replication\_applier\_configuration table contains configuration settings affecting replica transactions.

It contains the following fields.

Field	Type	Null	Description
CHANNEL_NAME	char(64)	NO	Replication channel name.
DESIRED_DELAY	int(11)	NO	Target number of seconds the replica should be delayed to the master.

## 1.1.2.9.2.1.53 Performance Schema replication\_applier\_status Table

MariaDB starting with [10.5.2](#)

The

```
replication_applier_status
table, along with many other new Performance Schema tables, was added in MariaDB 10.5.2.
```

The [Performance Schema](#) replication\_applier\_status table contains information about the general transaction execution status on the replica.

It contains the following fields.

Field	Type	Null	Description
CHANNEL_NAME	char(64)	NO	The replication channel name.
SERVICE_STATE	enum('ON','OFF')	NO	Shows ON when the replication channel's applier threads are active or idle, OFF means that the applier threads are not active.
REMAINING_DELAY	int(10) unsigned	YES	Seconds the replica needs to wait to reach the desired delay from master.
COUNT_TRANSACTIONS_RETRIES	bigint(20) unsigned	NO	The number of retries that were made because the replication SQL thread failed to apply a transaction.

## 1.1.2.9.2.1.54 Performance Schema replication\_applier\_status\_by\_coordinator Table

MariaDB starting with [10.5.2](#)

The

```
replication_applier_status_by_coordinator
table was added in MariaDB 10.5.2 .
```

The [Performance Schema](#) replication\_applier\_status\_by\_coordinator table displays the status of the coordinator thread used in multi-threaded replicas to manage multiple worker threads.

It contains the following fields.

Column	Type	Null	Description
CHANNEL_NAME	varchar(256)	NO	Replication channel name.
THREAD_ID	bigint(20) unsigned	YES	The SQL/coordinator thread ID.
SERVICE_STATE	enum('ON','OFF')	NO	ON (thread exists and is active or idle) or OFF (thread no longer exists).
LAST_ERROR_NUMBER	int(11)	NO	Last error number that caused the SQL/coordinator thread to stop.
LAST_ERROR_MESSAGE	varchar(1024)	NO	Last error message that caused the SQL/coordinator thread to stop.
LAST_ERROR_TIMESTAMP	timestamp	NO	Timestamp that shows when the most recent SQL/coordinator error occurred.
LAST_SEEN_TRANSACTION	char(57)	NO	The transaction the worker has last seen.
LAST_TRANS_RETRY_COUNT	int(11)	NO	Total number of retries attempted by last transaction.

## 1.1.2.9.2.1.55 Performance Schema replication\_applier\_status\_by\_worker Table

MariaDB starting with [10.6.0](#)

The

```
replication_applier_status_by_worker
table was added in MariaDB 10.6.0 .
```

The [Performance Schema](#) replication\_applier\_status\_by\_worker table displays replica worker thread specific information.

It contains the following fields.

Column	Description
CHANNEL_NAME	Name of replication channel through which the transaction is received.
THREAD_ID	Thread_Id as displayed in the <a href="#">performance_schema.threads</a> table for thread with name 'thread/sql/rpl_parallel_thread'. THREAD_ID will be NULL when worker threads are stopped due to error/force stop.
SERVICE_STATE	Whether or not the thread is running.
LAST_SEEN_TRANSACTION	Last GTID executed by worker
LAST_ERROR_NUMBER	Last Error that occurred on a particular worker.
LAST_ERROR_MESSAGE	Last error specific message.
LAST_ERROR_TIMESTAMP	Time stamp of last error.
WORKER_IDLE_TIME	Total idle time in seconds that the worker thread has spent waiting for work from SQL thread.
LAST_TRANS_RETRY_COUNT	Total number of retries attempted by last transaction.

## 1.1.2.9.2.1.56 Performance Schema replication\_connection\_configuration Table

MariaDB starting with [10.5.2](#)

The

`replication_connection_configuration`  
table was added in [MariaDB 10.6.0](#).

The [Performance Schema](#) `replication_connection_configuration` table displays replica's configuration settings used for connecting to the primary.

It contains the following fields.

Column	Type	Null	Description
CHANNEL_NAME	varchar(256)	NO	The replication channel used.
HOST	char(60)	NO	The host name of the source that the replica is connected to.
PORT	int(11)	NO	The port used to connect to the source.
USER	char(32)	NO	The user name of the replication user account used to connect to the source.
USING_GTID	enum('NO', 'CURRENT_POS', 'SLAVE_POS')	NO	Whether replication is using GTIDs or not.
SSL_ALLOWED	enum('YES', 'NO', 'IGNORED')	NO	Whether SSL is allowed for the replica connection.
SSL_CA_FILE	varchar(512)	NO	Path to the file that contains one or more certificates for trusted Certificate Authorities (CA) to use for TLS.
SSL_CA_PATH	varchar(512)	NO	Path to a directory that contains one or more PEM files that contain X509 certificates for a trusted Certificate Authority (CA) to use for TLS.
SSL_CERTIFICATE	varchar(512)	NO	Path to the certificate used to authenticate the master.
SSL_CIPHER	varchar(512)	NO	Which cipher is used for encryption.
SSL_KEY	varchar(512)	NO	Path to the private key used for TLS.
SSL_VERIFY_SERVER_CERTIFICATE	enum('YES', 'NO')	NO	Whether the server certificate is verified as part of the SSL connection.
SSL_CRL_FILE	varchar(255)	NO	Path to the PEM file containing one or more revoked X.509 certificates.
SSL_CRL_PATH	varchar(255)	NO	PATH to a folder containing PEM files containing one or more revoked X.509 certificates.
CONNECTION_RETRY_INTERVAL	int(11)	NO	The number of seconds between connect retries.
CONNECTION_RETRY_COUNT	bigint(20) unsigned	NO	The number of times the replica can attempt to reconnect to the source in the event of a lost connection.
HEARTBEAT_INTERVAL	double(10,3) unsigned	NO	Number of seconds after which a heartbeat will be sent.
IGNORE_SERVER_IDS	longtext	NO	Binary log events from servers (ids) to ignore.
REPL_DO_DOMAIN_IDS	longtext	NO	Only apply binary logs from these domain ids.
REPL_IGNORE_DOMAIN_IDS	longtext	NO	Binary log events from domains to ignore.

## 1.1.2.9.2.1.57 Performance Schema rwlock\_instances Table

The

`rwlock_instances`

table lists all read write lock (rwlock) instances that the Performance Schema sees while the server is executing. A read write is a mechanism for ensuring threads can either share access to common resources, or have exclusive access.

The [performance\\_schema\\_max\\_rwlock\\_instances](#) system variable specifies the maximum number of instrumented rwlock objects.

The

`rwlock_instances`

table contains the following columns:

Column	Description
--------	-------------

NAME	Instrument name associated with the read write lock
OBJECT_INSTANCE_BEGIN	Address in memory of the instrumented lock
WRITE_LOCKED_BY_THREAD_ID	THREAD_ID of the locking thread if locked in write (exclusive) mode, otherwise NULL
READ_LOCKED_BY_COUNT	Count of current read locks held

## 1.1.2.9.2.1.58 Performance Schema session\_account\_connect\_attrs Table

### Description

The

```
session_account_connect_attrs
```

table shows connection attributes for the current session.

Applications can pass key/value connection attributes to the server when a connection is made. The [session\\_connect\\_attrs](#) and [session\\_account\\_connect\\_attrs](#) tables provide access to this information, for all sessions and the current session respectively.

The C API functions [mysql\\_options\(\)](#) and [mysql\\_optionsv\(\)](#) are used for passing connection attributes to the server.

```
session_account_connect_attrs
```

contains the following columns:

Column	Description
PROCESSLIST_ID	Session connection identifier.
ATTR_NAME	Attribute name.
ATTR_VALUE	Attribute value.
ORDINAL_POSITION	Order in which attribute was added to the connection attributes.

### Example

```
SELECT * FROM performance_schema.session_account_connect_attrs;
+-----+-----+-----+-----+
| PROCESSLIST_ID | ATTR_NAME      | ATTR_VALUE      | ORDINAL_POSITION |
+-----+-----+-----+-----+
|        45 | _os           | debian-linux-gnu |          0 |
|        45 | _client_name   | libmysql        |          1 |
|        45 | _pid           | 7711            |          2 |
|        45 | _client_version | 10.0.5          |          3 |
|        45 | _platform       | x86_64          |          4 |
|        45 | program_name    | mysql           |          5 |
+-----+-----+-----+-----+
```

## 1.1.2.9.2.1.59 Performance Schema session\_connect\_attrs

# Table

## Description

The

```
session_connect_attrs  
table shows connection attributes for all sessions.
```

Applications can pass key/value connection attributes to the server when a connection is made. The

```
session_connect_attrs  
and session_account_connect_attrs tables provide access to this information, for all sessions and the current session respectively.
```

The C API functions [mysql\\_options\(\)](#) and [mysql\\_optionsv\(\)](#) are used for passing connection attributes to the server.

```
session_connect_attrs  
contains the following columns:
```

Column	Description
PROCESSLIST_ID	Session connection identifier.
ATTR_NAME	Attribute name.
ATTR_VALUE	Attribute value.
ORDINAL_POSITION	Order in which attribute was added to the connection attributes.

## Example

Returning the current connection's attributes:

```
SELECT * FROM performance_schema.session_connect_attrs WHERE processlist_id=CONNECTION_ID();  
+-----+-----+-----+-----+  
| PROCESSLIST_ID | ATTR_NAME | ATTR_VALUE | ORDINAL_POSITION |  
+-----+-----+-----+-----+  
| 45 | _os | debian-linux-gnu | 0 |  
| 45 | _client_name | libmysql | 1 |  
| 45 | _pid | 7711 | 2 |  
| 45 | _client_version | 10.0.5 | 3 |  
| 45 | _platform | x86_64 | 4 |  
| 45 | program_name | mysql | 5 |  
+-----+-----+-----+-----+
```

## 1.1.2.9.2.1.60 Performance Schema session\_status Table

MariaDB starting with [10.5.2](#)

The

```
session_status  
table was added in MariaDB 10.5.2.
```

The

```
session_status  
table contains a list of status variables for the current session. The table only stores status variable statistics for threads which are instrumented,  
and does not collect statistics for  
Com_xxx  
variables.
```

The table contains the following columns:

Column	Description
--------	-------------

VARIABLE_NAME	The session status variable name.
VARIABLE_VALUE	The session status variable value.

It is not possible to empty this table with a

```
TRUNCATE TABLE  
statement.
```

## 1.1.2.9.2.1.61 Performance Schema setup\_actors Table

The

```
setup_actors
```

table contains information for determining whether monitoring should be enabled for new client connection threads.

The default size is 100 rows, which can be changed by modifying the

```
performance_schema_setup_actors_size
```

system variable at server startup.

If a row in the table matches a new foreground thread's client and host, the matching

```
INSTRUMENTED  
column in the threads table is set to either  
YES  
or  
NO
```

, which allows selective application of instrumenting by host, by user, or combination thereof.

Column	Description
HOST	Host name, either a literal, or the % wildcard representing any host.
USER	User name, either a literal or the % wildcard representing any name.
ROLE	Unused

Initially, any user and host is matched:

```
SELECT * FROM performance_schema.setup_actors;
+-----+-----+
| HOST | USER | ROLE |
+-----+-----+
| %   | %   | %   |
+-----+-----+
```

## 1.1.2.9.2.1.62 Performance Schema setup\_consumers Table

Lists the types of consumers for which event information is available.

The

```
setup_consumers
```

table contains the following columns:

Column	Description
NAME	Consumer name

ENABLED	YES or NO for whether or not the consumer is enabled. You can modify this column to ensure that event information is added, or is not added.
---------	---

The table can be modified directly, or the server started with the option enabled, for example:

```
performance-schema-consumer-events-waits-history=ON
```

## Example

```
SELECT * FROM performance_schema.setup_consumers;
```

NAME	ENABLED
events_stages_current	NO
events_stages_history	NO
events_stages_history_long	NO
events_statements_current	YES
events_statements_history	NO
events_statements_history_long	NO
events_waits_current	NO
events_waits_history	NO
events_waits_history_long	NO
global_instrumentation	YES
thread_instrumentation	YES
statements_digest	YES

## 1.1.2.9.2.1.63 Performance Schema setup\_instruments Table

The

`setup_instruments`

table contains a list of instrumented object classes for which it is possible to collect events. There is one row for each instrument in the source code. When an instrument is enabled and executed, instances are created which are then stored in the `cond_instances`, `file_instances`, `mutex_instances`, `rwlock_instances` or `socket_instance` tables.

It contains the following columns:

Column	Description
NAME	Instrument name
ENABLED	Whether or not the instrument is enabled. It can be disabled, and the instrument will produce no events.
TIMED	Whether or not the instrument is timed. It can be set, but if disabled, events produced by the instrument will have NULL values for the corresponding TIMER_START , TIMER_END , and TIMER_WAIT values.

## Example

From [MariaDB 10.5.7](#), default settings with the Performance Schema enabled:

```
SELECT * FROM setup_instruments ORDER BY name;
```

NAME	ENABLED	TIMED
events_stages	NO	NO

	YES	YES
idle	YES	YES
memory/csv/blobroot	NO	NO
memory/csv/row	NO	NO
memory/csv/tina_set	NO	NO
memory/csv/TINA_SHARE	NO	NO
memory/csv/Transparent_file	NO	NO
memory/innodb/adaptive hash index	NO	NO
memory/innodb/btr0btr	NO	NO
memory/innodb/btr0buf	NO	NO
memory/innodb/btr0bulk	NO	NO
memory/innodb/btr0cur	NO	NO
memory/innodb/btr0pcur	NO	NO
memory/innodb/btr0sea	NO	NO
memory/innodb/buf0buf	NO	NO
memory/innodb/buf0dblwr	NO	NO
memory/innodb/buf0dump	NO	NO
memory/innodb/buf_buf_pool	NO	NO
memory/innodb/dict0dict	NO	NO
memory/innodb/dict0mem	NO	NO
memory/innodb/dict0stats	NO	NO
memory/innodb/dict_stats_bg_recalc_pool_t	NO	NO
memory/innodb/dict_stats_index_map_t	NO	NO
memory/innodb/dict_stats_n_diff_on_level	NO	NO
memory/innodb/eval0eval	NO	NO
memory/innodb/fil0crypt	NO	NO
memory/innodb/fil0fil	NO	NO
memory/innodb/fsp0file	NO	NO
memory/innodb/fts0ast	NO	NO
memory/innodb/fts0plex	NO	NO
memory/innodb/fts0config	NO	NO
memory/innodb/fts0file	NO	NO
memory/innodb/fts0fts	NO	NO
memory/innodb/fts0opt	NO	NO
memory/innodb/fts0pars	NO	NO
memory/innodb/fts0que	NO	NO
memory/innodb/fts0sql	NO	NO
memory/innodb/fts0tlex	NO	NO
memory/innodb/gis0sea	NO	NO
memory/innodb/handler0alter	NO	NO
memory/innodb/hash0hash	NO	NO
memory/innodb/ha_innodb	NO	NO
memory/innodb/i_s	NO	NO
memory/innodb/lexyy	NO	NO
memory/innodb/lock0lock	NO	NO
memory/innodb/mem0mem	NO	NO
memory/innodb/os0event	NO	NO
memory/innodb/os0file	NO	NO
memory/innodb/other	NO	NO
memory/innodb/pars0lex	NO	NO
memory/innodb/rem0rec	NO	NO
memory/innodb/row0ftsort	NO	NO
memory/innodb/row0import	NO	NO
memory/innodb/row0log	NO	NO
memory/innodb/row0merge	NO	NO
memory/innodb/row0mysql	NO	NO
memory/innodb/row0sel	NO	NO
memory/innodb/row_log_buf	NO	NO
memory/innodb/row_merge_sort	NO	NO
memory/innodb/srv0start	NO	NO
memory/innodb/std	NO	NO
memory/innodb/sync0arr	NO	NO
memory/innodb/sync0debug	NO	NO
memory/innodb/sync0rw	NO	NO
memory/innodb/sync0start	NO	NO
memory/innodb/sync0types	NO	NO
memory/innodb/trx0i_s	NO	NO
memory/innodb/trx0roll	NO	NO
memory/innodb/trx0seg	NO	NO
memory/innodb/trx0seg	NO	NO
memory/innodb/trx0trx	NO	NO
memory/innodb/trx0undo	NO	NO
memory/innodb/ut0list	NO	NO
memory/innodb/ut0mem	NO	NO
memory/innodb/ut0new	NO	NO
memory/innodb/ut0pool	NO	NO
memory/innodb/ut0rbt	NO	NO
memory/innodb/ut0queue	NO	NO

memory/innodb/xtrabackup	NO	NO
memory/memory/HP_INFO	NO	NO
memory/memory/HP_KEYDEF	NO	NO
memory/memory/HP_PTRS	NO	NO
memory/memory/HP_SHARE	NO	NO
memory/myisam/filecopy	NO	NO
memory/myisam/FTB	NO	NO
memory/myisam/FTPARSER_PARAM	NO	NO
memory/myisam/FT_INFO	NO	NO
memory/myisam/ft_memroot	NO	NO
memory/myisam/ft_stopwords	NO	NO
memory/myisam/keycache_thread_var	NO	NO
memory/myisam/MI_DECODE_TREE	NO	NO
memory/myisam/MI_INFO	NO	NO
memory/myisam/MI_INFO::bulk_insert	NO	NO
memory/myisam/MI_INFO::ft1_to_ft2	NO	NO
memory/myisam/MI_SORT_PARAM	NO	NO
memory/myisam/MI_SORT_PARAM::wordroot	NO	NO
memory/myisam/MYISAM_SHARE	NO	NO
memory/myisam/MYISAM_SHARE::decode_tables	NO	NO
memory/myisam/preload_buffer	NO	NO
memory/myisam/record_buffer	NO	NO
memory/myisam/SORT_FT_BUF	NO	NO
memory/myisam/SORT_INFO::buffer	NO	NO
memory/myisam/SORT_KEY_BLOCKS	NO	NO
memory/myisam/stPageList::pages	NO	NO
memory/myisammrg/children	NO	NO
memory/myisammrg/MYRG_INFO	NO	NO
memory/partition/ha_partition::file	NO	NO
memory/partition/ha_partition::part_ids	NO	NO
memory/partition/Partition_admin	NO	NO
memory/partition/Partition_share	NO	NO
memory/partition/partition_sort_buffer	NO	NO
memory/performance_schema/accounts	YES	NO
memory/performance_schema/cond_class	YES	NO
memory/performance_schema/cond_instances	YES	NO
memory/performance_schema/events_stages_history	YES	NO
memory/performance_schema/events_stages_history_long	YES	NO
memory/performance_schema/events_stages_summary_by_account_by_event_name	YES	NO
memory/performance_schema/events_stages_summary_by_host_by_event_name	YES	NO
memory/performance_schema/events_stages_summary_by_thread_by_event_name	YES	NO
memory/performance_schema/events_stages_summary_by_user_by_event_name	YES	NO
memory/performance_schema/events_stages_summary_global_by_event_name	YES	NO
memory/performance_schema/events_statements_current	YES	NO
memory/performance_schema/events_statements_current.sqltext	YES	NO
memory/performance_schema/events_statements_current.tokens	YES	NO
memory/performance_schema/events_statements_history	YES	NO
memory/performance_schema/events_statements_history.sqltext	YES	NO
memory/performance_schema/events_statements_history.tokens	YES	NO
memory/performance_schema/events_statements_history_long	YES	NO
memory/performance_schema/events_statements_history_long.sqltext	YES	NO
memory/performance_schema/events_statements_history_long.tokens	YES	NO
memory/performance_schema/events_statements_summary_by_account_by_event_name	YES	NO
memory/performance_schema/events_statements_summary_by_digest	YES	NO
memory/performance_schema/events_statements_summary_by_digest.tokens	YES	NO
memory/performance_schema/events_statements_summary_by_host_by_event_name	YES	NO
memory/performance_schema/events_statements_summary_by_program	YES	NO
memory/performance_schema/events_statements_summary_by_thread_by_event_name	YES	NO
memory/performance_schema/events_statements_summary_by_user_by_event_name	YES	NO
memory/performance_schema/events_statements_summary_global_by_event_name	YES	NO
memory/performance_schema/events_transactions_history	YES	NO
memory/performance_schema/events_transactions_history_long	YES	NO
memory/performance_schema/events_transactions_summary_by_account_by_event_name	YES	NO
memory/performance_schema/events_transactions_summary_by_host_by_event_name	YES	NO
memory/performance_schema/events_transactions_summary_by_thread_by_event_name	YES	NO
memory/performance_schema/events_transactions_summary_by_user_by_event_name	YES	NO
memory/performance_schema/events_waits_history	YES	NO
memory/performance_schema/events_waits_history_long	YES	NO
memory/performance_schema/events_waits_summary_by_account_by_event_name	YES	NO
memory/performance_schema/events_waits_summary_by_host_by_event_name	YES	NO
memory/performance_schema/events_waits_summary_by_thread_by_event_name	YES	NO
memory/performance_schema/events_waits_summary_by_user_by_event_name	YES	NO
memory/performance_schema/file_class	YES	NO
memory/performance_schema/file_handle	YES	NO
memory/performance_schema/file_instances	YES	NO
memory/performance_schema/hosts	YES	NO
memory/performance_schema/memory_class	YES	NO

memory/performance_scnema/memory_summary_by_account_by_event_name	YES	NO
memory/performance_schema/memory_summary_by_host_by_event_name	YES	NO
memory/performance_schema/memory_summary_by_thread_by_event_name	YES	NO
memory/performance_schema/memory_summary_by_user_by_event_name	YES	NO
memory/performance_schema/memory_summary_global_by_event_name	YES	NO
memory/performance_schema/metadata_locks	YES	NO
memory/performance_schema/mutex_class	YES	NO
memory/performance_schema/mutex_instances	YES	NO
memory/performance_schema/prepared_statements_instances	YES	NO
memory/performance_schema/rwlock_class	YES	NO
memory/performance_schema/rwlock_instances	YES	NO
memory/performance_schema/scalable_buffer	YES	NO
memory/performance_schema/session_connect_attrs	YES	NO
memory/performance_schema/setup_actors	YES	NO
memory/performance_schema/setup_objects	YES	NO
memory/performance_schema/socket_class	YES	NO
memory/performance_schema/socket_instances	YES	NO
memory/performance_schema/stage_class	YES	NO
memory/performance_schema/statement_class	YES	NO
memory/performance_schema/table_handles	YES	NO
memory/performance_schema/table_io_waits_summary_by_index_usage	YES	NO
memory/performance_schema/table_lock_waits_summary_by_table	YES	NO
memory/performance_schema/table_shares	YES	NO
memory/performance_schema/threads	YES	NO
memory/performance_schema/thread_class	YES	NO
memory/performance_schema/users	YES	NO
memory/sql/acl_cache	NO	NO
memory/sql/binlog_cache_mngr	NO	NO
memory/sql/binlog_pos	NO	NO
memory/sql/binlog_statement_buffer	NO	NO
memory/sql/binlog_ver_1_event	NO	NO
memory/sql/bison_stack	NO	NO
memory/sql/Blob_mem_storage::storage	NO	NO
memory/sql/DATE_TIME_FORMAT	NO	NO
memory/sql/dboptions_hash	NO	NO
memory/sql/DDL_LOG_MEMORY_ENTRY	NO	NO
memory/sql/display_table_locks	NO	NO
memory/sql/errmsgs	NO	NO
memory/sql/Event_basic::mem_root	NO	NO
memory/sql/Event_queue_element_for_exec::names	NO	NO
memory/sql/Event_scheduler::scheduler_param	NO	NO
memory/sql/Filesort_info::merge	NO	NO
memory/sql/Filesort_info::record_pointers	NO	NO
memory/sql/frm::string	NO	NO
memory/sql/gdl	NO	NO
memory/sql/Gis_read_stream::err_msg	NO	NO
memory/sql/global_system_variables	NO	NO
memory/sql/handler::errmsgs	NO	NO
memory/sql/handler_ton	NO	NO
memory/sql/hash_index_key_buffer	NO	NO
memory/sql/host_cache::hostname	NO	NO
memory/sql/ignored_db	NO	NO
memory/sql/JOIN_CACHE	NO	NO
memory/sql/load_env_plugins	NO	NO
memory/sql/Locked_tables_list::m_locked_tables_root	NO	NO
memory/sql/MDL_context::acquire_locks	NO	NO
memory/sql/MPVIO_EXT::auth_info	NO	NO
memory/sql/MYSQL_BIN_LOG::basename	NO	NO
memory/sql/MYSQL_BIN_LOG::index	NO	NO
memory/sql/MYSQL_BIN_LOG::recover	NO	NO
memory/sql/MYSQL_LOCK	NO	NO
memory/sql/MYSQL_LOG::name	NO	NO
memory/sql/mysql_plugin	NO	NO
memory/sql/mysql_plugin_dl	NO	NO
memory/sql/MYSQL_RELAY_LOG::basename	NO	NO
memory/sql/MYSQL_RELAY_LOG::index	NO	NO
memory/sql/my_str_malloc	NO	NO
memory/sql/NAMED_ILINK::name	NO	NO
memory/sql/native_functions	NO	NO
memory/sql/plugin_bookmark	NO	NO
memory/sql/plugin_int_mem_root	NO	NO
memory/sql/plugin_mem_root	NO	NO
memory/sql/Prepared_statement::main_mem_root	NO	NO
memory/sql/Prepared_statement_map	NO	NO
memory/sql/PROFILE	NO	NO
memory/sql/Query_cache	NO	NO
memory/sql/Queue::queue_item	NO	NO
memory/sql/QUICK_RANGE_SELECT::alloc	NO	NO

memory/sql/QUICK_RANGE_SELECT::mrr_buf_desc	NO	NO	
memory/sql/Relay_log_info::group_relay_log_name	NO	NO	
memory/sql/root	NO	NO	
memory/sql/Row_data_memory::memory	NO	NO	
memory/sql/rpl_filter memory	NO	NO	
memory/sql/Rpl_info_file::buffer	NO	NO	
memory/sql/servers_cache	NO	NO	
memory/sql/SLAVE_INFO	NO	NO	
memory/sql/Sort_param::tmp_buffer	NO	NO	
memory/sql/sp_head::call_mem_root	NO	NO	
memory/sql/sp_head::execute_mem_root	NO	NO	
memory/sql/sp_head::main_mem_root	NO	NO	
memory/sql/sql_acl_mem	NO	NO	
memory/sql/sql_acl_memex	NO	NO	
memory/sql/String::value	NO	NO	
memory/sql/ST_SCHEMA_TABLE	NO	NO	
memory/sql/Sys_var_charptr::value	NO	NO	
memory/sql/TABLE	NO	NO	
memory/sql/table_mapping::m_mem_root	NO	NO	
memory/sql(TABLE_RULE_ENT	NO	NO	
memory/sql/TABLE_SHARE::mem_root	NO	NO	
memory/sql/Table_triggers_list	NO	NO	
memory/sql/Table_trigger_dispatcher::m_mem_root	NO	NO	
memory/sql/TC_LOG_MMAP::pages	NO	NO	
memory/sql/THD::db	NO	NO	
memory/sql/THD::handler_tables_hash	NO	NO	
memory/sql/thd::main_mem_root	NO	NO	
memory/sql/THD::sp_cache	NO	NO	
memory/sql/THD::transactions::mem_root	NO	NO	
memory/sql/THD::variables	NO	NO	
memory/sql/tz_storage	NO	NO	
memory/sql/udf_mem	NO	NO	
memory/sql/Unique::merge_buffer	NO	NO	
memory/sql/Unique::sort_buffer	NO	NO	
memory/sql/user_conn	NO	NO	
memory/sql/User_level_lock	NO	NO	
memory/sql/user_var_entry	NO	NO	
memory/sql/user_var_entry::value	NO	NO	
memory/sql/XID	NO	NO	
stage/aria/Waiting for a resource	NO	NO	
stage/innodb/alter table (end)	YES	YES	
stage/innodb/alter table (insert)	YES	YES	
stage/innodb/alter table (log apply index)	YES	YES	
stage/innodb/alter table (log apply table)	YES	YES	
stage/innodb/alter table (merge sort)	YES	YES	
stage/innodb/alter table (read PK and internal sort)	YES	YES	
stage/innodb/buffer pool load	YES	YES	
stage/mysys/Waiting for table level lock	NO	NO	
stage/sql/After apply log event	NO	NO	
stage/sql/After create	NO	NO	
stage/sql/After opening tables	NO	NO	
stage/sql/After table lock	NO	NO	
stage/sql/Allocating local table	NO	NO	
stage/sql/altering table	NO	NO	
stage/sql/Apply log event	NO	NO	
stage/sql/Changing master	NO	NO	
stage/sql/Checking master version	NO	NO	
stage/sql/checking permissions	NO	NO	
stage/sql/checking privileges on cached query	NO	NO	
stage/sql/Checking query cache for query	NO	NO	
stage/sql/closing tables	NO	NO	
stage/sql/Commit	NO	NO	
stage/sql/Commit implicit	NO	NO	
stage/sql/Committing alter table to storage engine	NO	NO	
stage/sql/Connecting to master	NO	NO	
stage/sql/Converting HEAP to Aria	NO	NO	
stage/sql/copy to tmp table	YES	YES	
stage/sql/Copying to group table	NO	NO	
stage/sql/Copying to tmp table	NO	NO	
stage/sql/Creating delayed handler	NO	NO	
stage/sql/Creating sort index	NO	NO	
stage/sql/creating table	NO	NO	
stage/sql/Creating tmp table	NO	NO	
stage/sql/Delete from main table	NO	NO	
stage/sql/Delete from reference tables	NO	NO	
stage/sql/discard_or_import_tablespace	NO	NO	
stage/sql/Enabling keys	NO	NO	
stage/sql/End of update loop	NO	NO	

stage/sql/End of update loop	NO	NO
stage/sql/Executing	NO	NO
stage/sql/Execution of init_command	NO	NO
stage/sql/Explaining	NO	NO
stage/sql/Filling schema table	NO	NO
stage/sql/Finding key cache	NO	NO
stage/sql/Finished reading one binlog; switching to next binlog	NO	NO
stage/sql/Flushing relay log and master info repository.	NO	NO
stage/sql/Flushing relay-log info file.	NO	NO
stage/sql/Freeing items	NO	NO
stage/sql/Fulltext initialization	NO	NO
stage/sql/Got handler lock	NO	NO
stage/sql/Got old table	NO	NO
stage/sql/init	NO	NO
stage/sql/init for update	NO	NO
stage/sql/Insert	NO	NO
stage/sql/Invalidating query cache entries (table list)	NO	NO
stage/sql/Invalidating query cache entries (table)	NO	NO
stage/sql/Killing slave	NO	NO
stage/sql/Logging slow query	NO	NO
stage/sql/Making temporary file (append) before replaying LOAD DATA INFILE	NO	NO
stage/sql/Making temporary file (create) before replaying LOAD DATA INFILE	NO	NO
stage/sql/Manage keys	NO	NO
stage/sql/Master has sent all binlog to slave; waiting for more updates	NO	NO
stage/sql/Opening tables	NO	NO
stage/sql/Optimizing	NO	NO
stage/sql/Preparing	NO	NO
stage/sql/preparing for alter table	NO	NO
stage/sql/Processing binlog checkpoint notification	NO	NO
stage/sql/Processing requests	NO	NO
stage/sql/Purging old relay logs	NO	NO
stage/sql/Query end	NO	NO
stage/sql/Queueing master event to the relay log	NO	NO
stage/sql/Reading event from the relay log	NO	NO
stage/sql/Reading semi-sync ACK from slave	NO	NO
stage/sql/Recreating table	NO	NO
stage/sql/Registering slave on master	NO	NO
stage/sql/Removing duplicates	NO	NO
stage/sql/Removing tmp table	NO	NO
stage/sql/Rename	NO	NO
stage/sql/Rename result table	NO	NO
stage/sql/Requesting binlog dump	NO	NO
stage/sql/Reschedule	NO	NO
stage/sql/Reset for next command	NO	NO
stage/sql/Rollback	NO	NO
stage/sql/Rollback_implicit	NO	NO
stage/sql/Searching rows for update	NO	NO
stage/sql/Sending binlog event to slave	NO	NO
stage/sql/Sending cached result to client	NO	NO
stage/sql/Sending data	NO	NO
stage/sql/setup	NO	NO
stage/sql>Show explain	NO	NO
stage/sql/Slave has read all relay log; waiting for more updates	NO	NO
stage/sql/Sorting	NO	NO
stage/sql/Sorting for group	NO	NO
stage/sql/Sorting for order	NO	NO
stage/sql/Sorting result	NO	NO
stage/sql/startting	NO	NO
stage/sql/Starting cleanup	NO	NO
stage/sql/Statistics	NO	NO
stage/sql/Stopping binlog background thread	NO	NO
stage/sql/Storing result in query cache	NO	NO
stage/sql/Storing row into queue	NO	NO
stage/sql/System lock	NO	NO
stage/sql/table lock	NO	NO
stage/sql/Unlocking tables	NO	NO
stage/sql/Update	NO	NO
stage/sql/Updating	NO	NO
stage/sql/Updating main table	NO	NO
stage/sql/Updating reference tables	NO	NO
stage/sql/Upgrading lock	NO	NO
stage/sql/User lock	NO	NO
stage/sql/User sleep	NO	NO
stage/sql/Verifying table	NO	NO
stage/sql/Waiting for background binlog tasks	NO	NO
stage/sql/Waiting for backup lock	NO	NO
stage/sql/Waiting for delay_list	NO	NO
stage/sql/Waiting for event metadata lock	NO	NO

stage/sql/Waiting for GTID to be written to binary log	NO	NO
stage/sql/Waiting for handler insert	NO	NO
stage/sql/Waiting for handler lock	NO	NO
stage/sql/Waiting for handler open	NO	NO
stage/sql/Waiting for INSERT	NO	NO
stage/sql/Waiting for master to send event	NO	NO
stage/sql/Waiting for master update	NO	NO
stage/sql/Waiting for next activation	NO	NO
stage/sql/Waiting for other master connection to process the same GTID	NO	NO
stage/sql/Waiting for parallel replication deadlock handling to complete	NO	NO
stage/sql/Waiting for prior transaction to commit	NO	NO
stage/sql/Waiting for prior transaction to start commit	NO	NO
stage/sql/Waiting for query cache lock	NO	NO
stage/sql/Waiting for requests	NO	NO
stage/sql/Waiting for room in worker thread event queue	NO	NO
stage/sql/Waiting for schema metadata lock	NO	NO
stage/sql/Waiting for semi-sync ACK from slave	NO	NO
stage/sql/Waiting for semi-sync slave connection	NO	NO
stage/sql/Waiting for slave mutex on exit	NO	NO
stage/sql/Waiting for slave thread to start	NO	NO
stage/sql/Waiting for stored function metadata lock	NO	NO
stage/sql/Waiting for stored package body metadata lock	NO	NO
stage/sql/Waiting for stored procedure metadata lock	NO	NO
stage/sql/Waiting for table flush	NO	NO
stage/sql/Waiting for table metadata lock	NO	NO
stage/sql/Waiting for the next event in relay log	NO	NO
stage/sql/Waiting for the scheduler to stop	NO	NO
stage/sql/Waiting for the slave SQL thread to advance position	NO	NO
stage/sql/Waiting for the slave SQL thread to free enough relay log space	NO	NO
stage/sql/Waiting for trigger metadata lock	NO	NO
stage/sql/Waiting for work from SQL thread	NO	NO
stage/sql/Waiting in MASTER_GTID_WAIT()	NO	NO
stage/sql/Waiting in MASTER_GTID_WAIT() (primary waiter)	NO	NO
stage/sql/Waiting on empty queue	NO	NO
stage/sql/Waiting to finalize termination	NO	NO
stage/sql/Waiting until MASTER_DELAY seconds after master executed event	NO	NO
stage/sql/Writing to binlog	NO	NO
statement/abstract/new_packet	YES	YES
statement/abstract/Query	YES	YES
statement/abstract/relay_log	YES	YES
statement/com/Binlog Dump	YES	YES
statement/com/Bulk_execute	YES	YES
statement/com/Change user	YES	YES
statement/com/Close stmt	YES	YES
statement/com/Com_multi	YES	YES
statement/com/Connect	YES	YES
statement/com/Connect Out	YES	YES
statement/com/Create DB	YES	YES
statement/com/Daemon	YES	YES
statement/com/Debug	YES	YES
statement/com/Delayed insert	YES	YES
statement/com/Drop DB	YES	YES
statement/com/Error	YES	YES
statement/com/Execute	YES	YES
statement/com/Fetch	YES	YES
statement/com/Field List	YES	YES
statement/com/Init DB	YES	YES
statement/com/Kill	YES	YES
statement/com/Long Data	YES	YES
statement/com/Ping	YES	YES
statement/com/Prepare	YES	YES
statement/com/Processlist	YES	YES
statement/com/Quit	YES	YES
statement/com/Refresh	YES	YES
statement/com/Register Slave	YES	YES
statement/com/Reset connection	YES	YES
statement/com/Reset stmt	YES	YES
statement/com/Set option	YES	YES
statement/com/Shutdown	YES	YES
statement/com/Slave_IO	YES	YES
statement/com/Slave_SQL	YES	YES
statement/com/Slave_worker	YES	YES
statement/com/Sleep	YES	YES
statement/com/Statistics	YES	YES
statement/com/Table Dump	YES	YES
statement/com/Time	YES	YES
statement/com/Unimpl get tid	YES	YES
statement/scheduler/event	YES	YES

statement/sp/agg_cfetch	YES	YES
statement/sp/cclose	YES	YES
statement/sp/cfetch	YES	YES
statement/sp/copen	YES	YES
statement/sp/cpop	YES	YES
statement/sp/cpush	YES	YES
statement/sp/cursor_copy_struct	YES	YES
statement/sp/error	YES	YES
statement/sp/freturn	YES	YES
statement/sp/hpop	YES	YES
statement/sp/hpush_jump	YES	YES
statement/sp/hreturn	YES	YES
statement/sp/jump	YES	YES
statement/sp/jump_if_not	YES	YES
statement/sp/preturn	YES	YES
statement/sp/set	YES	YES
statement/sp/set_case_expr	YES	YES
statement/sp/set_trigger_field	YES	YES
statement/sp/stmt	YES	YES
statement/sql/	YES	YES
statement/sql/alter_db	YES	YES
statement/sql/alter_db_upgrade	YES	YES
statement/sql/alter_event	YES	YES
statement/sql/alter_function	YES	YES
statement/sql/alter_procedure	YES	YES
statement/sql/alter_sequence	YES	YES
statement/sql/alter_server	YES	YES
statement/sql/alter_table	YES	YES
statement/sql/alter_tablespace	YES	YES
statement/sql/alter_user	YES	YES
statement/sql/analyze	YES	YES
statement/sql/assign_to_keycache	YES	YES
statement/sql/backup	YES	YES
statement/sql/backup_lock	YES	YES
statement/sql/begin	YES	YES
statement/sql/binlog	YES	YES
statement/sql/call_procedure	YES	YES
statement/sql/change_db	YES	YES
statement/sql/change_master	YES	YES
statement/sql/check	YES	YES
statement/sql/checksum	YES	YES
statement/sql/commit	YES	YES
statement/sql/compound_sql	YES	YES
statement/sql/create_db	YES	YES
statement/sql/create_event	YES	YES
statement/sql/create_function	YES	YES
statement/sql/create_index	YES	YES
statement/sql/create_package	YES	YES
statement/sql/create_package_body	YES	YES
statement/sql/create_procedure	YES	YES
statement/sql/create_role	YES	YES
statement/sql/create_sequence	YES	YES
statement/sql/create_server	YES	YES
statement/sql/create_table	YES	YES
statement/sql/create_trigger	YES	YES
statement/sql/create_udf	YES	YES
statement/sql/create_user	YES	YES
statement/sql/create_view	YES	YES
statement/sql/dealloc_sql	YES	YES
statement/sql/delete	YES	YES
statement/sql/delete_multi	YES	YES
statement/sql/do	YES	YES
statement/sql/drop_db	YES	YES
statement/sql/drop_event	YES	YES
statement/sql/drop_function	YES	YES
statement/sql/drop_index	YES	YES
statement/sql/drop_package	YES	YES
statement/sql/drop_package_body	YES	YES
statement/sql/drop_procedure	YES	YES
statement/sql/drop_role	YES	YES
statement/sql/drop_sequence	YES	YES
statement/sql/drop_server	YES	YES
statement/sql/drop_table	YES	YES
statement/sql/drop_trigger	YES	YES
statement/sql/drop_user	YES	YES
statement/sql/drop_view	YES	YES
statement/sql/empty_query	YES	YES

statement/sql/error	YES	YES
statement/sql/execute_immediate	YES	YES
statement/sql/execute_sql	YES	YES
statement/sql/flush	YES	YES
statement/sql/get_diagnostics	YES	YES
statement/sql/grant	YES	YES
statement/sql/grant_role	YES	YES
statement/sql/ha_close	YES	YES
statement/sql/ha_open	YES	YES
statement/sql/ha_read	YES	YES
statement/sql/help	YES	YES
statement/sql/insert	YES	YES
statement/sql/insert_select	YES	YES
statement/sql/install_plugin	YES	YES
statement/sql/kill	YES	YES
statement/sql/load	YES	YES
statement/sql/lock_tables	YES	YES
statement/sql/optimize	YES	YES
statement/sql/preload_keys	YES	YES
statement/sql/prepare_sql	YES	YES
statement/sql/purge	YES	YES
statement/sql/purge_before_date	YES	YES
statement/sql/release_savepoint	YES	YES
statement/sql/rename_table	YES	YES
statement/sql/rename_user	YES	YES
statement/sql/repair	YES	YES
statement/sql/replace	YES	YES
statement/sql/replace_select	YES	YES
statement/sql/reset	YES	YES
statement/sql/resignal	YES	YES
statement/sql/revoke	YES	YES
statement/sql/revoke_all	YES	YES
statement/sql/revoke_role	YES	YES
statement/sql/rollback	YES	YES
statement/sql/rollback_to_savepoint	YES	YES
statement/sql/savepoint	YES	YES
statement/sql/select	YES	YES
statement/sql/set_option	YES	YES
statement/sql/show_authors	YES	YES
statement/sql/show_binlogs	YES	YES
statement/sql/show_binlog_events	YES	YES
statement/sql/show_binlog_status	YES	YES
statement/sql/showCharsets	YES	YES
statement/sql/show_collations	YES	YES
statement/sql/show_contributors	YES	YES
statement/sql/show_create_db	YES	YES
statement/sql/show_create_event	YES	YES
statement/sql/show_create_func	YES	YES
statement/sql/show_create_package	YES	YES
statement/sql/show_create_package_body	YES	YES
statement/sql/show_create_proc	YES	YES
statement/sql/show_create_table	YES	YES
statement/sql/show_create_trigger	YES	YES
statement/sql/show_create_user	YES	YES
statement/sql/show_databases	YES	YES
statement/sql/show_engine_logs	YES	YES
statement/sql/show_engine_mutex	YES	YES
statement/sql/show_engine_status	YES	YES
statement/sql/show_errors	YES	YES
statement/sql/show_events	YES	YES
statement/sql/show_explain	YES	YES
statement/sql/show_fields	YES	YES
statement/sql/show_function_status	YES	YES
statement/sql/show_generic	YES	YES
statement/sql/show_grants	YES	YES
statement/sql/show_keys	YES	YES
statement/sql/show_open_tables	YES	YES
statement/sql/show_package_body_status	YES	YES
statement/sql/show_package_status	YES	YES
statement/sql/show_plugins	YES	YES
statement/sql/show_privileges	YES	YES
statement/sql/show_procedure_status	YES	YES
statement/sql/show_processlist	YES	YES
statement/sql/show_profile	YES	YES
statement/sql/show_profiles	YES	YES
statement/sql/show_relaylog_events	YES	YES
statement/sql/show_slave_hosts	YES	YES
statement/sql/show_slave_status	YES	YES

statement/sql/show_status	YES	YES	
statement/sql/show_storage_engines	YES	YES	
statement/sql/show_tables	YES	YES	
statement/sql/show_table_status	YES	YES	
statement/sql/show_triggers	YES	YES	
statement/sql/show_variables	YES	YES	
statement/sql/show_warnings	YES	YES	
statement/sql/shutdown	YES	YES	
statement/sql/signal	YES	YES	
statement/sql/start_all_slaves	YES	YES	
statement/sql/start_slave	YES	YES	
statement/sql/stop_all_slaves	YES	YES	
statement/sql/stop_slave	YES	YES	
statement/sql/truncate	YES	YES	
statement/sql/uninstall_plugin	YES	YES	
statement/sql/unlock_tables	YES	YES	
statement/sql/update	YES	YES	
statement/sql/update_multi	YES	YES	
statement/sql/xa_commit	YES	YES	
statement/sql/xa_end	YES	YES	
statement/sql/xa_prepare	YES	YES	
statement/sql/xa_recover	YES	YES	
statement/sql/xa_rollback	YES	YES	
statement/sql/xa_start	YES	YES	
transaction	NO	NO	
wait/io/file/aria/control	YES	YES	
wait/io/file/aria/MAD	YES	YES	
wait/io/file/aria/MAI	YES	YES	
wait/io/file/aria/translog	YES	YES	
wait/io/file/csv/data	YES	YES	
wait/io/file/csv/metadata	YES	YES	
wait/io/file/csv/update	YES	YES	
wait/io/file/innodb/innodb_data_file	YES	YES	
wait/io/file/innodb/innodb_log_file	YES	YES	
wait/io/file/innodb/innodb_temp_file	YES	YES	
wait/io/file/myisam/data_tmp	YES	YES	
wait/io/file/myisam/dfile	YES	YES	
wait/io/file/myisam/kfile	YES	YES	
wait/io/file/myisam/log	YES	YES	
wait/io/file/myisammrg/MRG	YES	YES	
wait/io/file/mysys/charset	YES	YES	
wait/io/file/mysys/cnf	YES	YES	
wait/io/file/partition/ha_partition::parfile	YES	YES	
wait/io/file/sql/binlog	YES	YES	
wait/io/file/sql/binlog_cache	YES	YES	
wait/io/file/sql/binlog_index	YES	YES	
wait/io/file/sql/binlog_index_cache	YES	YES	
wait/io/file/sql/binlog_state	YES	YES	
wait/io/file/sql/casetest	YES	YES	
wait/io/file/sql/dbopt	YES	YES	
wait/io/file/sql/des_key_file	YES	YES	
wait/io/file/sql/ERRMSG	YES	YES	
wait/io/file/sql/file_parser	YES	YES	
wait/io/file/sql/FRM	YES	YES	
wait/io/file/sql/global_ddl_log	YES	YES	
wait/io/file/sql/init	YES	YES	
wait/io/file/sql/io_cache	YES	YES	
wait/io/file/sql/load	YES	YES	
wait/io/file/sql/LOAD_FILE	YES	YES	
wait/io/file/sql/log_event_data	YES	YES	
wait/io/file/sql/log_event_info	YES	YES	
wait/io/file/sql/map	YES	YES	
wait/io/file/sql/master_info	YES	YES	
wait/io/file/sql/misc	YES	YES	
wait/io/file/sql/partition_ddl_log	YES	YES	
wait/io/file/sql/pid	YES	YES	
wait/io/file/sql/query_log	YES	YES	
wait/io/file/sql/relaylog	YES	YES	
wait/io/file/sql/relaylog_cache	YES	YES	
wait/io/file/sql/relaylog_index	YES	YES	
wait/io/file/sql/relaylog_index_cache	YES	YES	
wait/io/file/sql/relay_log_info	YES	YES	
wait/io/file/sql/select_to_file	YES	YES	
wait/io/file/sql/send_file	YES	YES	
wait/io/file/sql/slow_log	YES	YES	
wait/io/file/sql/tclog	YES	YES	
wait/io/file/sql/trigger	YES	YES	
wait/io/file/sql/trigexec_name	YES	YES	

	YES	YES	
wait/10/trigger/sql/trigger_name	YES	YES	
wait/io/file/sql/wsrep_gra_log	NO	NO	
wait/io/socket/sql/client_connection	NO	NO	
wait/io/socket/sql/server_tcpip_socket	NO	NO	
wait/io/socket/sql/server_unix_socket	NO	NO	
wait/io/table/sql/handler	YES	YES	
wait/lock/metadata/sql/mdl	NO	NO	
wait/lock/table/sql/handler	YES	YES	
wait/synch/cond/aria/BITMAP::bitmap_cond	NO	NO	
wait/synch/cond/aria/COND_soft_sync	NO	NO	
wait/synch/cond/aria/SERVICE_THREAD_CONTROL::COND_control	NO	NO	
wait/synch/cond/aria/SHARE::key_del_cond	NO	NO	
wait/synch/cond/aria/SORT_INFO::cond	NO	NO	
wait/synch/cond/aria/TRANSLOG_BUFFER::prev_sent_to_disk_cond	NO	NO	
wait/synch/cond/aria/TRANSLOG_BUFFER::waiting_filling_buffer	NO	NO	
wait/synch/cond/aria/TRANSLOG_DESCRIPTOR::log_flush_cond	NO	NO	
wait/synch/cond/aria/TRANSLOG_DESCRIPTOR::new_goal_cond	NO	NO	
wait/synch/cond/innodb/commit_cond	NO	NO	
wait/synch/cond/myisam/MI_SORT_INFO::cond	NO	NO	
wait/synch/cond/mysys/COND_alarm	NO	NO	
wait/synch/cond/mysys/COND_timer	NO	NO	
wait/synch/cond/mysys/IO_CACHE_SHARE::cond	NO	NO	
wait/synch/cond/mysys/IO_CACHE_SHARE::cond_writer	NO	NO	
wait/synch/cond/mysys/my_thread_var::suspend	NO	NO	
wait/synch/cond/mysys/THR_COND_threads	NO	NO	
wait/synch/cond/mysys/WT_RESOURCE::cond	NO	NO	
wait/synch/cond/sql/Ack_receiver::cond	NO	NO	
wait/synch/cond/sql/COND_binlog_send	NO	NO	
wait/synch/cond/sql/COND_flush_thread_cache	NO	NO	
wait/synch/cond/sql/COND_group_commit_orderer	NO	NO	
wait/synch/cond/sql/COND_gtid_ignore_duplicates	NO	NO	
wait/synch/cond/sql/COND_manager	NO	NO	
wait/synch/cond/sql/COND_parallel_entry	NO	NO	
wait/synch/cond/sql/COND_prepare_ordered	NO	NO	
wait/synch/cond/sql/COND_queue_state	NO	NO	
wait/synch/cond/sql/COND_rpl_thread	NO	NO	
wait/synch/cond/sql/COND_rpl_thread_pool	NO	NO	
wait/synch/cond/sql/COND_rpl_thread_queue	NO	NO	
wait/synch/cond/sql/COND_rpl_thread_stop	NO	NO	
wait/synch/cond/sql/COND_server_started	NO	NO	
wait/synch/cond/sql/COND_slave_background	NO	NO	
wait/synch/cond/sql/COND_start_thread	NO	NO	
wait/synch/cond/sql/COND_thread_cache	NO	NO	
wait/synch/cond/sql/COND_wait_gtid	NO	NO	
wait/synch/cond/sql/COND_wsrep_donor_monitor	NO	NO	
wait/synch/cond/sql/COND_wsrep_gtid_wait_upto	NO	NO	
wait/synch/cond/sql/COND_wsrep_joiner_monitor	NO	NO	
wait/synch/cond/sql/COND_wsrep_ready	NO	NO	
wait/synch/cond/sql/COND_wsrep_replaying	NO	NO	
wait/synch/cond/sql/COND_wsrep_sst	NO	NO	
wait/synch/cond/sql/COND_wsrep_sst_init	NO	NO	
wait/synch/cond/sql/COND_wsrep_slave_threads	NO	NO	
wait/synch/cond/sql/Delayed_insert::cond	NO	NO	
wait/synch/cond/sql/Delayed_insert::cond_client	NO	NO	
wait/synch/cond/sql/Event_scheduler::COND_state	NO	NO	
wait/synch/cond/sql/Item_func_sleep::cond	NO	NO	
wait/synch/cond/sql/Master_info::data_cond	NO	NO	
wait/synch/cond/sql/Master_info::sleep_cond	NO	NO	
wait/synch/cond/sql/Master_info::start_cond	NO	NO	
wait/synch/cond/sql/Master_info::stop_cond	NO	NO	
wait/synch/cond/sql/MDL_context::COND_wait_status	NO	NO	
wait/synch/cond/sql/MYSQL_BIN_LOG::COND_binlog_background_thread	NO	NO	
wait/synch/cond/sql/MYSQL_BIN_LOG::COND_binlog_background_thread_end	NO	NO	
wait/synch/cond/sql/MYSQL_BIN_LOG::COND_bin_log_updated	NO	NO	
wait/synch/cond/sql/MYSQL_BIN_LOG::COND_queue_busy	NO	NO	
wait/synch/cond/sql/MYSQL_BIN_LOG::COND_relay_log_updated	NO	NO	
wait/synch/cond/sql/MYSQL_BIN_LOG::COND_xid_list	NO	NO	
wait/synch/cond/sql/MYSQL_RELAY_LOG::COND_bin_log_updated	NO	NO	
wait/synch/cond/sql/MYSQL_RELAY_LOG::COND_queue_busy	NO	NO	
wait/synch/cond/sql/MYSQL_RELAY_LOG::COND_relay_log_updated	NO	NO	
wait/synch/cond/sql/PAGE::cond	NO	NO	
wait/synch/cond/sql/Query_cache::COND_cache_status_changed	NO	NO	
wait/synch/cond/sql/Relay_log_info::data_cond	NO	NO	
wait/synch/cond/sql/Relay_log_info::log_space_cond	NO	NO	
wait/synch/cond/sql/Relay_log_info::start_cond	NO	NO	
wait/synch/cond/sql/Relay_log_info::stop_cond	NO	NO	
wait/synch/cond/sql/Rpl_group_info::sleep_cond	NO	NO	
wait/synch/cond/sql/show_explain	NO	NO	

wait/synch/cond/sql/TABLE_SHARE::cond	NO	NO	
wait/synch/cond/sql/TABLE_SHARE::COND_rotation	NO	NO	
wait/synch/cond/sql/TABLE_SHARE::tdc.COND_release	NO	NO	
wait/synch/cond/sql/TC_LOG_MMAP::COND_active	NO	NO	
wait/synch/cond/sql/TC_LOG_MMAP::COND_pool	NO	NO	
wait/synch/cond/sql/TC_LOG_MMAP::COND_queue_busy	NO	NO	
wait/synch/cond/sql/THD::COND_wakeup_ready	NO	NO	
wait/synch/cond/sql/THD::COND_wsrep_thd	NO	NO	
wait/synch/cond/sql/User_level_lock::cond	NO	NO	
wait/synch/cond/sql/wait_for_commit::COND_wait_commit	NO	NO	
wait/synch/cond/sql/wsrep_sst_thread	NO	NO	
wait/synch/mutex/aria/LOCK_soft_sync	NO	NO	
wait/synch/mutex/aria/LOCK_trn_list	NO	NO	
wait/synch/mutex/aria/PAGECACHE::cache_lock	NO	NO	
wait/synch/mutex/aria/SERVICE_THREAD_CONTROL::LOCK_control	NO	NO	
wait/synch/mutex/aria/SHARE::bitmap::bitmap_lock	NO	NO	
wait/synch/mutex/aria/SHARE::close_lock	NO	NO	
wait/synch/mutex/aria/SHARE::intern_lock	NO	NO	
wait/synch/mutex/aria/SHARE::key_del_lock	NO	NO	
wait/synch/mutex/aria/SORT_INFO::mutex	NO	NO	
wait/synch/mutex/aria/THR_LOCK_maria	NO	NO	
wait/synch/mutex/aria/TRANSLOG_BUFFER::mutex	NO	NO	
wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::dirty_buffer_mask_lock	NO	NO	
wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::file_header_lock	NO	NO	
wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::log_flush_lock	NO	NO	
wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::purger_lock	NO	NO	
wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::sent_to_disk_lock	NO	NO	
wait/synch/mutex/aria/TRANSLOG_DESCRIPTOR::unfinished_files_lock	NO	NO	
wait/synch/mutex/aria/TRN::state_lock	NO	NO	
wait/synch/mutex/csv/tina	NO	NO	
wait/synch/mutex/csv/TINA_SHARE::mutex	NO	NO	
wait/synch/mutex/innodb/buf dblwr_mutex	NO	NO	
wait/synch/mutex/innodb/buf_pool_mutex	NO	NO	
wait/synch/mutex/innodb/commit_cond_mutex	NO	NO	
wait/synch/mutex/innodb/dict_foreign_err_mutex	NO	NO	
wait/synch/mutex/innodb/dict_sys_mutex	NO	NO	
wait/synch/mutex/innodb/fil_system_mutex	NO	NO	
wait/synch/mutex/innodb/flush_list_mutex	NO	NO	
wait/synch/mutex/innodb/fts_delete_mutex	NO	NO	
wait/synch/mutex/innodb/fts_doc_id_mutex	NO	NO	
wait/synch/mutex/innodb/ibuf_bitmap_mutex	NO	NO	
wait/synch/mutex/innodb/ibuf_mutex	NO	NO	
wait/synch/mutex/innodb/ibuf_pessimistic_insert_mutex	NO	NO	
wait/synch/mutex/innodb/lock_mutex	NO	NO	
wait/synch/mutex/innodb/lock_wait_mutex	NO	NO	
wait/synch/mutex/innodb/log_flush_order_mutex	NO	NO	
wait/synch/mutex/innodb/log_sys_mutex	NO	NO	
wait/synch/mutex/innodb/noredo_rseg_mutex	NO	NO	
wait/synch/mutex/innodb/page_zip_stat_per_index_mutex	NO	NO	
wait/synch/mutex/innodb/pending_checkpoint_mutex	NO	NO	
wait/synch/mutex/innodb/purge_sys_pg_mutex	NO	NO	
wait/synch/mutex/innodb/recalc_pool_mutex	NO	NO	
wait/synch/mutex/innodb/recv_sys_mutex	NO	NO	
wait/synch/mutex/innodb/redo_rseg_mutex	NO	NO	
wait/synch/mutex/innodb/rtr_active_mutex	NO	NO	
wait/synch/mutex/innodb/rtr_match_mutex	NO	NO	
wait/synch/mutex/innodb/rtr_path_mutex	NO	NO	
wait/synch/mutex/innodb/rw_lock_list_mutex	NO	NO	
wait/synch/mutex/innodb/srv_innodb_monitor_mutex	NO	NO	
wait/synch/mutex/innodb/srv_misc_tmpfile_mutex	NO	NO	
wait/synch/mutex/innodb/srv_monitor_file_mutex	NO	NO	
wait/synch/mutex/innodb/srv_threads_mutex	NO	NO	
wait/synch/mutex/innodb/trx_mutex	NO	NO	
wait/synch/mutex/innodb/trx_pool_manager_mutex	NO	NO	
wait/synch/mutex/innodb/trx_pool_mutex	NO	NO	
wait/synch/mutex/innodb/trx_sys_mutex	NO	NO	
wait/synch/mutex/myisam/MI_CHECK::print_msg	NO	NO	
wait/synch/mutex/myisam/MI_SORT_INFO::mutex	NO	NO	
wait/synch/mutex/myisam/MYISAM_SHARE::intern_lock	NO	NO	
wait/synch/mutex/myisammrg/MYRG_INFO::mutex	NO	NO	
wait/synch/mutex/mysys/BITMAP::mutex	NO	NO	
wait/synch/mutex/mysys/IO_CACHE::append_buffer_lock	NO	NO	
wait/synch/mutex/mysys/IO_CACHE::SHARE_mutex	NO	NO	
wait/synch/mutex/mysys/KEY_CACHE::cache_lock	NO	NO	
wait/synch/mutex/mysys/LOCK_alarm	NO	NO	
wait/synch/mutex/mysys/LOCK_timer	NO	NO	
wait/synch/mutex/mysys/LOCK_uuid_generator	NO	NO	
wait/synch/mutex/mvsvs/mv_thread_var::mutex	NO	NO	

wait/synch/mutex/myisam/THR_LOCK_myisam	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_charset	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_heap	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_lock	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_malloc	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_myisam	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_myisam_mmap	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_net	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_open	NO	NO	
wait/synch/mutex/myisam/THR_LOCK_threads	NO	NO	
wait/synch/mutex/tmpdir/TMPDIR_mutex	NO	NO	
wait/synch/mutex/partition/Partition_share::auto_inc_mutex	NO	NO	
wait/synch/mutex/sql/Ack_receiver::mutex	NO	NO	
wait/synch/mutex/sql/Cversion_lock	NO	NO	
wait/synch/mutex/sql/Delayed_insert::mutex	NO	NO	
wait/synch/mutex/sql/Event_scheduler::LOCK_scheduler_state	NO	NO	
wait/synch/mutex/sql/gtid_waiting::LOCK_gtid_waiting	NO	NO	
wait/synch/mutex/sql/hash_filo::lock	NO	NO	
wait/synch/mutex/sql/HA_DATA_PARTITION::LOCK_auto_inc	NO	NO	
wait/synch/mutex/sql/LOCK_active_mi	NO	NO	
wait/synch/mutex/sql/LOCK_after_binlog_sync	NO	NO	
wait/synch/mutex/sql/LOCK_audit_mask	NO	NO	
wait/synch/mutex/sql/LOCK_binlog	NO	NO	
wait/synch/mutex/sql/LOCK_binlog_state	NO	NO	
wait/synch/mutex/sql/LOCK_commit_ordered	NO	NO	
wait/synch/mutex/sql/LOCK_crypt	NO	NO	
wait/synch/mutex/sql/LOCK_delayed_create	NO	NO	
wait/synch/mutex/sql/LOCK_delayed_insert	NO	NO	
wait/synch/mutex/sql/LOCK_delayed_status	NO	NO	
wait/synch/mutex/sql/LOCK_des_key_file	NO	NO	
wait/synch/mutex/sql/LOCK_error_log	NO	NO	
wait/synch/mutex/sql/LOCK_error_messages	NO	NO	
wait/synch/mutex/sql/LOCK_event_queue	NO	NO	
wait/synch/mutex/sql/LOCK_gdl	NO	NO	
wait/synch/mutex/sql/LOCK_global_index_stats	NO	NO	
wait/synch/mutex/sql/LOCK_global_system_variables	NO	NO	
wait/synch/mutex/sql/LOCK_global_table_stats	NO	NO	
wait/synch/mutex/sql/LOCK_global_user_client_stats	NO	NO	
wait/synch/mutex/sql/LOCK_item_func_sleep	NO	NO	
wait/synch/mutex/sql/LOCK_load_client_plugin	NO	NO	
wait/synch/mutex/sql/LOCK_manager	NO	NO	
wait/synch/mutex/sql/LOCK_parallel_entry	NO	NO	
wait/synch/mutex/sql/LOCK_plugin	NO	NO	
wait/synch/mutex/sql/LOCK_prepared_stmt_count	NO	NO	
wait/synch/mutex/sql/LOCK_prepare_ordered	NO	NO	
wait/synch/mutex/sql/LOCK_rpl_semi_sync_master_enabled	NO	NO	
wait/synch/mutex/sql/LOCK_rpl_status	NO	NO	
wait/synch/mutex/sql/LOCK_rpl_thread	NO	NO	
wait/synch/mutex/sql/LOCK_rpl_thread_pool	NO	NO	
wait/synch/mutex/sql/LOCK_server_started	NO	NO	
wait/synch/mutex/sql/LOCK_slave_background	NO	NO	
wait/synch/mutex/sql/LOCK_slave_state	NO	NO	
wait/synch/mutex/sql/LOCK_start_thread	NO	NO	
wait/synch/mutex/sql/LOCK_stats	NO	NO	
wait/synch/mutex/sql/LOCK_status	NO	NO	
wait/synch/mutex/sql/LOCK_system_variables_hash	NO	NO	
wait/synch/mutex/sql/LOCK_table_cache	NO	NO	
wait/synch/mutex/sql/LOCK_thread_cache	NO	NO	
wait/synch/mutex/sql/LOCK_thread_id	NO	NO	
wait/synch/mutex/sql/LOCK_unused_shares	NO	NO	
wait/synch/mutex/sql/LOCK_user_conn	NO	NO	
wait/synch/mutex/sql/LOCK_uuid_short_generator	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_cluster_config	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_config_state	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_desync	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_donor_monitor	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_group_commit	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_gtid_wait_upto	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_joiner_monitor	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_ready	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_replaying	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_slave_threads	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_SR_pool	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_SR_store	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_sst	NO	NO	
wait/synch/mutex/sql/LOCK_wsrep_sst_init	NO	NO	
wait/synch/mutex/sql/LOG::LOCK_log	NO	NO	

wait/synch/mutex/sql/Master_info::data_lock	NO	NO	
wait/synch/mutex/sql/Master_info::run_lock	NO	NO	
wait/synch/mutex/sql/Master_info::sleep_lock	NO	NO	
wait/synch/mutex/sql/Master_info::start_stop_lock	NO	NO	
wait/synch/mutex/sql/MDL_wait::LOCK_wait_status	NO	NO	
wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_binlog_background_thread	NO	NO	
wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_binlog_end_pos	NO	NO	
wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_index	NO	NO	
wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_xid_list	NO	NO	
wait/synch/mutex/sql/MYSQL_RELAY_LOG::LOCK_binlog_end_pos	NO	NO	
wait/synch/mutex/sql/MYSQL_RELAY_LOG::LOCK_index	NO	NO	
wait/synch/mutex/sql/PAGE::lock	NO	NO	
wait/synch/mutex/sql/Query_cache::structure_guard_mutex	NO	NO	
wait/synch/mutex/sql/Relay_log_info::data_lock	NO	NO	
wait/synch/mutex/sql/Relay_log_info::log_space_lock	NO	NO	
wait/synch/mutex/sql/Relay_log_info::run_lock	NO	NO	
wait/synch/mutex/sql/Rpl_group_info::sleep_lock	NO	NO	
wait/synch/mutex/sql/Slave_reporting_capability::err_lock	NO	NO	
wait/synch/mutex/sql/TABLE_SHARE::LOCK_ha_data	NO	NO	
wait/synch/mutex/sql/TABLE_SHARE::LOCK_rotation	NO	NO	
wait/synch/mutex/sql/TABLE_SHARE::LOCK_share	NO	NO	
wait/synch/mutex/sql/TABLE_SHARE::tdc.LOCK_table_share	NO	NO	
wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_active	NO	NO	
wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_pending_checkpoint	NO	NO	
wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_pool	NO	NO	
wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_sync	NO	NO	
wait/synch/mutex/sql/THD::LOCK_thd_data	NO	NO	
wait/synch/mutex/sql/THD::LOCK_thd_kill	NO	NO	
wait/synch/mutex/sql/THD::LOCK_wakeup_ready	NO	NO	
wait/synch/mutex/sql/tz_LOCK	NO	NO	
wait/synch/mutex/sql/wait_for_commit::LOCK_wait_commit	NO	NO	
wait/synch/mutex/sql/wsrep_sst_thread	NO	NO	
wait/synch/rwlock/aria/KEYINFO::root_lock	NO	NO	
wait/synch/rwlock/aria/SHARE::mmap_lock	NO	NO	
wait/synch/rwlock/aria/TRANSLOG_DESCRIPTOR::open_files_lock	NO	NO	
wait/synch/rwlock/myisam/MYISAM_SHARE::key_root_lock	NO	NO	
wait/synch/rwlock/myisam/MYISAM_SHARE::mmap_lock	NO	NO	
wait/synch/rwlock/mysys/SAFE_HASH::mutex	NO	NO	
wait/synch/rwlock/proxy_proto/rwlock	NO	NO	
wait/synch/rwlock/sql/CRYPTO_dynlock_value::lock	NO	NO	
wait/synch/rwlock/sql/LOCK_all_status_vars	NO	NO	
wait/synch/rwlock/sql/LOCK_dboptions	NO	NO	
wait/synch/rwlock/sql/LOCK_grant	NO	NO	
wait/synch/rwlock/sql/LOCK_SEQUENCE	NO	NO	
wait/synch/rwlock/sql/LOCK_ssl_refresh	NO	NO	
wait/synch/rwlock/sql/LOCK_system_variables_hash	NO	NO	
wait/synch/rwlock/sql/LOCK_sys_init_connect	NO	NO	
wait/synch/rwlock/sql/LOCK_sys_init_slave	NO	NO	
wait/synch/rwlock/sql/LOGGER::LOCK_logger	NO	NO	
wait/synch/rwlock/sql/MDL_context::LOCK_waiting_for	NO	NO	
wait/synch/rwlock/sql/MDL_lock::rwlock	NO	NO	
wait/synch/rwlock/sql/Query_cache_query::lock	NO	NO	
wait/synch/rwlock/sql(TABLE_SHARE::LOCK_stat_serial	NO	NO	
wait/synch/rwlock/sql/THD_list::lock	NO	NO	
wait/synch/rwlock/sql/THR_LOCK_servers	NO	NO	
wait/synch/rwlock/sql/THR_LOCK_udf	NO	NO	
wait/synch/rwlock/sql/Vers_field_stats::lock	NO	NO	
wait/synch/sxlock/innodb/btr_search_latch	NO	NO	
wait/synch/sxlock/innodb/dict_operation_lock	NO	NO	
wait/synch/sxlock/innodb/fil_space_latch	NO	NO	
wait/synch/sxlock/innodb/fts_cache_init_rw_lock	NO	NO	
wait/synch/sxlock/innodb/fts_cache_rw_lock	NO	NO	
wait/synch/sxlock/innodb/index_online_log	NO	NO	
wait/synch/sxlock/innodb/index_tree_rw_lock	NO	NO	
wait/synch/sxlock/innodb/trx_i_s_cache_lock	NO	NO	
wait/synch/sxlock/innodb/trx_purge_latch	NO	NO	

+-----+-----+-----+

996 rows in set (0.005 sec)

## 1.1.2.9.2.1.64 Performance Schema setup\_objects Table

### Description

The

### setup\_objects

table determines whether objects are monitored by the performance schema or not. By default limited to 100 rows, this can be changed by setting the [performance\\_schema\\_setup\\_objects\\_size](#) system variable when the server starts.

It contains the following columns:

Column	Description
OBJECT_TYPE	Type of object to instrument, currently only . Currently, only TABLE , for base table.
OBJECT_SCHEMA	Schema containing the object, either the literal or % for any schema.
OBJECT_NAME	Name of the instrumented object, either the literal or % for any object.
ENABLED	Whether the object's events are instrumented or not. Can be disabled, in which case monitoring is not enabled for those objects.
TIMED	Whether the object's events are timed or not. Can be modified.

When the Performance Schema looks for matches in the

```
setup_objects
, there may be more than one row matching, with different
ENABLED
and
TIMED
```

values. It looks for the most specific matches first, that is, it will first look for the specific database and table name combination, then the specific database, only then falling back to a wildcard for both.

Rows can be added or removed from the table, while for existing rows, only the

```
TIMED
and
ENABLED
columns can be updated. By default, all tables except those in the
performance_schema
,
information_schema
and
mysql
databases are instrumented.
```

## 1.1.2.9.2.1.65 Performance Schema setup\_timers Table

### Description

The

```
setup_timers
table shows the currently selected event timers.
```

It contains the following columns:

Column	Description
NAME	Type of instrument the timer is used for.
TIMER_NAME	Timer applying to the instrument type. Can be modified.

The

```
TIMER_NAME
value can be changed to choose a different timer, and can be any non-NULL value in the performance\_timers.TIMER\_NAME column.
```

If you modify the table, monitoring is immediately affected, and currently monitored events would use a combination of old and new timers, which is probably undesirable. It is best to reset the Performance Schema statistics if you make changes to this table.

## Example

```
SELECT * FROM setup_timers;
+-----+-----+
| NAME      | TIMER_NAME   |
+-----+-----+
| idle      | MICROSECOND |
| wait      | CYCLE        |
| stage     | NANOSECOND   |
| statement | NANOSECOND   |
+-----+-----+
```

## 1.1.2.9.2.1.66 Performance Schema socket\_instances Table

The

socket\_instances  
table lists active server connections, with each record being a Unix socket file or TCP/IP connection.

The

socket\_instances  
table contains the following columns:

Column	Description
EVENT_NAME	NAME from the setup_instruments table, and the name of the wait/io/socket/* instrument that produced the event.
OBJECT_INSTANCE_BEGIN	Memory address of the object.
THREAD_ID	Thread identifier that the server assigns to each socket.
SOCKET_ID	The socket's internal file handle.
IP	Client IP address. Blank for Unix socket file, otherwise an IPv4 or IPv6 address. Together with the PORT identifies the connection.
PORT	TCP/IP port number, from 0 to 65535. Together with the IP identifies the connection.
STATE	Socket status, either IDLE if waiting to receive a request from a client, or ACTIVE

## 1.1.2.9.2.1.67 Performance Schema socket\_summary\_by\_event\_name Table

It aggregates timer and byte count statistics for all socket I/O operations by socket instrument.

Column	Description
EVENT_NAME	Socket instrument.

COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	<p>Number of all read operations, including</p> <p>RECV ,</p> <p>RECVFROM , and</p> <p>RECVMSG</p>
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
SUM_NUMBER_OF_BYTES_READ	Bytes read by read operations.
COUNT_WRITE	<p>Number of all write operations, including</p> <p>SEND ,</p> <p>SENDTO , and</p> <p>SENDMSG</p>
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
SUM_NUMBER_OF_BYTES_WRITE	Bytes written by write operations.

COUNT_MISC	Number of all miscellaneous operations not counted above, including CONNECT , LISTEN , ACCEPT , CLOSE , and SHUTDOWN .
SUM_TIMER_MISC	Total wait time of all miscellaneous operations that are timed.
MIN_TIMER_MISC	Minimum wait time of all miscellaneous operations that are timed.
AVG_TIMER_MISC	Average wait time of all miscellaneous operations that are timed.
MAX_TIMER_MISC	Maximum wait time of all miscellaneous operations that are timed.

You can [TRUNCATE](#) the table, which will reset all counters to zero.

## Example

```

SELECT * FROM socket_summary_by_event_name\G
*****
***** 1. row *****
    EVENT_NAME: wait/io/socket/sql/server_tcpip_socket
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
        COUNT_READ: 0
    SUM_TIMER_READ: 0
    MIN_TIMER_READ: 0
    AVG_TIMER_READ: 0
    MAX_TIMER_READ: 0
SUM_NUMBER_OF_BYTES_READ: 0
    COUNT_WRITE: 0
    SUM_TIMER_WRITE: 0
    MIN_TIMER_WRITE: 0
    AVG_TIMER_WRITE: 0
    MAX_TIMER_WRITE: 0
SUM_NUMBER_OF_BYTES_WRITE: 0
    COUNT_MISC: 0
    SUM_TIMER_MISC: 0
    MIN_TIMER_MISC: 0
    AVG_TIMER_MISC: 0
    MAX_TIMER_MISC: 0
*****
***** 2. row *****
    EVENT_NAME: wait/io/socket/sql/server_unix_socket
    COUNT_STAR: 0
    SUM_TIMER_WAIT: 0
    MIN_TIMER_WAIT: 0
    AVG_TIMER_WAIT: 0
    MAX_TIMER_WAIT: 0
        COUNT_READ: 0
    SUM_TIMER_READ: 0
    MIN_TIMER_READ: 0
    AVG_TIMER_READ: 0
    MAX_TIMER_READ: 0
SUM_NUMBER_OF_BYTES_READ: 0
    COUNT_WRITE: 0
    SUM_TIMER_WRITE: 0
    MIN_TIMER_WRITE: 0
    AVG_TIMER_WRITE: 0
    MAX_TIMER_WRITE: 0
SUM_NUMBER_OF_BYTES_WRITE: 0
    COUNT_MISC: 0
    SUM_TIMER_MISC: 0
    MIN_TIMER_MISC: 0
    AVG_TIMER_MISC: 0
    MAX_TIMER_MISC: 0
...

```

## 1.1.2.9.2.1.68 Performance Schema socket\_summary\_by\_instance Table

It aggregates timer and byte count statistics for all socket I/O operations by socket instance.

Column	Description
EVENT_NAME	Socket instrument.
OBJECT_INSTANCE_BEGIN	Address in memory.
COUNT_STAR	Number of summarized events
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.

MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, including RECV , RECVFROM , and RECVMSG .
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
SUM_NUMBER_OF_BYTES_READ	Bytes read by read operations.
COUNT_WRITE	Number of all write operations, including SEND , SENDTO , and SENDMSG .
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
SUM_NUMBER_OF_BYTES_WRITE	Bytes written by write operations.

COUNT_MISC	Number of all miscellaneous operations not counted above, including CONNECT , LISTEN , ACCEPT , CLOSE , and SHUTDOWN
SUM_TIMER_MISC	Total wait time of all miscellaneous operations that are timed.
MIN_TIMER_MISC	Minimum wait time of all miscellaneous operations that are timed.
AVG_TIMER_MISC	Average wait time of all miscellaneous operations that are timed.
MAX_TIMER_MISC	Maximum wait time of all miscellaneous operations that are timed.

The corresponding row in the table is deleted when a connection terminates.

You can [TRUNCATE](#) the table, which will reset all counters to zero.

## 1.1.2.9.2.1.69 Performance Schema status\_by\_thread Table

MariaDB starting with [10.5.2](#)

The

```
session_status
table was added in MariaDB 10.5.2 .
```

The

```
status_by_thread
table contains status variable information about active foreground threads. The table does not collect statistics for
Com_xxx
variables.
```

The table contains the following columns:

Column	Description
THREAD_ID	The thread identifier of the session in which the status variable is defined.
VARIABLE_NAME	Status variable name.
VARIABLE_VALUE	Aggregated status variable value.

If [TRUNCATE TABLE](#) is run, will aggregate the status for all threads to the global status and account status, then reset the thread status. If account statistics are not collected but host and user status are, the session status is added to host and user status.

## 1.1.2.9.2.1.70 Performance Schema table\_io\_waits\_summary\_by\_index\_usage Table

The

```
table_io_waits_summary_by_index_usage
table records table I/O waits by index.
```

Column	Description
OBJECT_TYPE	TABLE in the case of all indexes.
OBJECT_SCHEMA	Schema name.
OBJECT_NAME	Table name.
INDEX_NAME	Index name, or PRIMARY for the primary index, NULL for no index (inserts are counted in this case).
COUNT_STAR	Number of summarized events and the sum of the x_READ and x_WRITE columns.
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, and the sum of the equivalent x_FETCH columns.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
COUNT_WRITE	Number of all write operations, and the sum of the equivalent x_INSERT ,x_UPDATE and x_DELETE columns.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.

AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
COUNT_FETCH	Number of all fetch operations.
SUM_TIMER_FETCH	Total wait time of all fetch operations that are timed.
MIN_TIMER_FETCH	Minimum wait time of all fetch operations that are timed.
AVG_TIMER_FETCH	Average wait time of all fetch operations that are timed.
MAX_TIMER_FETCH	Maximum wait time of all fetch operations that are timed.
COUNT_INSERT	Number of all insert operations.
SUM_TIMER_INSERT	Total wait time of all insert operations that are timed.
MIN_TIMER_INSERT	Minimum wait time of all insert operations that are timed.
AVG_TIMER_INSERT	Average wait time of all insert operations that are timed.
MAX_TIMER_INSERT	Maximum wait time of all insert operations that are timed.
COUNT_UPDATE	Number of all update operations.
SUM_TIMER_UPDATE	Total wait time of all update operations that are timed.
MIN_TIMER_UPDATE	Minimum wait time of all update operations that are timed.
AVG_TIMER_UPDATE	Average wait time of all update operations that are timed.
MAX_TIMER_UPDATE	Maximum wait time of all update operations that are timed.
COUNT_DELETE	Number of all delete operations.
SUM_TIMER_DELETE	Total wait time of all delete operations that are timed.

MIN_TIMER_DELETE	Minimum wait time of all delete operations that are timed.
AVG_TIMER_DELETE	Average wait time of all delete operations that are timed.
MAX_TIMER_DELETE	Maximum wait time of all delete operations that are timed.

You can [TRUNCATE](#) the table, which will reset all counters to zero. The table is also truncated if the [table\\_io\\_waits\\_summary\\_by\\_table](#) table is truncated.

If a table's index structure is changed, index statistics recorded in this table may also be reset.

## 1.1.2.9.2.1.71 Performance Schema table\_io\_waits\_summary\_by\_table Table

The

```
table_io_waits_summary_by_table
table records table I/O waits by table.
```

Column	Description
OBJECT_TYPE	Since this table records waits by table, always set to TABLE.
OBJECT_SCHEMA	Schema name.
OBJECT_NAME	Table name.
COUNT_STAR	Number of summarized events and the sum of the x_READ and x_WRITE columns.
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.
COUNT_READ	Number of all read operations, and the sum of the equivalent x_FETCH columns.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.

MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
COUNT_WRITE	Number of all write operations, and the sum of the equivalent x_INSERT , x_UPDATE and x_DELETE columns.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
COUNT_FETCH	Number of all fetch operations.
SUM_TIMER_FETCH	Total wait time of all fetch operations that are timed.
MIN_TIMER_FETCH	Minimum wait time of all fetch operations that are timed.
AVG_TIMER_FETCH	Average wait time of all fetch operations that are timed.
MAX_TIMER_FETCH	Maximum wait time of all fetch operations that are timed.
COUNT_INSERT	Number of all insert operations.
SUM_TIMER_INSERT	Total wait time of all insert operations that are timed.
MIN_TIMER_INSERT	Minimum wait time of all insert operations that are timed.
AVG_TIMER_INSERT	Average wait time of all insert operations that are timed.
MAX_TIMER_INSERT	Maximum wait time of all insert operations that are timed.
COUNT_UPDATE	Number of all update operations.
SUM_TIMER_UPDATE	Total wait time of all update operations that are timed.

MIN_TIMER_UPDATE	Minimum wait time of all update operations that are timed.
AVG_TIMER_UPDATE	Average wait time of all update operations that are timed.
MAX_TIMER_UPDATE	Maximum wait time of all update operations that are timed.
COUNT_DELETE	Number of all delete operations.
SUM_TIMER_DELETE	Total wait time of all delete operations that are timed.
MIN_TIMER_DELETE	Minimum wait time of all delete operations that are timed.
AVG_TIMER_DELETE	Average wait time of all delete operations that are timed.
MAX_TIMER_DELETE	Maximum wait time of all delete operations that are timed.

You can [TRUNCATE](#) the table, which will reset all counters to zero. Truncating this table will also truncate the [table\\_io\\_waits\\_summary\\_by\\_index\\_usage](#) table.

## 1.1.2.9.2.1.72 Performance Schema table\_lock\_waits\_summary\_by\_table Table

The

```
table_lock_waits_summary_by_table
table records table lock waits by table.
```

Column	Description
OBJECT_TYPE	Since this table records waits by table, always set to TABLE.
OBJECT_SCHEMA	Schema name.
OBJECT_NAME	Table name.
COUNT_STAR	Number of summarized events and the sum of the x_READ and x_WRITE columns.
SUM_TIMER_WAIT	Total wait time of the summarized events that are timed.
MIN_TIMER_WAIT	Minimum wait time of the summarized events that are timed.
AVG_TIMER_WAIT	Average wait time of the summarized events that are timed.
MAX_TIMER_WAIT	Maximum wait time of the summarized events that are timed.

COUNT_READ	Number of all read operations, and the sum of the equivalent x_READ_NORMAL , x_READ_WITH_SHARED_LOCKS , x_READ_HIGH_PRIORITY and x_READ_NO_INSERT columns.
SUM_TIMER_READ	Total wait time of all read operations that are timed.
MIN_TIMER_READ	Minimum wait time of all read operations that are timed.
AVG_TIMER_READ	Average wait time of all read operations that are timed.
MAX_TIMER_READ	Maximum wait time of all read operations that are timed.
COUNT_WRITE	Number of all write operations, and the sum of the equivalent x_WRITE_ALLOW_WRITE , x_WRITE_CONCURRENT_INSERT , x_WRITE_DELAYED , x_WRITE_LOW_PRIORITY and x_WRITE_NORMAL columns.
SUM_TIMER_WRITE	Total wait time of all write operations that are timed.
MIN_TIMER_WRITE	Minimum wait time of all write operations that are timed.
AVG_TIMER_WRITE	Average wait time of all write operations that are timed.
MAX_TIMER_WRITE	Maximum wait time of all write operations that are timed.
COUNT_READ_NORMAL	Number of all internal read normal locks.
SUM_TIMER_READ_NORMAL	Total wait time of all internal read normal locks that are timed.
MIN_TIMER_READ_NORMAL	Minimum wait time of all internal read normal locks that are timed.
AVG_TIMER_READ_NORMAL	Average wait time of all internal read normal locks that are timed.
MAX_TIMER_READ_NORMAL	Maximum wait time of all internal read normal locks that are timed.
COUNT_READ_WITH_SHARED_LOCKS	Number of all internal read with shared locks.

SUM_TIMER_READ_WITH_SHARED_LOCKS	Total wait time of all internal read with shared locks that are timed.
MIN_TIMER_READ_WITH_SHARED_LOCKS	Minimum wait time of all internal read with shared locks that are timed.
AVG_TIMER_READ_WITH_SHARED_LOCKS	Average wait time of all internal read with shared locks that are timed.
MAX_TIMER_READ_WITH_SHARED_LOCKS	Maximum wait time of all internal read with shared locks that are timed.
COUNT_READ_HIGH_PRIORITY	Number of all internal read high priority locks.
SUM_TIMER_READ_HIGH_PRIORITY	Total wait time of all internal read high priority locks that are timed.
MIN_TIMER_READ_HIGH_PRIORITY	Minimum wait time of all internal read high priority locks that are timed.
AVG_TIMER_READ_HIGH_PRIORITY	Average wait time of all internal read high priority locks that are timed.
MAX_TIMER_READ_HIGH_PRIORITY	Maximum wait time of all internal read high priority locks that are timed.
COUNT_READ_NO_INSERT	Number of all internal read no insert locks.
SUM_TIMER_READ_NO_INSERT	Total wait time of all internal read no insert locks that are timed.
MIN_TIMER_READ_NO_INSERT	Minimum wait time of all internal read no insert locks that are timed.
AVG_TIMER_READ_NO_INSERT	Average wait time of all internal read no insert locks that are timed.
MAX_TIMER_READ_NO_INSERT	Maximum wait time of all internal read no insert locks that are timed.
COUNT_READ_EXTERNAL	Number of all external read locks.
SUM_TIMER_READ_EXTERNAL	Total wait time of all external read locks that are timed.
MIN_TIMER_READ_EXTERNAL	Minimum wait time of all external read locks that are timed.
AVG_TIMER_READ_EXTERNAL	Average wait time of all external read locks that are timed.
MAX_TIMER_READ_EXTERNAL	Maximum wait time of all external read locks that are timed.
COUNT_WRITE_ALLOW_WRITE	Number of all internal read normal locks.

SUM_TIMER_WRITE_ALLOW_WRITE	Total wait time of all internal write allow write locks that are timed.
MIN_TIMER_WRITE_ALLOW_WRITE	Minimum wait time of all internal write allow write locks that are timed.
AVG_TIMER_WRITE_ALLOW_WRITE	Average wait time of all internal write allow write locks that are timed.
MAX_TIMER_WRITE_ALLOW_WRITE	Maximum wait time of all internal write allow write locks that are timed.
COUNT_WRITE_CONCURRENT_INSERT	Number of all internal concurrent insert write locks.
SUM_TIMER_WRITE_CONCURRENT_INSERT	Total wait time of all internal concurrent insert write locks that are timed.
MIN_TIMER_WRITE_CONCURRENT_INSERT	Minimum wait time of all internal concurrent insert write locks that are timed.
AVG_TIMER_WRITE_CONCURRENT_INSERT	Average wait time of all internal concurrent insert write locks that are timed.
MAX_TIMER_WRITE_CONCURRENT_INSERT	Maximum wait time of all internal concurrent insert write locks that are timed.
COUNT_WRITE_DELAYED	Number of all internal write delayed locks.
SUM_TIMER_WRITE_DELAYED	Total wait time of all internal write delayed locks that are timed.
MIN_TIMER_WRITE_DELAYED	Minimum wait time of all internal write delayed locks that are timed.
AVG_TIMER_WRITE_DELAYED	Average wait time of all internal write delayed locks that are timed.
MAX_TIMER_WRITE_DELAYED	Maximum wait time of all internal write delayed locks that are timed.
COUNT_WRITE_LOW_PRIORITY	Number of all internal write low priority locks.
SUM_TIMER_WRITE_LOW_PRIORITY	Total wait time of all internal write low priority locks that are timed.
MIN_TIMER_WRITE_LOW_PRIORITY	Minimum wait time of all internal write low priority locks that are timed.
AVG_TIMER_WRITE_LOW_PRIORITY	Average wait time of all internal write low priority locks that are timed.
MAX_TIMER_WRITE_LOW_PRIORITY	Maximum wait time of all internal write low priority locks that are timed.
COUNT_WRITE_NORMAL	Number of all internal write normal locks.

SUM_TIMER_WRITE_NORMAL	Total wait time of all internal write normal locks that are timed.
MIN_TIMER_WRITE_NORMAL	Minimum wait time of all internal write normal locks that are timed.
AVG_TIMER_WRITE_NORMAL	Average wait time of all internal write normal locks that are timed.
MAX_TIMER_WRITE_NORMAL	Maximum wait time of all internal write normal locks that are timed.
COUNT_WRITE_EXTERNAL	Number of all external write locks.
SUM_TIMER_WRITE_EXTERNAL	Total wait time of all external write locks that are timed.
MIN_TIMER_WRITE_EXTERNAL	Minimum wait time of all external write locks that are timed.
AVG_TIMER_WRITE_EXTERNAL	Average wait time of all external write locks that are timed.
MAX_TIMER_WRITE_EXTERNAL	Maximum wait time of all external write locks that are timed.

You can [TRUNCATE](#) the table, which will reset all counters to zero.

## 1.1.2.9.2.1.73 Performance Schema threads Table

Each server thread is represented as a row in the

threads  
table.

The

threads  
table contains the following columns:

Column	Description
THREAD_ID	A unique thread identifier.
NAME	Name associated with the server's thread instrumentation code, for example thread/sql/main for the server's main() function, and thread/sql/one_connection for a user connection.
TYPE	FOREGROUND or BACKGROUND , depending on the thread type. User connection threads are FOREGROUND , internal server threads are BACKGROUND .

PROCESSLIST_ID	The PROCESSLIST.ID value for threads displayed in the INFORMATION_SCHEMA.PROCESSLIST table, or 0 for background threads. Also corresponds with the CONNECTION_ID() return value for the thread.
PROCESSLIST_USER	Foreground thread user, or NULL for a background thread.
PROCESSLIST_HOST	Foreground thread host, or NULL for a background thread.
PROCESSLIST_DB	Thread's default database, or NULL if none exists.
PROCESSLIST_COMMAND	Type of command executed by the thread. These correspond to the the COM_xxx client/server protocol commands, and the Com_xxx status variables . See <a href="#">Thread Command Values</a> .
PROCESSLIST_TIME	Time in seconds the thread has been in its current state.
PROCESSLIST_STATE	Action, event or state indicating what the thread is doing.
PROCESSLIST_INFO	Statement being executed by the thread, or NULL if a statement is not being executed. If a statement results in calling other statements, such as for a <a href="#">stored procedure</a> , the innermost statement from the stored procedure is shown here.
PARENT_THREAD_ID	THREAD_ID of the parent thread, if any. Subthreads can for example be spawned as a result of <a href="#">INSERT DELAYED</a> statements.
ROLE	Unused.
INSTRUMENTED	YES or NO for Whether the thread is instrumented or not. For foreground threads, the initial value is determined by whether there's a user/host match in the <a href="#">setup_actors</a> table. Subthreads are again matched, while for background threads, this will be set to YES by default. To monitor events that the thread executes, INSTRUMENTED must be YES and the thread_instrumentation consumer in the <a href="#">setup_consumers</a> table must also be YES

HISTORY	<p>YES or NO</p> <p>for Whether to log historical events for the thread. For foreground threads, the initial value is determined by whether there's a user/host match in the <code>setup_actors</code> table. Subthreads are again matched, while for background threads, this will be set to</p> <p>YES by default. To monitor events that the thread executes, INSTRUMENTED must be YES and the thread_instrumentation consumer in the <code>setup_consumers</code> table must also be YES . Added in <a href="#">MariaDB 10.5</a>.</p>
CONNECTION_TYPE	The protocol used to establish the connection, or NULL for background threads. Added in <a href="#">MariaDB 10.5</a> .
THREAD_OS_ID	The thread or task identifier as defined by the underlying operating system, if there is one. Added in <a href="#">MariaDB 10.5</a>

## Example

```
SELECT * FROM performance_schema.threads\G;
*****
1. row ****
    THREAD_ID: 1
        NAME: thread/sql/main
        TYPE: BACKGROUND
    PROCESSLIST_ID: NULL
    PROCESSLIST_USER: NULL
    PROCESSLIST_HOST: NULL
    PROCESSLIST_DB: NULL
PROCESSLIST_COMMAND: NULL
PROCESSLIST_TIME: 215859
PROCESSLIST_STATE: Table lock
PROCESSLIST_INFO: INTERNAL DDL LOG RECOVER IN PROGRESS
PARENT_THREAD_ID: NULL
        ROLE: NULL
    INSTRUMENTED: YES
...
*****
21. row ****
    THREAD_ID: 64
        NAME: thread/sql/one_connection
        TYPE: FOREGROUND
    PROCESSLIST_ID: 44
    PROCESSLIST_USER: root
    PROCESSLIST_HOST: localhost
    PROCESSLIST_DB: NULL
PROCESSLIST_COMMAND: Query
PROCESSLIST_TIME: 0
PROCESSLIST_STATE: Sending data
PROCESSLIST_INFO: SELECT * FROM performance_schema.threads
PARENT_THREAD_ID: NULL
        ROLE: NULL
    INSTRUMENTED: YES
```

## 1.1.2.9.2.1.74 Performance Schema users Table

### Description

Each user that connects to the server is stored as a row in the

`users`  
table, along with current and total connections.

The table size is determined at startup by the value of the `performance_schema_users_size` system variable. If this is set to

0

, user statistics will be disabled.

Column	Description
USER	The connection's client user name for the connection, or NULL if an internal thread.
CURRENT_CONNECTIONS	Current connections for the user.
TOTAL_CONNECTIONS	Total connections for the user.

## Example

```
SELECT * FROM performance_schema.users;
+-----+-----+-----+
| USER      | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+-----+-----+-----+
| debian-sys-maint |          0 |          35 |
| NULL       |         20 |          23 |
| root        |          1 |           2 |
+-----+-----+-----+
```

## 1.1.2.9.2.2 Performance Schema Overview

### Contents

1. [Introduction](#)
2. [Activating the Performance Schema](#)
3. [Enabling the Performance Schema](#)
4. [Listing Performance Schema Variables](#)
5. [Column Comments](#)
6. [See Also](#)

The Performance Schema is a feature for monitoring server performance.

## Introduction

It is implemented as a storage engine, and so will appear in the list of storage engines available.

```
SHOW ENGINES;
+-----+-----+-----+-----+-----+-----+
| Engine      | Support | Comment           | Transactions | XA   | Savepoints |
+-----+-----+-----+-----+-----+-----+
| ...         |     |     |     |     |     |
| PERFORMANCE_SCHEMA | YES    | Performance Schema | NO        | NO  | NO
| ...         |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+
```

However,

```
performance_schema
is not a regular storage engine for storing data, it's a mechanism for implementing the Performance Schema feature.
```

The storage engine contains a database called

```
performance_schema
, which in turn consists of a number of tables that can be queried with regular SQL statements, returning specific performance information.
```

```
USE performance_schema
```

```
SHOW TABLES;
+-----+
| Tables_in_performance_schema |
+-----+
| accounts                      |
...
| users                         |
+-----+
80 rows in set (0.00 sec)
```

See [List of Performance Schema Tables](#) for a full list and links to detailed descriptions of each table. From [MariaDB 10.5](#), there are 80 Performance Schema tables, while until [MariaDB 10.4](#), there are 52.

## Activating the Performance Schema

The performance schema is disabled by default for performance reasons. You can check its current status by looking at the value of the `performance_schema` system variable.

```
SHOW VARIABLES LIKE 'performance_schema';
+-----+
| Variable_name      | Value   |
+-----+
| performance_schema | ON      |
+-----+
```

The performance schema cannot be activated at runtime - it must be set when the server starts by adding the following line in your `my.cnf` configuration file.

```
performance_schema=ON
```

Until [MariaDB 10.4](#), all memory used by the Performance Schema is allocated at startup. From [MariaDB 10.5](#), some memory is allocated dynamically, depending on load, number of connections, number of tables open etc.

## Enabling the Performance Schema

You need to set up all consumers (starting collection of data) and instrumentations (what to collect):

```
UPDATE performance_schema.setup_consumers SET ENABLED = 'YES';
UPDATE performance_schema.setup_instruments SET ENABLED = 'YES', TIMED = 'YES';
```

You can decide what to enable/disable with

```
WHERE NAME like "%what_to_enable"
; You can disable instrumentations by setting
ENABLED
to
"NO"
```

You can also do this in your `my.cnf` file. The following enables all instrumentation of all stages (computation units) in MariaDB:

```
[mysqld]
performance_schema=ON
performance-schema-instrument='stage/%=ON'
performance-schema-consumer-events-stages-current=ON
performance-schema-consumer-events-stages-history=ON
performance-schema-consumer-events-stages-history-long=ON
```

## Listing Performance Schema Variables

```
SHOW VARIABLES LIKE "perf%";
+-----+
| Variable_name          | Value   |
+-----+
| performance_schema     | ON      |
...
| performance_schema_users_size | 100    |
+-----+
```

See [Performance Schema System Variables](#) for a full list of available system variables.

Note that the "consumer" events are not shown on this list, as they are only available as options, not as system variables, and they can only be enabled at [startup](#).

## Column Comments

MariaDB starting with 10.7.1

From [MariaDB 10.7.1](#), comments have been added to table columns in the Performance Schema. These can be viewed with, for example:

```
SELECT column_name, column_comment FROM information_schema.columns
  WHERE table_schema='performance_schema' AND table_name='file_instances';
...
***** 2. row *****
column_name: EVENT_NAME
column_comment: Instrument name associated with the file.
***** 3. row *****
column_name: OPEN_COUNT
column_comment: Open handles on the file. A value of greater than zero means
                that the file is currently open.
...
```

## See Also

- [Performance schema options](#)
- [SHOW ENGINE STATUS](#)
- [SHOW PROFILE](#)
- [ANALYZE STATEMENT](#)
- [Performance schema in MySQL 5.6](#). All things here should also work for MariaDB.

### 1.1.2.9.2.3 Performance Schema Status Variables

#### Contents

1. [Performance\\_schema\\_accounts\\_lost](#)
2. [Performance\\_schema\\_cond\\_classes\\_lost](#)
3. [Performance\\_schema\\_cond\\_instances\\_lost](#)
4. [Performance\\_schema\\_digest\\_lost](#)
5. [Performance\\_schema\\_file\\_classes\\_lost](#)
6. [Performance\\_schema\\_file\\_handles\\_lost](#)
7. [Performance\\_schema\\_file\\_instances\\_lost](#)
8. [Performance\\_schema\\_hosts\\_lost](#)
9. [Performance\\_schema\\_index\\_stat\\_lost](#)
10. [Performance\\_schema\\_locker\\_lost](#)
11. [Performance\\_schema\\_memory\\_classes\\_lost](#)
12. [Performance\\_schema\\_metadata\\_lock\\_lost](#)
13. [Performance\\_schema\\_mutex\\_classes\\_lost](#)
14. [Performance\\_schema\\_mutex\\_instances\\_lost](#)
15. [Performance\\_schema\\_nested\\_statement\\_id](#)
16. [Performance\\_schema\\_prepared\\_statement\\_id](#)
17. [Performance\\_schema\\_program\\_lost](#)
18. [Performance\\_schema\\_rwlock\\_classes\\_lost](#)
19. [Performance\\_schema\\_rwlock\\_instances\\_lost](#)
20. [Performance\\_schema\\_session\\_connect\\_attributes](#)
21. [Performance\\_schema\\_socket\\_classes\\_lost](#)
22. [Performance\\_schema\\_socket\\_instances\\_lost](#)
23. [Performance\\_schema\\_stage\\_classes\\_lost](#)
24. [Performance\\_schema\\_statement\\_classes\\_lost](#)
25. [Performance\\_schema\\_table\\_handles\\_lost](#)
26. [Performance\\_schema\\_table\\_instances\\_lost](#)
27. [Performance\\_schema\\_table\\_lock\\_stat\\_lost](#)
28. [Performance\\_schema\\_thread\\_classes\\_lost](#)
29. [Performance\\_schema\\_thread\\_instances\\_lost](#)
30. [Performance\\_schema\\_users\\_lost](#)

This page documents status variables related to the [Performance Schema](#). See [Server Status Variables](#) for a complete list of status variables that can be viewed with [SHOW STATUS](#).

See also the [Full list of MariaDB options, system and status variables](#).

## `Performance_schema_accounts_lost`

- **Description:** Number of times a row could not be added to the performance schema accounts table due to it being full. The global value can be flushed by [FLUSH STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

## `Performance_schema_cond_classes_lost`

- **Description:** Number of condition instruments that could not be loaded.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

## `Performance_schema_cond_instances_lost`

- **Description:** Number of instances a condition object could not be created. The global value can be flushed by [FLUSH STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

## `Performance_schema_digest_lost`

- **Description:** The global value can be flushed by [FLUSH STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

## `Performance_schema_file_classes_lost`

- **Description:** Number of file instruments that could not be loaded.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

## `Performance_schema_file_handles_lost`

- **Description:** Number of instances a file object could not be opened. The global value can be flushed by [FLUSH STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

## `Performance_schema_file_instances_lost`

- **Description:** Number of instances a file object could not be created. The global value can be flushed by [FLUSH STATUS](#) .

- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Performance\_schema\_hosts\_lost

- **Description:** Number of times a row could not be added to the performance schema hosts table due to it being full. The global value can be flushed by [FLUSH STATUS](#) .

- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Performance\_schema\_index\_stat\_lost

- **Description:**

- **Scope:** Global, Session

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.5.2
- 

#### Performance\_schema\_locker\_lost

- **Description:** Number of events not recorded, due to either being recursive, or having a deeper nested events stack than the implementation limit. The global value can be flushed by [FLUSH STATUS](#) .

- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Performance\_schema\_memory\_classes\_lost

- **Description:**

- **Scope:** Global, Session

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.5.2
- 

#### Performance\_schema\_metadata\_lock\_lost

- **Description:**

- **Scope:** Global, Session

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.5.2
- 

#### Performance\_schema\_mutex\_classes\_lost

- **Description:** Number of mutual exclusion instruments that could not be loaded.
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

#### Performance\_schema\_mutex\_instances\_lost

- **Description:** Number of instances a mutual exclusion object could not be created. The global value can be flushed by [FLUSH STATUS](#).
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

#### Performance\_schema\_nested\_statement\_lost

- **Description:**
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- **Introduced:** [MariaDB 10.5.2](#)
- 

#### Performance\_schema\_prepared\_statements\_lost

- **Description:**
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- **Introduced:** [MariaDB 10.5.2](#)
- 

#### Performance\_schema\_program\_lost

- **Description:**
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- **Introduced:** [MariaDB 10.5.2](#)
- 

#### Performance\_schema\_rwlock\_classes\_lost

- **Description:** Number of read/write lock instruments that could not be loaded.
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

#### Performance\_schema\_rwlock\_instances\_lost

- **Description:** Number of instances a read/write lock object could not be created. The global value can be flushed by [FLUSH STATUS](#).
- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Performance\_schema\_session\_connect\_attrs\_lost

- **Description:** Number of connections for which connection attribute truncation has occurred. The global value can be flushed by [FLUSH STATUS](#).
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### Performance\_schema\_socket\_classes\_lost

- **Description:**
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### Performance\_schema\_socket\_instances\_lost

- **Description:** Number of instances a socket object could not be created. The global value can be flushed by [FLUSH STATUS](#).
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### Performance\_schema\_stage\_classes\_lost

- **Description:** Number of stage event instruments that could not be loaded. The global value can be flushed by [FLUSH STATUS](#).
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### Performance\_schema\_statement\_classes\_lost

- **Description:** Number of statement instruments that could not be loaded. The global value can be flushed by [FLUSH STATUS](#).
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### Performance\_schema\_table\_handles\_lost

- **Description:** Number of instances a table object could not be opened. The global value can be flushed by [FLUSH STATUS](#).
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### `Performance_schema_table_instances_lost`

- **Description:** Number of instances a table object could not be created. The global value can be flushed by [FLUSH STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

#### `Performance_schema_table_lock_stat_lost`

- **Description:**
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.5.2](#)
- 

#### `Performance_schema_thread_classes_lost`

- **Description:** Number of thread instruments that could not be loaded.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

#### `Performance_schema_thread_instances_lost`

- **Description:** Number of instances thread object could not be created. The global value can be flushed by [FLUSH STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

#### `Performance_schema_users_lost`

- **Description:** Number of times a row could not be added to the performance schema users table due to it being full. The global value can be flushed by [FLUSH STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

## 1.1.2.9.2.4 Performance Schema System Variables

## Contents

1. performance\_schema
2. performance\_schema\_accounts\_size
3. performance\_schema\_digests\_size
4. performance\_schema\_events\_stages histo
5. performance\_schema\_events\_stages histo
6. performance\_schema\_events\_statements\_
7. performance\_schema\_events\_statements\_
8. performance\_schema\_events\_transactions\_
9. performance\_schema\_events\_transactions\_
10. performance\_schema\_events\_waits\_history
11. performance\_schema\_events\_waits\_history
12. performance\_schema\_hosts\_size
13. performance\_schema\_max\_cond\_classes
14. performance\_schema\_max\_cond\_instances
15. performance\_schema\_max\_digest\_length
16. performance\_schema\_max\_file\_classes
17. performance\_schema\_max\_file\_handles
18. performance\_schema\_max\_file\_instances
19. performance\_schema\_max\_index\_stat
20. performance\_schema\_max\_memory\_classes
21. performance\_schema\_max\_metadata\_locks
22. performance\_schema\_max\_mutex\_classes
23. performance\_schema\_max\_mutex\_instances
24. performance\_schema\_max\_prepared\_state
25. performance\_schema\_max\_program\_instances
26. performance\_schema\_max\_rwlock\_classes
27. performance\_schema\_max\_rwlock\_instances
28. performance\_schema\_max\_socket\_classes
29. performance\_schema\_max\_socket\_instances
30. performance\_schema\_max\_sql\_text\_length
31. performance\_schema\_max\_stage\_classes
32. performance\_schema\_max\_statement\_classes
33. performance\_schema\_max\_statement\_instances
34. performance\_schema\_max\_table\_handles
35. performance\_schema\_max\_table\_instances
36. performance\_schema\_max\_table\_lock\_stat
37. performance\_schema\_max\_thread\_classes
38. performance\_schema\_max\_thread\_instances
39. performance\_schema\_session\_connect\_attributes
40. performance\_schema\_setup\_actors\_size
41. performance\_schema\_setup\_objects\_size
42. performance\_schema\_users\_size

The following variables are used with MariaDB's [Performance Schema](#). See [Performance Schema Options](#) for Performance Schema options that are not system variables. See [Server System Variables](#) for a complete list of system variables and instructions on setting them.

See also the [Full list of MariaDB options, system and status variables](#).

### performance\_schema

- **Description:** If set to

1  
(  
0

is default), enables the Performance Schema

- **Commandline:**

--performance-schema=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

## performance\_schema\_accounts\_size

- **Description:** Maximum number of rows in the [performance\\_schema.accounts](#) table. If set to 0, the Performance Schema will not store statistics in the accounts table. Use

-1  
(the default) for automated sizing.

- **Commandline:**

--performance-schema-accounts-size=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1  
to  
1048576

## performance\_schema\_digests\_size

- **Description:** Maximum number of rows that can be stored in the [events\\_statements\\_summary\\_by\\_digest](#) table.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**

--performance-schema-digests-size=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1  
to  
200

## performance\_schema\_events\_stages\_history\_long\_size

- **Description:** Number of rows in the [events\\_stages\\_history\\_long](#) table.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**

--performance-schema-events-stages-history-long-size=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

```
-1  
to  
1048576
```

---

performance\_schema\_events\_stages\_history\_size

- **Description:** Number of rows per thread in the [events\\_stages\\_history](#) table.

```
0  
for disabling,  
-1  
(the default) for automated sizing.
```

- **Commandline:**

```
--performance-schema-events-stages-history-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
numeric
```

- **Default Value:**

```
-1
```

- **Range:**

```
-1  
to  
1024
```

---

performance\_schema\_events\_statements\_history\_long\_size

- **Description:** Number of rows in the [events\\_statements\\_history\\_long](#) table.

```
0  
for disabling,  
-1  
(the default) for automated sizing.
```

- **Commandline:**

```
--performance-schema-events-statements-history-long-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
numeric
```

- **Default Value:**

```
-1
```

- **Range:**

```
-1  
to  
1048576
```

---

performance\_schema\_events\_statements\_history\_size

- **Description:** Number of rows per thread in the [events\\_statements\\_history](#) table.

```
0  
for disabling,  
-1  
(the default) for automated sizing.
```

- **Commandline:**

```
--performance-schema-events-statements-history-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    -1

- **Range:**

    -1

    to

    1024

---

## performance\_schema\_events\_transactions\_history\_long\_size

- **Description:** Number of rows in [events\\_transactions\\_history\\_long](#) table. Use

    0

    to disable,

    -1

    for automated sizing.

- **Commandline:**

```
--performance-schema-events-transactions-history-long-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    -1

- **Range:**

    -1

    to

    1048576

- **Introduced:** [MariaDB 10.5.2](#)

---

## performance\_schema\_events\_transactions\_history\_size

- **Description:** Number of rows per thread in [events\\_transactions\\_history](#). Use 0 to disable, -1 for automated sizing.

- **Commandline:**

```
--performance-schema-events-transactions-history-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    -1

- **Range:**

    -1

    to

    1024

- **Introduced:** [MariaDB 10.5.2](#)

---

## performance\_schema\_events\_waits\_history\_long\_size

- **Description:** Number of rows contained in the [events\\_waits\\_history\\_long](#) table.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**

```
--performance-schema-events-waits-history-long-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1  
to  
1048576

---

## performance\_schema\_events\_waits\_history\_size

- **Description:** Number of rows per thread contained in the [events\\_waits\\_history](#) table.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**

```
--performance-schema-events-waits-history-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1  
to  
1024

---

## performance\_schema\_hosts\_size

- **Description:** Number of rows stored in the [hosts](#) table. If set to zero, no connection statistics are kept for the hosts table.

-1  
(the default) for automated sizing.

- **Commandline:**

```
--performance-schema-hosts-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1  
to  
1048576

---

## performance\_schema\_max\_cond\_classes

- **Description:** Specifies the maximum number of condition instruments.
- **Commandline:**

--performance-schema-max-cond-classes=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**

numeric

- **Default Value:**

90  
(>= MariaDB 10.5.1 ),  
80  
(<= MariaDB 10.5.0 )

- **Range:**

0  
to  
256

---

## performance\_schema\_max\_cond\_instances

- **Description:** Specifies the maximum number of instrumented condition objects.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**

--performance-schema-max-cond-instances=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1  
to  
1048576

---

## performance\_schema\_max\_digest\_length

- **Description:** Maximum length considered for digest text, when stored in performance\_schema tables.
- **Commandline:**

--performance-schema-max-digest-length=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**

numeric

- **Default Value:**

1024

- **Range:**

0

to

1048576

---

### performance\_schema\_max\_file\_classes

- **Description:** Specifies the maximum number of file instruments.

- **Commandline:**

--performance-schema-max-file-classes=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

80

(>= MariaDB 10.5.2 ),

50

(<= MariaDB 10.5.1 )

- **Range:**

0

to

256

---

### performance\_schema\_max\_file\_handles

- **Description:** Specifies the maximum number of opened file objects. Should always be higher than [open\\_files\\_limit](#).

- **Commandline:**

--performance-schema-max-file-handles=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

32768

- **Range:**

-1

to

32768

---

### performance\_schema\_max\_file\_instances

- **Description:** Specifies the maximum number of instrumented file objects.

0

for disabling,

-1

(the default) for automated sizing.

- **Commandline:**

--performance-schema-max-file-instances=#

- **Scope:** Global

- **Dynamic:** No
- **Data Type:** numeric

- **Default Value:**

-1

- **Range:**

-1

to

1048576

---

### performance\_schema\_max\_index\_stat

- **Description:** Maximum number of index statistics for instrumented tables. Use 0 to disable, -1 for automated scaling.

- **Commandline:**

--performance-schema-max-index-stat=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1

to

1048576

- **Introduced:** MariaDB 10.5.2
- 

### performance\_schema\_max\_memory\_classes

- **Description:** Maximum number of memory pool instruments.

- **Commandline:**

--performance-schema-max-memory-classes=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

320

- **Range:**

0

to

1024

- **Introduced:** MariaDB 10.5.2
- 

### performance\_schema\_max\_metadata\_locks

- **Description:** Maximum number of [Performance Schema metadata locks](#). Use 0 to disable, -1 for automated scaling.

- **Commandline:**

--performance-schema-max-metadata-locks=#

- **Scope:** Global

- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**  
-1

- **Range:**  
-1  
to  
104857600

- **Introduced:** MariaDB 10.5.2
- 

### performance\_schema\_max\_mutex\_classes

- **Description:** Specifies the maximum number of mutex instruments.

- **Commandline:**  
--performance-schema-max-mutex-classes=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**  
210  
(>= MariaDB 10.5.2 ),  
200  
(<= MariaDB 10.5.1 )

- **Range:**  
0  
to  
256
- 

### performance\_schema\_max\_mutex\_instances

- **Description:** Specifies the maximum number of instrumented mutex instances.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**  
--performance-schema-max-mutex-instances=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**  
-1

- **Range:**  
-1  
to  
104857600
- 

### performance\_schema\_max\_prepared\_statement\_instances

- **Description:** Maximum number of instrumented prepared statements. Use 0 to disable, -1 for automated scaling.
  - **Commandline:**

```
--performance-schema-max-prepared-statement-instances=#
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**

```
numeric
```
  - **Default Value:**

```
-1
```
  - **Range:**

```
-1  
to  
1048576
```
  - **Introduced:** [MariaDB 10.5.2](#)
- 

### performance\_schema\_max\_program\_instances

- **Description:** Maximum number of instrumented programs. Use 0 to disable, -1 for automated scaling.
  - **Commandline:**

```
--performance-schema-max-program-instances=#
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**

```
numeric
```
  - **Default Value:**

```
-1
```
  - **Range:**

```
-1  
to  
1048576
```
  - **Introduced:** [MariaDB 10.5.2](#)
- 

### performance\_schema\_max\_rwlock\_classes

- **Description:** Specifies the maximum number of rwlock instruments.
  - **Commandline:**

```
--performance-schema-max-rwlock-classes=#
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**

```
numeric
```
  - **Default Value:**

```
50  
(>= MariaDB 10.5.2 ),  
40  
(<= MariaDB 10.5.1 )
```
  - **Range:**

```
0  
to  
256
```
- 

### performance\_schema\_max\_rwlock\_instances

- **Description:** Specifies the maximum number of instrumented rwlock objects.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**

--performance-schema-max-rwlock-instances=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1  
to  
104857600

---

#### performance\_schema\_max\_socket\_classes

- **Description:** Specifies the maximum number of socket instruments.

- **Commandline:**

--performance-schema-max-socket-classes=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

10

- **Range:**

0  
to  
256

---

#### performance\_schema\_max\_socket\_instances

- **Description:** Specifies the maximum number of instrumented socket objects.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**

--performance-schema-max-socket-instances=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1  
to  
1048576

---

## performance\_schema\_max\_sql\_text\_length

- **Description:** Maximum length of displayed sql text.

- **Commandline:**

```
--performance-schema-max-sql-text-length=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
numeric
```

- **Default Value:**

```
1024
```

- **Range:**

```
0
```

```
to
```

```
1048576
```

- **Introduced:** [MariaDB 10.5.2](#)
- 

## performance\_schema\_max\_stage\_classes

- **Description:** Specifies the maximum number of stage instruments.

- **Commandline:**

```
--performance-schema-max-stage-classes=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
numeric
```

- **Default Value:**

```
160  
(>= MariaDB 10.3.3 ),
```

```
150  
(<= MariaDB 10.3.2 )
```

- **Range:**

```
0
```

```
to
```

```
256
```

---

## performance\_schema\_max\_statement\_classes

- **Description:** Specifies the maximum number of statement instruments. Automatically calculated at server build based on the number of available statements. Should be left as either autosized or disabled, as changing to any positive value has no benefit and will most likely allocate unnecessary memory. Setting to zero disables all statement instrumentation, and no memory will be allocated for this purpose.

- **Commandline:**

```
--performance-schema-max-statement-classes=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
numeric
```

- **Default Value:** Autosized (see description)

- **Range:**

```
0
```

```
to
```

---

`performance_schema_max_statement_stack`

- **Description:** Number of rows per thread in EVENTS\_STATEMENTS\_CURRENT.
- **Commandline:**

--performance-schema-max-statement-stack=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

10

- **Range:**

1

to

256

- **Introduced:** MariaDB 10.5.2
- 

`performance_schema_max_table_handles`

- **Description:** Specifies the maximum number of opened table objects.

0

for disabling,

-1

(the default) for automated sizing.

- **Commandline:**

--performance-schema-max-table-handles=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1

to

1048576

---

`performance_schema_max_table_instances`

- **Description:** Specifies the maximum number of instrumented table objects.

0

for disabling,

-1

(the default) for automated sizing.

- **Commandline:**

--performance-schema-max-table-instances=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1

to

1048576

---

### performance\_schema\_max\_table\_lock\_stat

- **Description:** Maximum number of lock statistics for instrumented tables. Use 0 to disable, -1 for automated scaling.

- **Commandline:**

--performance-schema-max-table-lock-stat=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

-1

- **Range:**

-1

to

1048576

- **Introduced:** MariaDB 10.5.2
- 

### performance\_schema\_max\_thread\_classes

- **Description:** Specifies the maximum number of thread instruments.

- **Commandline:**

--performance-schema-max-thread-classes=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

50

- **Range:**

0

to

256

---

### performance\_schema\_max\_thread\_instances

- **Description:** Specifies how many of the running server threads (see [max\\_connections](#) and [max\\_delayed\\_threads](#) ) can be instrumented. Should be greater than the sum of max\_connections and max\_delayed\_threads.

0

for disabling,

-1

(the default) for automated sizing.

- **Commandline:**

--performance-schema-max-thread-instances=#

---

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    numeric

- **Default Value:**  
    -1

- **Range:**

-1  
to  
1048576

---

### performance\_schema\_session\_connect\_attrs\_size

- **Description:** Per thread preallocated memory for holding connection attribute strings. Incremented if the strings are larger than the reserved space.

0  
for disabling,  
-1  
(the default) for automated sizing.

- **Commandline:**

--performance-schema-session-connect-attrs-size=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    numeric

- **Default Value:**  
    -1

- **Range:**

-1  
to  
1048576

---

### performance\_schema\_setup\_actors\_size

- **Description:** The maximum number of rows to store in the performance schema `setup_actors` table.

-1  
(from [MariaDB 10.5.2](#) ) denotes automated sizing.

- **Commandline:**

--performance-schema-setup-actors-size=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    numeric

- **Default Value:**

-1  
(>= [MariaDB 10.5.2](#) ),  
100  
(<= [MariaDB 10.5.1](#) )

- **Range:**

-1  
to  
1024  
(>= [MariaDB 10.5.2](#) ),  
0  
to  
1024

### performance\_schema\_setup\_objects\_size

- **Description:** The maximum number of rows that can be stored in the performance schema `setup_objects` table.
    - 1
      - (from MariaDB 10.5.2) denotes automated sizing.
  - **Commandline:**

```
--performance-schema-setup-objects-size=#
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**

```
numeric
```
  - **Default Value:**

```
-1  
(>= MariaDB 10.5.2),  
100  
(<= MariaDB 10.5.1)
```
  - **Range:**

```
-1  
to  
1048576  
(>= MariaDB 10.5.2),  
0  
to  
1048576  
(<= MariaDB 10.5.1)
```
- 

### performance\_schema\_users\_size

- **Description:** Number of rows in the `performance_schema.users` table. If set to 0, the Performance Schema will not store connection statistics in the users table.
  - 1
    - (the default) for automated sizing.
- **Commandline:**

```
--performance-schema-users-size=#
```
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**

```
numeric
```
- **Default Value:**

```
-1
```
- **Range:**

```
-1  
to  
1048576
```

## 1.1.2.9.2.5 Performance Schema Digests

The Performance Schema digest is a normalized form of a statement, with the specific data values removed. It allows statistics to be gathered for similar kinds of statements.

For example:

```
SELECT * FROM customer WHERE age < 20
SELECT * FROM customer WHERE age < 30
```

With the data values removed, both of these statements normalize to:

```
SELECT * FROM customer WHERE age < ?
```

which is the digest text. The digest text is then MD5 hashed, resulting in a digest. For example:

```
DIGEST_TEXT: SELECT * FROM `performance_schema` . `users`  
DIGEST: 0f70cec4015f2a346df4ac0e9475d9f1
```

By contrast, the following two statements would not have the same digest as, while the data values are the same, they call upon different tables.

```
SELECT * FROM customer1 WHERE age < 20  
SELECT * FROM customer2 WHERE age < 20
```

The digest text is limited to 1024 bytes. Queries exceeding this limit are truncated with '...', meaning that long queries that would otherwise have different digests may share the same digest.

Digest information is used in a number of performance scheme tables. These include

- `events_statements_current`
- `events_statements_history`
- `events_statements_history_long`
- `events_statements_summary_by_digest` (a summary table by schema and digest)

## 1.1.2.9.2.6 PERFORMANCE\_SCHEMA Storage Engine

If you run `SHOW ENGINES`, you'll see the following storage engine listed.

```
SHOW ENGINES\G  
...  
Engine: PERFORMANCE_SCHEMA  
Support: YES  
Comment: Performance Schema  
Transactions: NO  
XA: NO  
Savepoints: NO  
...
```

The PERFORMANCE\_SCHEMA is not a regular storage engine for storing data, it's a mechanism for implementing the [Performance Schema](#) feature.

The `SHOW ENGINE PERFORMANCE_SCHEMA STATUS` statement is also available, which shows how much memory is used by the tables and internal buffers.

See [Performance Schema](#) for more details.

## 1.1.2.9.3 The mysql Database Tables

### 1.1.2.9.3.1 mysql.column\_stats Table

The

`mysql.column_stats` table is one of three tables storing data used for [Engine-independent table statistics](#). The others are `mysql.table_stats` and `mysql.index_stats`.

Note that statistics for blob and text columns are not collected. If explicitly specified, a warning is returned.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.column_stats` table contains the following fields:

Field	Type	Null	Key	Default	Description
-------	------	------	-----	---------	-------------

db_name	varchar(64)	NO	PRI	NULL	Database the table is in.
table_name	varchar(64)	NO	PRI	NULL	Table name.
column_name	varchar(64)	NO	PRI	NULL	Name of the column.
min_value	varchar(255)	YES		NULL	Minimum value in the table (in text form).
max_value	varchar(255)	YES		NULL	Maximum value in the table (in text form).
nulls_ratio	decimal(12,4)	YES		NULL	Fraction of NULL values (0- no NULL s, 0.5 - half values are NULL s, 1 - all values are NULL s).
avg_length	decimal(12,4)	YES		NULL	Average length of column value, in bytes. Counted as if one ran  SELECT AVG(LENGTH(col)) . This doesn't count NULL bytes, assumes endspace removal for CHAR(n) , etc.
avg_frequency	decimal(12,4)	YES		NULL	Average number of records with the same value
hist_size	tinyint(3) unsigned	YES		NULL	Histogram size in bytes, from 0-255, or, from MariaDB 10.7 , number of buckets if the histogram type is JSON_HB .
hist_type	enum('SINGLE_PREC_HB', 'DOUBLE_PREC_HB') (>= MariaDB 10.7 )  enum('SINGLE_PREC_HB', 'DOUBLE_PREC_HB','JSON_HB') (<= MariaDB 10.6 )	YES		NULL	Histogram type. See the histogram_type system variable.
histogram	blob (>= MariaDB 10.7 )  varbinary(255) (<= MariaDB 10.6 )	YES		NULL	

It is possible to manually update the table. See [Manual updates to statistics tables](#) for details.

## 1.1.2.9.3.2 mysql.columns\_priv Table

The

### `mysql.columns_priv`

table contains information about column-level privileges. The table can be queried and although it is possible to directly update it, it is best to use [GRANT](#) for setting privileges.

Note that the MariaDB privileges occur at many levels. A user may be granted a privilege at the column level, but may still not have permission on a table level, for example. See [privileges](#) for a more complete view of the MariaDB privilege system.

The [INFORMATION\\_SCHEMA.COLUMN\\_PRIVILEGES](#) table derives its contents from

### `mysql.columns_priv`

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

### `mysql.columns_priv`

table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User, Db, Table_name and Column_name makes up the unique identifier for this record.)
Db	char(64)	NO	PRI		Database name (together with User, Host, Table_name and Column_name makes up the unique identifier for this record.)
User	char(80)	NO	PRI		User (together with Host, Db, Table_name and Column_name makes up the unique identifier for this record.)
Table_name	char(64)	NO	PRI		Table name (together with User, Db, Host and Column_name makes up the unique identifier for this record.)

Column_name	char(64)	NO	PRI		Column name (together with User , Db , Table_name and Host makes up the unique identifier for this record.)
Timestamp	timestamp	NO		CURRENT_TIMESTAMP	
Column_priv	set('Select', 'Insert', 'Update', 'References')	NO			The privilege type. See <a href="#">Column Privileges</a> for details.

The [Acl\\_column\\_grants](#) status variable, added in [MariaDB 10.1.4](#), indicates how many rows the

```
mysql.columns_priv
table contains.
```

## 1.1.2.9.3.3 mysql.db Table

The

```
mysql.db
```

table contains information about database-level privileges. The table can be queried and although it is possible to directly update it, it is best to use [GRANT](#) for setting privileges.

Note that the MariaDB privileges occur at many levels. A user may not be granted a privilege at the database level, but may still have permission on a table level, for example. See [privileges](#) for a more complete view of the MariaDB privilege system.

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

The

```
mysql.db
```

table contains the following fields:

Field	Type	Null	Key	Default	Description	Introduced
Host	char(60)	NO	PRI		Host (together with User and Db makes up the unique identifier for this record. Until <a href="#">MariaDB 5.5</a> , if the host field was blank, the corresponding record in the <a href="#">mysql.host</a> table would be examined. From <a href="#">MariaDB 10.0</a> , a blank host field is the same as the % wildcard.)	
Db	char(64)	NO	PRI		Database (together with User and Host makes up the unique identifier for this record.)	
User	char(80)	NO	PRI		User (together with Host and Db makes up the unique identifier for this record.)	

Select_priv	enum('N','Y')	NO	N	Can perform <b>SELECT</b> statements.	
Insert_priv	enum('N','Y')	NO	N	Can perform <b>INSERT</b> statements.	
Update_priv	enum('N','Y')	NO	N	Can perform <b>UPDATE</b> statements.	
Delete_priv	enum('N','Y')	NO	N	Can perform <b>DELETE</b> statements.	
Create_priv	enum('N','Y')	NO	N	Can <b>CREATE TABLE</b> 's .	
Drop_priv	enum('N','Y')	NO	N	Can <b>DROP DATABASE</b> 's or <b>DROP TABLE</b> 's .	
Grant_priv	enum('N','Y')	NO	N	User can <b>grant</b> privileges they possess.	
References_priv	enum('N','Y')	NO	N	Unused	
Index_priv	enum('N','Y')	NO	N	Can create an index on a table using the <b>CREATE INDEX</b> statement. Without the <b>INDEX</b> privilege, user can still create indexes when creating a table using the <b>CREATE TABLE</b> statement if the user has have the <b>CREATE</b> privilege, and user can create indexes using the <b>ALTER</b> <b>TABLE</b> statement if they have the <b>ALTER</b> privilege.	
Alter_priv	enum('N','Y')	NO	N	Can perform <b>ALTER TABLE</b> statements.	
Create_tmp_table_priv	enum('N','Y')	NO	N	Can create temporary tables with the <b>CREATE TEMPORARY TABLE</b> statement.	
Lock_tables_priv	enum('N','Y')	NO	N	Acquire explicit locks using the <b>LOCK TABLES</b> statement; user also needs to have the <b>SELECT</b> privilege on a table in order to lock it.	
Create_view_priv	enum('N','Y')	NO	N	Can create a view using the <b>CREATE_VIEW</b> statement.	
Show_view_priv	enum('N','Y')	NO	N	Can show the <b>CREATE VIEW</b> statement to create a view using the <b>SHOW</b> <b>CREATE VIEW</b> statement.	

Create_routine_priv	enum('N','Y')	NO	N	Can create stored programs using the <a href="#">CREATE PROCEDURE</a> and <a href="#">CREATE FUNCTION</a> statements.	
Alter_routine_priv	enum('N','Y')	NO	N	Can change the characteristics of a stored function using the <a href="#">ALTER FUNCTION</a> statement.	
Execute_priv	enum('N','Y')	NO	N	Can execute <a href="#">stored procedure</a> or functions.	
Event_priv	enum('N','Y')	NO	N	Create, drop and alter <a href="#">events</a> .	
Trigger_priv	enum('N','Y')	NO	N	Can execute <a href="#">triggers</a> associated with tables the user updates, execute the <a href="#">CREATE TRIGGER</a> and <a href="#">DROP TRIGGER</a> statements.	
Delete_history_priv	enum('N','Y')	NO	N	Can delete rows created through <a href="#">system versioning</a> .	MariaDB 10.3.5

The [Acl\\_database\\_grants](#) status variable, added in [MariaDB 10.1.4](#), indicates how many rows the `mysql.db` table contains.

## 1.1.2.9.3.4 mysql.event Table

The

`mysql.event`

table contains information about MariaDB [events](#). Similar information can be obtained by viewing the [INFORMATION\\_SCHEMA.EVENTS](#) table, or with the [SHOW EVENTS](#) and [SHOW CREATE EVENT](#) statements.

The table is upgraded live, and there is no need to restart the server if the table has changed.

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.event`

table contains the following fields:

Field	Type	Null	Key	Default	Description
db	char(64)	NO	PRI		
name	char(64)	NO	PRI		
body	longblob	NO		NULL	
definer	char(141)	NO			

execute_at	datetime	YES		NULL	
interval_value	int(11)	YES		NULL	
interval_field	enum('YEAR', 'QUARTER', 'MONTH', 'DAY', 'HOUR', 'MINUTE', 'WEEK', 'SECOND', 'MICROSECOND', 'YEAR_MONTH', 'DAY_HOUR', 'DAY_MINUTE', 'DAY_SECOND', 'HOUR_MINUTE', 'HOUR_SECOND', 'MINUTE_SECOND', 'DAY_MICROSECOND', 'HOUR_MICROSECOND', 'MINUTE_MICROSECOND', 'SECOND_MICROSECOND')	YES		NULL	
created	timestamp	NO		CURRENT_TIMESTAMP	
modified	timestamp	NO		0000-00-00 00:00:00	
last_executed	datetime	YES		NULL	
starts	datetime	YES		NULL	
ends	datetime	YES		NULL	
status	enum('ENABLED', 'DISABLED', 'SLAVESIDE_DISABLED')	NO		ENABLED	Current status of the event, one of enabled, disabled, or disabled on the slaveside.
on_completion	enum('DROP', 'PRESERVE')	NO		DROP	
sql_mode	set('REAL_AS_FLOAT', 'PIPES_AS_CONCAT', 'ANSI_QUOTES', 'IGNORE_SPACE', 'IGNORE_BAD_TABLE_OPTIONS', 'ONLY_FULL_GROUP_BY', 'NO_UNSIGNED_SUBTRACTION', 'NO_DIR_IN_CREATE', 'POSTGRESQL', 'ORACLE', 'MSSQL', 'DB2', 'MAXDB', 'NO_KEY_OPTIONS', 'NO_TABLE_OPTIONS', 'NO_FIELD_OPTIONS', 'MYSQL323', 'MYSQL40', 'ANSI', 'NO_AUTO_VALUE_ON_ZERO', 'NO_BACKSLASH_ESCAPES', 'STRICT_TRANS_TABLES', 'STRICT_ALL_TABLES', 'NO_ZERO_IN_DATE', 'NO_ZERO_DATE', 'INVALID_DATES', 'ERROR_FOR_DIVISION_BY_ZERO', 'TRADITIONAL', 'NO_AUTO_CREATE_USER', 'HIGH_NOT_PRECEDENCE', 'NO_ENGINE_SUBSTITUTION', 'PAD_CHAR_TO_FULL_LENGTH')	NO			The <a href="#">SQL_MODE</a> at the time the event was created.
comment	char(64)	NO			
originator	int(10) unsigned	NO		NULL	

time_zone	char(64)	NO		SYSTEM	
character_set_client	char(32)	YES		NULL	
collation_connection	char(32)	YES		NULL	
db_collation	char(32)	YES		NULL	
body_utf8	longblob	YES		NULL	

## 1.1.2.9.3.5 mysql.func Table

The

`mysql.func`

table stores information about [user-defined functions](#) (UDFs) created with the [CREATE FUNCTION UDF](#) statement.

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.func`

table contains the following fields:

Field	Type	Null	Key	Default	Description
name	char(64)	NO	PRI		UDF name
ret	tinyint(1)	NO		0	
d1	char(128)	NO			Shared library name
type	enum('function','aggregate')	NO		NULL	Type, either function or aggregate . Aggregate functions are summary functions such as <a href="#">SUM()</a> and <a href="#">AVG()</a> .

## Example

```
SELECT * FROM mysql.func;
+-----+-----+-----+
| name | ret | dl | type |
+-----+-----+-----+
| spider_direct_sql | 2 | ha_spider.so | function |
| spider_bg_direct_sql | 2 | ha_spider.so | aggregate |
| spider_ping_table | 2 | ha_spider.so | function |
| spider_copy_tables | 2 | ha_spider.so | function |
| spider_flush_table_mon_cache | 2 | ha_spider.so | function |
+-----+-----+-----+
```

## 1.1.2.9.3.6 mysql.general\_log Table

The

`mysql.general_log`

table stores the contents of the [General Query Log](#) if general logging is active and the output is being written to table (see [Writing logs into tables](#)).

It contains the following fields:

Field	Type	Null	Key	Default	Description
event_time	timestamp(6)	NO		CURRENT_TIMESTAMP(6)	Time the query was executed.
user_host	mediumtext	NO		NULL	User and host combination.
thread_id	int(11)	NO		NULL	Thread id.
server_id	int(10) unsigned	NO		NULL	Server id.
command_type	varchar(64)	NO		NULL	Type of command.
argument	mediumtext	NO		NULL	Full query.

## Example

```
SELECT * FROM mysql.general_log\G
*****
1. row *****
event_time: 2014-11-11 08:40:04.117177
user_host: root[root] @ localhost []
thread_id: 74
server_id: 1
command_type: Query
argument: SELECT * FROM test.s
*****
2. row *****
event_time: 2014-11-11 08:40:10.501131
user_host: root[root] @ localhost []
thread_id: 74
server_id: 1
command_type: Query
argument: SELECT * FROM mysql.general_log
...
```

## 1.1.2.9.3.7 mysql.global\_priv Table

MariaDB starting with [10.4.1](#)

The

`mysql.global_priv`

table was introduced in [MariaDB 10.4.1](#) to replace the `mysql.user` table.

The

**mysql.global\_priv**  
table contains information about users that have permission to access the MariaDB server, and their global privileges.

Note that the MariaDB privileges occur at many levels. A user may not be granted

privilege at the user level, but may still have  
create  
permission on certain tables or databases, for example. See [privileges](#) for a more complete view of the MariaDB privilege system.

The

`mysql.global_priv`  
table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User makes up the unique identifier for this account).
User	char(80)	NO	PRI		User (together with Host makes up the unique identifier for this account).
Priv	longtext	NO			Global privileges, granted to the account and other account properties

From MariaDB 10.5.2, in order to help the server understand which version a privilege record was written by, the

```
priv  
field contains a new JSON field,  
version_id  
( MDEV-21704 ).
```

## Examples

```
select * from mysql.global_priv;
+-----+-----+-----+
| Host      | User       | Priv
+-----+-----+-----+
| localhost | root        | {"access": 18446744073709551615,"plugin":"mysql_native_password","authentication_string":"*6C387FC38
| 127.%     | msandbox    | {"access":1073740799,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA1
| localhost | msandbox    | {"access":1073740799,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA1
| localhost | msandbox_rw | {"access":487487,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E7
| 127.%     | msandbox_rw | {"access":487487,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E7
| 127.%     | msandbox_ro | {"access":262145,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E7
| localhost | msandbox_ro | {"access":262145,"plugin":"mysql_native_password","authentication_string":"*6C387FC3893DBA1E3BA155E7
| 127.%     | rsandbox    | {"access":524288,"plugin":"mysql_native_password","authentication_string":"*B07EB15A2E7BD9620DAE47B1
+-----+-----+-----+
```

Readable format:

A particular user:

```

SELECT CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)) FROM mysql.global_priv
WHERE user='marijn';
+-----+
| CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)) |
+-----+
| marijn@localhost => {
  "access": 0,
  "plugin": "mysql_native_password",
  "authentication_string": "",
  "account_locked": true,
  "password_last_changed": 1558017158
} |
+-----+

```

From MariaDB 10.5.2 :

```

GRANT FILE ON *.* TO user1@localhost;
SELECT Host, User, JSON_DETAILED(Priv) FROM mysql.global_priv WHERE user='user1'\G

***** 1. row *****
    Host: localhost
    User: user1
JSON_DETAILED(Priv): {
  "access": 512,
  "plugin": "mysql_native_password",
  "authentication_string": "",
  "password_last_changed": 1581070979,
  "version_id": 100502
}

```

## 1.1.2.9.3.8 mysql.gtid\_slave\_pos Table

The

`mysql.gtid_slave_pos`

table is used in [replication](#) by replica servers to keep track of their current position (the [global transaction ID](#) of the last transaction applied).

Using the table allows the replica to maintain a consistent value for the `gtid_slave_pos` system variable across server restarts. See [Global Transaction ID](#).

You should never attempt to modify the table directly. If you do need to change the global `gtid_slave_pos` value, use

```
SET GLOBAL gtid_slave_pos = ...
instead.
```

The table is updated with the new position as part of each transaction committed during replication. This makes it preferable that the table is using the same storage engine as the tables otherwise being modified in the transaction, since otherwise a multi-engine transaction is needed that can reduce performance.

Starting from MariaDB 10.3.1 , multiple versions of this table are supported, each using a different storage engine. This is selected with the [gtid\\_pos\\_auto\\_engines option](#) , by giving a comma-separated list of engine names. The server will then on-demand create an extra version of the table using the appropriate storage engine, and select the table version using the same engine as the rest of the transaction, avoiding multi-engine transactions.

For example, when

```
gtid_pos_auto_engines=innodb,rocksdb
, tables
mysql.gtid_slave_pos_InnoDB
and
mysql.gtid_slave_pos_RocksDB
will be created and used, if needed. If there is no match to the storage engine, the default
mysql.gtid_slave_pos
table will be used; this also happens if non-transactional updates (like MyISAM) are replicated, since there is then no active transaction at the
time of the
mysql.gtid_slave_pos
table update.
```

Prior to MariaDB 10.3.1 , only the default

```
mysql.gtid_slave_pos
table is available. In these versions, the table should preferably be using the storage engine that is used for most replicated transactions.
```

The default

```
mysql.gtid_slave_pos
table will be initially created using the default storage engine set for the server (which itself defaults to InnoDB). If the application load is primarily
non-transactional MyISAM or Aria tables, it can be beneficial to change the storage engine to avoid including an InnoDB update with every operation:
```

```
ALTER TABLE mysql.gtid_slave_pos ENGINE=MyISAM;
```

The

```
mysql.gtid_slave_pos
```

table should not be changed manually in any other way. From [MariaDB 10.3.1](#), it is preferable to use the `gtid_pos_auto_engines` server variable to get the GTID position updates to use the TokuDB or RocksDB storage engine.

Note that for scalability reasons, the automatic creation of a new

```
mysql.gtid_slave_posXXX
```

table happens asynchronously when the first transaction with the new storage engine is committed. So the very first few transactions will update the old version of the table, until the new version is created and available.

The table

```
mysql.gtid_slave_pos
```

contains the following fields

Field	Type	Null	Key	Default	Description
domain_id	unsigned int(10)	NO	PRI	NULL	Domain id (see <a href="#">Global Transaction ID domain ID</a> ).
sub_id	bigint(20) unsigned	NO	PRI	NULL	This field enables multiple parallel transactions within same domain_id to update this table without contention. At any instant, the replication state corresponds to records with largest sub_id for each domain_id
server_id	unsigned int(10)	NO		NULL	<a href="#">Server id</a> .
seq_no	bigint(20) unsigned	NO		NULL	Sequence number, an integer that is monotonically increasing for each new event group logged into the binlog.

From [MariaDB 10.3.1](#), some status variables are available to monitor the use of the different

```
gtid_slave_pos
```

table versions:

#### [Transactions\\_gtid\\_foreign\\_engine](#)

Number of replicated transactions where the update of the

```
gtid_slave_pos
```

table had to choose a storage engine that did not otherwise participate in the transaction. This can indicate that setting `gtid_pos_auto_engines` might be useful.

#### [Rpl\\_transactions\\_multi\\_engine](#)

Number of replicated transactions that involved changes in multiple (transactional) storage engines, before considering the update of

```
gtid_slave_pos
```

. These are transactions that were already cross-engine, independent of the GTID position update introduced by replication

#### [Transactions\\_multi\\_engine](#)

Number of transactions that changed data in multiple (transactional) storage engines. If this is significantly larger than [Rpl\\_transactions\\_multi\\_engine](#), it indicates that setting

```
gtid_pos_auto_engines
```

could reduce the need for cross-engine transactions.

## 1.1.2.9.3.9 mysql.help\_category Table

```
mysql.help_category
```

is one of the four tables used by the [HELP command](#). It is populated when the server is installed by the `fill_help_table.sql` script. The other help tables are [help\\_relation](#), [help\\_topic](#) and [help\\_keyword](#).

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the MyISAM storage engine.

The

mysql.help\_category

table contains the following fields:

Field	Type	Null	Key	Default	Description
help_category_id	smallint(5) unsigned	NO	PRI	NULL	
name	char(64)	NO	UNI	NULL	
parent_category_id	smallint(5) unsigned	YES		NULL	
url	char(128)	NO		NULL	

## Example

```
SELECT * FROM help_category;
```

help_category_id	name	parent_category_id	url
1	Geographic	0	
2	Polygon properties	34	
3	WKT	34	
4	Numeric Functions	38	
5	Plugins	35	
6	MBR	34	
7	Control flow functions	38	
8	Transactions	35	
9	Help Metadata	35	
10	Account Management	35	
11	Point properties	34	
12	Encryption Functions	38	
13	LineString properties	34	
14	Miscellaneous Functions	38	
15	Logical operators	38	
16	Functions and Modifiers for Use with GROUP BY	35	
17	Information Functions	38	
18	Comparison operators	38	
19	Bit Functions	38	
20	Table Maintenance	35	
21	User-Defined Functions	35	
22	Data Types	35	
23	Compound Statements	35	
24	Geometry constructors	34	
25	GeometryCollection properties	1	
26	Administration	35	
27	Data Manipulation	35	
28	Utility	35	
29	Language Structure	35	
30	Geometry relations	34	
31	Date and Time Functions	38	
32	WKB	34	
33	Procedures	35	
34	Geographic Features	35	
35	Contents	0	
36	Geometry properties	34	
37	String Functions	38	
38	Functions	35	
39	Data Definition	35	

## 1.1.2.9.3.10 mysql.help\_keyword Table

`mysql.help_keyword` is one of the four tables used by the [HELP command](#). It is populated when the server is installed by the `fill_help_table.sql` script. The other help tables are `help_relation`, `help_category` and `help_topic`.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.help_keyword` table contains the following fields:

Field	Type	Null	Key	Default	Description
<code>help_keyword_id</code>	<code>int(10) unsigned</code>	NO	PRI	NULL	
<code>name</code>	<code>char(64)</code>	NO	UNI	NULL	

### Example

```
SELECT * FROM help_keyword;
+-----+-----+
| help_keyword_id | name      |
+-----+-----+
|     0 | JOIN    |
|     1 | HOST    |
|     2 | REPEAT  |
|     3 | SERIALIZABLE |
|     4 | REPLACE |
|     5 | AT      |
|     6 | SCHEDULE |
|     7 | RETURNS |
|     8 | STARTS  |
|     9 | MASTER_SSL_CA |
|    10 | NCHAR   |
|    11 | COLUMNS |
|    12 | COMPLETION |
...
...
```

## 1.1.2.9.3.11 mysql.help\_relation Table

`mysql.help_relation` is one of the four tables used by the [HELP command](#). It is populated when the server is installed by the `fill_help_table.sql` script. The other help tables are `help_topic`, `help_category` and `help_keyword`.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.help_relation`

table contains the following fields:

Field	Type	Null	Key	Default	Description
help_topic_id	int(10) unsigned	NO	PRI	NULL	
help_keyword_id	int(10) unsigned	NO	PRI	NULL	

## Example

```
...  
| 106 | 456 |  
| 463 | 456 |  
| 468 | 456 |  
| 463 | 457 |  
| 194 | 458 |  
| 478 | 458 |  
| 374 | 459 |  
| 459 | 459 |  
| 39 | 460 |  
| 58 | 460 |  
| 185 | 460 |  
| 264 | 460 |  
| 269 | 460 |  
| 209 | 461 |  
| 468 | 461 |  
| 201 | 462 |  
| 468 | 463 |  
+-----+-----+
```

## 1.1.2.9.3.12 mysql.help\_topic Table

`mysql.help_topic` is one of the four tables used by the [HELP command](#). It is populated when the server is installed by the `fill_help_table.sql` script. The other help tables are `help_relation`, `help_category` and `help_keyword`.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.help_topic`

table contains the following fields:

Field	Type	Null	Key	Default	Description
help_topic_id	int(10) unsigned	NO	PRI	NULL	
name	char(64)	NO	UNI	NULL	
help_category_id	smallint(5) unsigned	NO		NULL	
description	text	NO		NULL	

example	text	NO		NULL	
url	char(128)	NO		NULL	

## Example

```

SELECT * FROM help_topic\G;
...
***** row *****
help_topic_id: 692
    name: JSON_DEPTH
help_category_id: 41
    description: JSON functions were added in MariaDB 10.2.3.

Syntax
-----
JSON_DEPTH(json_doc)

Description
-----
Returns the maximum depth of the given JSON document, or
NULL if the argument is null. An error will occur if the
argument is an invalid JSON document.
Scalar values or empty arrays or objects have a depth of 1.
Arrays or objects that are not empty but contain only
elements or member values of depth 1 will have a depth of 2.
In other cases, the depth will be greater than 2.

Examples
-----
SELECT JSON_DEPTH('[]'), JSON_DEPTH('true'),
JSON_DEPTH('{}');
+-----+-----+
| JSON_DEPTH('[]') | JSON_DEPTH('true') |
| JSON_DEPTH('{}') |
+-----+-----+
| 1 | 1 | 1 |
+-----+-----+

SELECT JSON_DEPTH('[1, 2, 3]'), JSON_DEPTH('[], {}, []');
+-----+-----+
| JSON_DEPTH('[1, 2, 3]') | JSON_DEPTH('[], {}, []') |
+-----+-----+
| 2 | 2 |
+-----+-----+

SELECT JSON_DEPTH('1, 2, [3, 4, 5, 6], 7');
+-----+
| JSON_DEPTH('1, 2, [3, 4, 5, 6], 7') |
+-----+
| 3 |
+-----+

URL: https://mariadb.com/kb/en/json_depth/
example:
    url: https://mariadb.com/kb/en/json_depth/

```

## 1.1.2.9.3.13 mysql.index\_stats Table

The

`mysql.index_stats`  
 table is one of three tables storing data used for [Engine-independent table statistics](#). The others are `mysql.column_stats` and `mysql.table_stats`.

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the MyISAM storage engine.

The

`mysql.index_stats`

table contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	varchar(64)	NO	PRI	NULL	Database the table is in.
table_name	varchar(64)	NO	PRI	NULL	Table name
index_name	varchar(64)	NO	PRI	NULL	Name of the index
prefix_arity	int(11) unsigned	NO	PRI	NULL	Index prefix length. 1 for the first keypart, 2 for the first two, and so on. InnoDB's extended keys are supported.
avg_frequency	decimal(12,4)	YES		NULL	Average number of records one will find for given values of (keypart1, keypart2, ...), provided the values will be found in the table.

It is possible to manually update the table. See [Manual updates to statistics tables](#) for details.

## 1.1.2.9.3.14 mysql.innodb\_index\_stats

### Contents

1. [Example](#)
2. [See Also](#)

The

`mysql.innodb_index_stats`

table stores data related to particular [InnoDB Persistent Statistics](#), and contains multiple rows for each index.

This table, along with the related `mysql.innodb_table_stats` table, can be manually updated in order to force or test differing query optimization plans. After updating,

```
FLUSH TABLE innodb_index_stats  
is required to load the changes.
```

`mysql.innodb_index_stats`

is not replicated, although any [ANALYZE TABLE](#) statements on the table will be by default..

It contains the following fields:

Field	Type	Null	Key	Default	Description
database_name	varchar(64)	NO	PRI	NULL	Database name.
table_name	varchar(64)	NO	PRI	NULL	Table, partition or subpartition name.
index_name	varchar(64)	NO	PRI	NULL	Index name.

last_update	timestamp	NO		current_timestamp()	Time that this row was last updated.
stat_name	varchar(64)	NO	PRI	NULL	Statistic name.
stat_value	bigint(20) unsigned	NO		NULL	Estimated statistic value.
sample_size	bigint(20) unsigned	YES		NULL	Number of pages sampled for the estimated statistic value.
stat_description	varchar(1024)	NO		NULL	Statistic description.

## Example

```

SELECT * FROM mysql.innodb_index_stats\G
***** 1. row *****
database_name: mysql
  table_name: gtid_slave_pos
  index_name: PRIMARY
last_update: 2017-08-19 20:38:34
    stat_name: n_diff_pfx01
    stat_value: 0
  sample_size: 1
stat_description: domain_id
***** 2. row *****
database_name: mysql
  table_name: gtid_slave_pos
  index_name: PRIMARY
last_update: 2017-08-19 20:38:34
    stat_name: n_diff_pfx02
    stat_value: 0
  sample_size: 1
stat_description: domain_id,sub_id
***** 3. row *****
database_name: mysql
  table_name: gtid_slave_pos
  index_name: PRIMARY
last_update: 2017-08-19 20:38:34
    stat_name: n_leaf_pages
    stat_value: 1
  sample_size: NULL
stat_description: Number of leaf pages in the index
***** 4. row *****
database_name: mysql
  table_name: gtid_slave_pos
  index_name: PRIMARY
last_update: 2017-08-19 20:38:34
    stat_name: size
    stat_value: 1
  sample_size: NULL
stat_description: Number of pages in the index
***** 5. row *****
database_name: test
  table_name: ft
  index_name: FTS_DOC_ID_INDEX
last_update: 2017-09-15 12:58:39
    stat_name: n_diff_pfx01
    stat_value: 0
  sample_size: 1
stat_description: FTS_DOC_ID
***** 6. row *****
database_name: test
  table_name: ft
  index_name: FTS_DOC_ID_INDEX
last_update: 2017-09-15 12:58:39
    stat_name: n_leaf_pages
    stat_value: 1
  sample_size: NULL
stat_description: Number of leaf pages in the index
...

```

## See Also

- [InnoDB Persistent Statistics](#)
- [mysql.innodb\\_table\\_stats](#)
- [ANALYZE TABLE](#)

## 1.1.2.9.3.15 mysql.innodb\_table\_stats

### Contents

1. [Example](#)
2. [See Also](#)

The

`mysql.innodb_table_stats`  
table stores data related to [InnoDB Persistent Statistics](#), and contains one row per table.

This table, along with the related `mysql.innodb_index_stats` table, can be manually updated in order to force or test differing query optimization plans.

After updating,

```
FLUSH TABLE innodb_table_stats  
is required to load the changes.
```

`mysql.innodb_table_stats`  
is not replicated, although any [ANALYZE TABLE](#) statements on the table will be by default..

It contains the following fields:

Field	Type	Null	Key	Default	Description
database_name	varchar(64)	NO	PRI	NULL	Database name.
table_name	varchar(64)	NO	PRI	NULL	Table, partition or subpartition name.
last_update	timestamp	NO		current_timestamp()	Time that this row was last updated.
n_rows	bigint(20) unsigned	NO		NULL	Number of rows in the table.
clustered_index_size	bigint(20) unsigned	NO		NULL	Size, in pages, of the primary index.
sum_of_other_index_sizes	bigint(20) unsigned	NO		NULL	Size, in pages, of non-primary indexes.

## Example

```
SELECT * FROM mysql.innodb_table_stats\G  
***** 1. row *****  
    database_name: mysql  
    table_name: gtid_slave_pos  
    last_update: 2017-08-19 20:38:34  
        n_rows: 0  
    clustered_index_size: 1  
sum_of_other_index_sizes: 0  
***** 2. row *****  
    database_name: test  
    table_name: ft  
    last_update: 2017-09-15 12:58:39  
        n_rows: 0  
    clustered_index_size: 1  
sum_of_other_index_sizes: 2  
...
```

## See Also

- [InnoDB Persistent Statistics](#)
- [mysql.innodb\\_index\\_stats](#)
- [ANALYZE TABLE](#)

## 1.1.2.9.3.16 mysql.password\_reuse\_check\_history Table

MariaDB starting with 10.7.0

The `mysql.password_reuse_check_history` Table is installed as part of the [password\\_reuse\\_check plugin](#), available from [MariaDB 10.7.0](#).

The

`mysql.password_reuse_check_history`

table stores old passwords, so that when a user sets a new password, it can be checked for purposes of preventing password reuse.

It contains the following fields:

Field	Type	Null	Key	Default	Description
hash	binary(64)	NO	PRI	NULL	
time	timestamp	NO	MUL	current_timestamp()	

## 1.1.2.9.3.17 mysql.plugin Table

The

`mysql.plugin`  
table can be queried to get information about installed [plugins](#).

This table only contains information about [plugins](#) that have been installed via the following methods:

- The

`INSTALL SONAME`

statement.

- The

`INSTALL PLUGIN`

statement.

- The

`mysql_plugin`

utility.

This table does not contain information about:

- Built-in plugins.
- Plugins loaded with the

`--plugin-load-add`

option.

- Plugins loaded with the

`--plugin-load`

option.

This table only contains enough information to reload the plugin when the server is restarted, which means it only contains the plugin name and the plugin library.

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.plugin`  
table contains the following fields:

Field	Type	Null	Key	Default	Description
name	varchar(64)	NO	PRI		Plugin name.
d1	varchar(128)	NO			Name of the plugin library.

## Example

```

SELECT * FROM mysql.plugin;
+-----+-----+
| name      | dl          |
+-----+-----+
| spider    | ha_spider.so |
| spider_alloc_mem | ha_spider.so |
| METADATA_LOCK_INFO | metadata_lock_info.so |
| OQGRAPH   | ha_oqgraph.so |
| cassandra | ha_cassandra.so |
| QUERY_RESPONSE_TIME | query_response_time.so |
| QUERY_RESPONSE_TIME_AUDIT | query_response_time.so |
| LOCALES    | locales.so |
| sequence   | ha_sequence.so |
+-----+-----+

```

## 1.1.2.9.3.18 mysql.proc Table

The

`mysql.proc`

table contains information about [stored procedures](#) and [stored functions](#). It contains similar information to that stored in the [INFORMATION SCHEMA.ROUTINES](#) table.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.proc`

table contains the following fields:

Field	Type	Null	Key	Default	Description
db	char(64)	NO	PRI		Database name.
name	char(64)	NO	PRI		Routine name.
type	enum('FUNCTION','PROCEDURE','PACKAGE', 'PACKAGE BODY')	NO	PRI	NULL	Whether <a href="#">stored procedure</a> , <a href="#">stored function</a> or, from MariaDB 10.3.5, a <a href="#">package</a> or <a href="#">package body</a> .
specific_name	char(64)	NO			
language	enum('SQL')	NO		SQL	Always SQL
sql_data_access	enum('CONTAINS_SQL', 'NO_SQL', 'READS_SQL_DATA', 'MODIFIES_SQL_DATA')	NO		CONTAINS_SQL	

is_deterministic	enum('YES','NO')	NO	NO	Whether the routine is deterministic (can produce only one result for a given list of parameters) or not.
security_type	enum('INVOKER','DEFINER')	NO	DEFINER	INVOKER or DEFINER Indicates which user's privileges apply to this routine.
param_list	blob	NO	NULL	List of parameters.
returns	longblob	NO	NULL	What the routine returns.
body	longblob	NO	NULL	Definition of the routine.
definer	char(141)	NO		If the security_type is DEFINER , this value indicates which user defined this routine.
created	timestamp	NO	CURRENT_TIMESTAMP	Date and time the routine was created.
modified	timestamp	NO	0000-00-00 00:00:00	Date and time the routine was modified.
sql_mode	<pre>set('REAL_AS_FLOAT', 'PIPES_AS_CONCAT', 'ANSI_QUOTES', 'IGNORE_SPACE', 'IGNORE_BAD_TABLE_OPTIONS', 'ONLY_FULL_GROUP_BY', 'NO_UNSIGNED_SUBTRACTION', 'NO_DIR_IN_CREATE', 'POSTGRESQL', 'ORACLE', 'MSSQL', 'DB2', 'MAXDB', 'NO_KEY_OPTIONS', 'NO_TABLE_OPTIONS', 'NO_FIELD_OPTIONS', 'MYSQL323', 'MYSQL40', 'ANSI', 'NO_AUTO_VALUE_ON_ZERO', 'NO_BACKSLASH_ESCAPES', 'STRICT_TRANS_TABLES', 'STRICT_ALL_TABLES', 'NO_ZERO_IN_DATE', 'NO_ZERO_DATE', 'INVALID_DATES', 'ERROR_FOR_DIVISION_BY_ZERO', 'TRADITIONAL', 'NO_AUTO_CREATE_USER', 'HIGH_NOT_PRECEDENCE', 'NO_ENGINE_SUBSTITUTION', 'PAD_CHAR_TO_FULL_LENGTH', 'EMPTY_STRING_IS_NULL', 'SIMULTANEOUS_ASSIGNMENT')</pre>	NO		The <a href="#">SQL_MODE</a> at the time the routine was created.

Field	Type	Null	Key	Default	Description
comment	text	NO		NULL	Comment associated with the routine.
character_set_client	char(32)	YES		NULL	The character set used by the client that created the routine.
collation_connection	char(32)	YES		NULL	The collation (and character set) used by the connection that created the routine.
db_collation	char(32)	YES		NULL	The default collation (and character set) for the database, at the time the routine was created.
body_utf8	longblob	YES		NULL	Definition of the routine in utf8.
aggregate	enum('NONE', 'GROUP')	NO		NONE	From MariaDB 10.3.3

## See Also

- [Stored Procedure Internals](#)
- [MySQL to MariaDB migration: handling privilege table differences when using mysqldump](#)

## 1.1.2.9.3.19 mysql.procs\_priv Table

The

`mysql.procs_priv`

table contains information about [stored procedure](#) and [stored function](#) privileges. See [CREATE PROCEDURE](#) and [CREATE FUNCTION](#) on creating these.

The [INFORMATION\\_SCHEMA.ROUTINES](#) table derives its contents from

`mysql.procs_priv`

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.procs_priv`

table contains the following fields:

Field	Type	Null	Key	Default	Description
-------	------	------	-----	---------	-------------

Host	char(60)	NO	PRI		Host (together with Db , User , Routine_name and Routine_type makes up the unique identifier for this record).
Db	char(64)	NO	PRI		Database (together with Host , User , Routine_name and Routine_type makes up the unique identifier for this record).
User	char(80)	NO	PRI		User (together with Host , Db , Routine_name and Routine_type makes up the unique identifier for this record).
Routine_name	char(64)	NO	PRI		Routine_name (together with Host , Db  User and Routine_type makes up the unique identifier for this record).
Routine_type	enum('FUNCTION','PROCEDURE','PACKAGE', 'PACKAGE BODY')	NO	PRI	NULL	Whether the routine is a <a href="#">stored procedure</a> , <a href="#">stored function</a> , or, from <a href="#">MariaDB 10.3.5</a> , a <a href="#">package</a> or <a href="#">package body</a> .
Grantor	char(141)	NO	MUL		
Proc_priv	set('Execute','Alter Routine','Grant')	NO			The routine privilege. See <a href="#">Function Privileges</a> and <a href="#">Procedure Privileges</a> for details.
Timestamp	timestamp	NO		CURRENT_TIMESTAMP	

The [Acl\\_function\\_grants](#) status variable, added in [MariaDB 10.1.4](#) , indicates how many rows the

```
mysql.columns_priv
table contains with the
FUNCTION
routine type.
```

The [Acl\\_procedure\\_grants](#) status variable, added in [MariaDB 10.1.4](#) , indicates how many rows the

```
mysql.columns_priv
table contains with the
PROCEDURE
routine type.
```

## 1.1.2.9.3.20 mysql.roles\_mapping Table

The

`mysql.roles_mapping`  
table contains information about mariaDB roles .

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the Aria storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the MyISAM storage engine.

The

`mysql.roles_mapping`  
table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User and Role makes up the unique identifier for this record.)
User	char(80)	NO	PRI		User (together with Host and Role makes up the unique identifier for this record.)
Role	char(80)	NO	PRI		Role (together with Host and User makes up the unique identifier for this record.)
Admin_option	enum('N','Y')	NO		N	Whether the role can be granted (see the CREATE ROLE WITH ADMIN clause).

The `Acl_role_grants` status variable, added in MariaDB 10.1.4 , indicates how many rows the

`mysql.roles_mapping`  
table contains.

## 1.1.2.9.3.21 mysql.servers Table

The

`mysql.servers`  
table contains information about servers as used by the Spider , FEDERATED or FederatedX , Connect storage engines (see CREATE SERVER ).

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the Aria storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the MyISAM storage engine.

The

`mysql.servers`  
table contains the following fields:

Field	Type	Null	Key	Default	Description
Server_name	char(64)	NO	PRI		
Host	char(64)	NO			

Db	char(64)	NO			
Username	char(80)	NO			
Password	char(64)	NO			
Port	int(4)	NO	0		
Socket	char(64)	NO			
Wrapper	char(64)	NO			mysql or mariadb
Owner	char(64)	NO			

## 1.1.2.9.3.22 mysql.slow\_log Table

The

`mysql.slow_log`

table stores the contents of the [Slow Query Log](#) if slow logging is active and the output is being written to table (see [Writing logs into tables](#) ).

It contains the following fields:

Field	Type	Null	Key	Default	Description
start_time	timestamp(6)	NO		CURRENT_TIMESTAMP(6)	Time the query began.
user_host	mediumtext	NO		NULL	User and host combination.
query_time	time(6)	NO		NULL	Total time the query took to execute.
lock_time	time(6)	NO		NULL	Total time the query was locked.
rows_sent	int(11)	NO		NULL	Number of rows sent.
rows_examined	int(11)	NO		NULL	Number of rows examined.
db	varchar(512)	NO		NULL	Default database.

last_insert_id	int(11)	NO		NULL	<a href="#">last_insert_id</a> .
insert_id	int(11)	NO		NULL	Insert id.
server_id	unsigned int(10)	NO		NULL	The server's id.
sql_text	mediumtext	NO		NULL	Full query.
thread_id	unsigned bigint(21)	NO		NULL	Thread id.
rows_affected	int(11)	NO		NULL	Number of rows affected by an <a href="#">UPDATE</a> or <a href="#">DELETE</a> (from <a href="#">MariaDB 10.1.2</a> )

## Example

```
SELECT * FROM mysql.slow_log\G
...
*****2. row *****
start_time: 2014-11-11 07:56:28.721519
user_host: root[root] @ localhost []
query_time: 00:00:12.000215
lock_time: 00:00:00.000000
rows_sent: 1
rows_examined: 0
db: test
last_insert_id: 0
insert_id: 0
server_id: 1
sql_text: SELECT SLEEP(12)
thread_id: 74
...
```

## 1.1.2.9.3.23 mysql.tables\_priv Table

The

`mysql.tables_priv`  
table contains information about table-level privileges. The table can be queried and although it is possible to directly update it, it is best to use [GRANT](#) for setting privileges.

Note that the MariaDB privileges occur at many levels. A user may be granted a privilege at the table level, but may still not have permission on a database level, for example. See [privileges](#) for a more complete view of the MariaDB privilege system.

The [INFORMATION\\_SCHEMA.TABLE\\_PRIVILEGES](#) table derives its contents from

`mysql.tables_priv`

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.tables_priv`

table contains the following fields:

Field	Type	Null	Key	Default	Description
Host	char(60)	NO	PRI		Host (together with User Db and Table_name makes up the unique identifier for this record.)
Db	char(64)	NO	PRI		Database (together with User Host and Table_name makes up the unique identifier for this record.)
User	char(80)	NO	PRI		User (together with Host Db and Table_name makes up the unique identifier for this record.)

Table_name	char(64)		NO	PRI		Table name (together with User, Db and Table makes up the unique identifier for this record.)
Grantor	char(141)		NO	MUL		
Timestamp	timestamp		NO		CURRENT_TIMESTAMP	
Table_priv	set('Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter', 'Create View', 'Show view', 'Trigger', 'Delete versioning rows')		NO			The table privilege type. See <a href="#">Table Privileges</a> for details.
Column_priv	set('Select', 'Insert', 'Update', 'References')		NO			The column privilege type. See <a href="#">Column Privileges</a> for details.

The [Acl\\_table\\_grants](#) status variable, added in [MariaDB 10.1.4](#), indicates how many rows the

```
mysql.tables_priv
```

table contains.

## 1.1.2.9.3.24 mysql.table\_stats Table

The

```
mysql.table_stats
```

table is one of three tables storing data used for [Engine-independent table statistics](#). The others are [mysql.column\\_stats](#) and [mysql.index\\_stats](#).

MariaDB starting with 10.4

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

The

```
mysql.table_stats
```

table contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	varchar(64)	NO	PRI	NULL	Database the table is in.
table_name	varchar(64)	NO	PRI	NULL	Table name.

cardinality	bigint(21) unsigned	YES		NULL	Number of records in the table.
-------------	---------------------	-----	--	------	---------------------------------

It is possible to manually update the table. See [Manual updates to statistics tables](#) for details.

## 1.1.2.9.3.25 mysql.time\_zone Table

The

`mysql.time_zone`

table is one of the mysql system tables that can contain [time zone](#) information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the mysql time zone tables using the `mysql_tzinfo_to_sql` utility. See [Time Zones](#) for details.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.time_zone`

table contains the following fields:

Field	Type	Null	Key	Default	Description
Time_zone_id	int(10) unsigned	NO	PRI	NULL	ID field, auto_increments.
Use_leap_seconds	enum('Y','N')	NO		N	Whether or not leap seconds are used.

## Example

```
SELECT * FROM mysql.time_zone;
+-----+-----+
| Time_zone_id | Use_leap_seconds |
+-----+-----+
|      1 | N           |
|      2 | N           |
|      3 | N           |
|      4 | N           |
|      5 | N           |
|      6 | N           |
|      7 | N           |
|      8 | N           |
|      9 | N           |
|     10 | N           |
...
+-----+-----+
```

## See Also

- [mysql.time\\_zone\\_leap\\_second table](#)
- [mysql.time\\_zone\\_name table](#)
- [mysql.time\\_zone\\_transition table](#)
- [mysql.time\\_zone\\_transition\\_type table](#)

## 1.1.2.9.3.26 mysql.time\_zone\_leap\_second Table

The

`mysql.time_zone_leap_second`

table is one of the mysql system tables that can contain [time zone](#) information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the mysql time zone tables using the `mysql_tzinfo_to_sql` utility. See [Time Zones](#) for details.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.time_zone_leap_second`

table contains the following fields:

Field	Type	Null	Key	Default	Description
Transition_time	bigint(20)	NO	PRI	NULL	
Correction	int(11)	NO		NULL	

## See Also

- [mysql.time\\_zone table](#)
- [mysql.time\\_zone\\_name table](#)
- [mysql.time\\_zone\\_transition table](#)
- [mysql.time\\_zone\\_transition\\_type table](#)

## 1.1.2.9.3.27 mysql.time\_zone\_name Table

The

`mysql.time_zone_name`

table is one of the mysql system tables that can contain [time zone](#) information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the mysql time zone tables using the [mysql\\_tzinfo\\_to\\_sql](#) utility. See [Time Zones](#) for details.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.time_zone_name`

table contains the following fields:

Field	Type	Null	Key	Default	Description
Name	char(64)	NO	PRI	NULL	Name of the time zone.
Time_zone_id	int(10) unsigned	NO	PRI	NULL	ID field, auto_increments.

## Example

```

SELECT * FROM mysql.time_zone_name;
+-----+-----+
| Name          | Time_zone_id |
+-----+-----+
| Africa/Abidjan |      1 |
| Africa/Accra   |      2 |
| Africa/Addis_Ababa | 3 |
| Africa/Algiers  |      4 |
| Africa/Asmara   |      5 |
| Africa/Asmera   |      6 |
| Africa/Bamako   |      7 |
| Africa/Bangui   |      8 |
| Africa/Banjul   |      9 |
| Africa/Bissau   |     10 |
...
+-----+-----+

```

## See Also

- [mysql.time\\_zone table](#)
- [mysql.time\\_zone\\_leap\\_second table](#)
- [mysql.time\\_zone\\_transition table](#)
- [mysql.time\\_zone\\_transition\\_type table](#)

## 1.1.2.9.3.28 mysql.time\_zone\_transition Table

The

`mysql.time_zone_transition`

table is one of the

`mysql`

system tables that can contain [time zone](#) information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the

`mysql`

time zone tables using the [mysql\\_tzinfo\\_to\\_sql](#) utility. See [Time Zones](#) for details.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.time_zone_transition`

table contains the following fields:

Field	Type	Null	Key	Default	Description
Time_zone_id	int(10) unsigned	NO	PRI	NULL	
Transition_time	bigint(20)	NO	PRI	NULL	
Transition_type_id	int(10) unsigned	NO		NULL	

## Example

```

SELECT * FROM mysql.time_zone_transition;
+-----+-----+-----+
| Time_zone_id | Transition_time | Transition_type_id |
+-----+-----+-----+
| 1 | -1830383032 | 1 |
| 2 | -1640995148 | 2 |
| 2 | -1556841600 | 1 |
| 2 | -1546388400 | 2 |
| 2 | -1525305600 | 1 |
| 2 | -1514852400 | 2 |
| 2 | -1493769600 | 1 |
| 2 | -1483316400 | 2 |
| 2 | -1462233600 | 1 |
| 2 | -1451780400 | 2 |
...
+-----+-----+-----+

```

## See Also

- [mysql.time\\_zone table](#)
- [mysql.time\\_zone\\_leap\\_second table](#)
- [mysql.time\\_zone\\_name table](#)
- [mysql.time\\_zone\\_transition\\_type table](#)

## 1.1.2.9.3.29 mysql.time\_zone\_transition\_type Table

The

`mysql.time_zone_transition_type`

table is one of the

`mysql`

system tables that can contain [time zone](#) information. It is usually preferable for the system to handle the time zone, in which case the table will be empty (the default), but you can populate the

`mysql`

time zone tables using the [mysql\\_tzinfo\\_to\\_sql](#) utility. See [Time Zones](#) for details.

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

The

`mysql.time_zone_transition_type`

table contains the following fields:

Field	Type	Null	Key	Default	Description
Time_zone_id	int(10) unsigned	NO	PRI	NULL	
Transition_type_id	int(10) unsigned	NO	PRI	NULL	
Offset	int(11)	NO		0	
Is_DST	tinyint(3) unsigned	NO		0	
Abbreviation	char(8)	NO			

## Example

```

SELECT * FROM mysql.time_zone_transition_type;
+-----+-----+-----+-----+
| Time_zone_id | Transition_type_id | Offset | Is_DST | Abbreviation |
+-----+-----+-----+-----+
| 1 | 0 | -968 | 0 | LMT |
| 1 | 1 | 0 | 0 | GMT |
| 2 | 0 | -52 | 0 | LMT |
| 2 | 1 | 1200 | 1 | GHST |
| 2 | 2 | 0 | 0 | GMT |
| 3 | 0 | 8836 | 0 | LMT |
| 3 | 1 | 10800 | 0 | EAT |
| 3 | 2 | 9000 | 0 | BEAT |
| 3 | 3 | 9900 | 0 | BEAUT |
| 3 | 4 | 10800 | 0 | EAT |
...
+-----+-----+-----+-----+

```

## See Also

- [mysql.time\\_zone table](#)
- [mysql.time\\_zone\\_leap\\_second table](#)
- [mysql.time\\_zone\\_name table](#)
- [mysql.time\\_zone\\_transition table](#)

## 1.1.2.9.3.30 mysql.transaction\_registry Table

MariaDB starting with [10.3.4](#)

The

`mysql.transaction_registry`  
table was introduced in [MariaDB 10.3.4](#) as part of [system-versioned tables](#).

The

`mysql.transaction_registry`  
table is used for transaction-precise versioning, and contains the following fields:

Field	Type	Null	Key	Default	Description
transaction_id	bigint(20) unsigned	NO	Primary	NULL	
commit_id	bigint(20) unsigned	NO	Unique	NULL	
begin_timestamp	timestamp(6)	NO	Multiple	0000-00-00 00:00:00.000000	Timestamp when the transaction began (BEGIN statement), however see <a href="#">MDEV-16024</a> .
commit	timestamp(6)	NO	Multiple	0000-00-00 00:00:00.000000	Timestamp when the transaction was committed.
isolation_level	enum('READ-UNCOMMITTED','READ-COMMITTED','REPEATABLE-READ','SERIALIZABLE')	NO		NULL	Transaction <a href="#">isolation level</a> .

## 1.1.2.9.3.31 mysql.user Table

The

`mysql.user`  
table contains information about users that have permission to access the MariaDB server, and their global privileges. The table can be queried and although it is possible to directly update it, it is best to use [GRANT](#) and [CREATE USER](#) for adding users and privileges.

Note that the MariaDB privileges occur at many levels. A user may not be granted

`create`  
`privilege` at the user level, but may still have  
`create`  
`permission` on certain tables or databases, for example. See [privileges](#) for a more complete view of the MariaDB privilege system.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

MariaDB starting with [10.4](#)

In MariaDB 10.4 and later, the [mysql.global\\_priv](#) table has replaced the [mysql.user](#) table, and [mysql.user](#) should be considered obsolete. It is now a [view](#) into [mysql.global\\_priv](#) created for compatibility with older applications and monitoring scripts. New tools are supposed to use [INFORMATION\\_SCHEMA](#) tables. From [MariaDB 10.4.13](#), the dedicated [mariadb.sys](#) user is created as the definer of the view. Previously, [root](#) was the definer, which resulted in privilege problems when this username was changed ([MDEV-19650](#)).

The

[mysql.user](#) table contains the following fields:

Field	Type	Null	Key	Default	Description	Introduced
Host	char(60)	NO	PRI		Host (together with User makes up the unique identifier for this account.)	
User	char(80)	NO	PRI		User (together with Host makes up the unique identifier for this account.)	
Password	longtext (>= <a href="#">MariaDB 10.4.1</a> ), char(41) (<= <a href="#">MariaDB 10.4.0</a> )	NO			Hashed password, generated by the <a href="#">PASSWORD()</a> function.	
Select_priv	enum('N','Y')	NO		N	Can perform <a href="#">SELECT</a> statements.	
Insert_priv	enum('N','Y')	NO		N	Can perform <a href="#">INSERT</a> statements.	
Update_priv	enum('N','Y')	NO		N	Can perform <a href="#">UPDATE</a> statements.	
Delete_priv	enum('N','Y')	NO		N	Can perform <a href="#">DELETE</a> statements.	
Create_priv	enum('N','Y')	NO		N	Can <a href="#">CREATE DATABASE</a> 's or <a href="#">CREATE TABLE</a> 's .	
Drop_priv	enum('N','Y')	NO		N	Can <a href="#">DROP DATABASE</a> 's or <a href="#">DROP TABLE</a> 's .	

Reload_priv	enum('N','Y')	NO	N	Can execute <a href="#">FLUSH</a> statements or equivalent <a href="#">mysqladmin</a> commands.	
Shutdown_priv	enum('N','Y')	NO	N	Can shut down the server with <a href="#">SHUTDOWN</a> or <a href="#">mysqladmin shutdown</a> .	
Process_priv	enum('N','Y')	NO	N	Can show information about active processes, via <a href="#">SHOW PROCESSLIST</a> or <a href="#">mysqladmin processlist</a> .	
File_priv	enum('N','Y')	NO	N	Read and write files on the server, using statements like <a href="#">LOAD DATA INFILE</a> or functions like <a href="#">LOAD_FILE()</a> . Also needed to create <a href="#">CONNECT</a> outward tables. MariaDB server must have permission to access those files.	
Grant_priv	enum('N','Y')	NO	N	User can <a href="#">grant</a> privileges they possess.	
References_priv	enum('N','Y')	NO	N	Unused	
Index_priv	enum('N','Y')	NO	N	Can create an index on a table using the <a href="#">CREATE INDEX</a> statement. Without the <a href="#">INDEX</a> privilege, user can still create indexes when creating a table using the <a href="#">CREATE TABLE</a> statement if the user has the <a href="#">CREATE</a> privilege, and user can create indexes using the <a href="#">ALTER TABLE</a> statement if they have the <a href="#">ALTER</a> privilege.	
Alter_priv	enum('N','Y')	NO	N	Can perform <a href="#">ALTER TABLE</a> statements.	
Show_db_priv	enum('N','Y')	NO	N	Can list all databases using the <a href="#">SHOW DATABASES</a> statement. Without the <a href="#">SHOW DATABASES</a> privilege, user can still issue the <a href="#">SHOW DATABASES</a> statement, but it will only list databases containing tables on which they have privileges.	
Super_priv	enum('N','Y')	NO	N	Can execute superuser statements: <a href="#">CHANGE MASTER TO</a> , <a href="#">KILL</a> (users who do not have this privilege can only <a href="#">KILL</a> their own threads), <a href="#">PURGE LOGS</a> , <a href="#">SET global system variables</a> , or the <a href="#">mysqladmin debug</a> command. Also, this permission allows the user to write data even if the <a href="#">read_only</a> startup option is set, enable or disable logging, enable or disable replication on slaves, specify a <a href="#">DEFINER</a> for statements that support that clause, connect once after reaching the <a href="#">MAX_CONNECTIONS</a> . If a statement has been specified for the <a href="#">init-connect</a> mysqld option, that command will not be executed when a user with <a href="#">SUPER</a> privileges connects to the server.	
Create_tmp_table_priv	enum('N','Y')	NO	N	Can create temporary tables with the <a href="#">CREATE TEMPORARY TABLE</a> statement.	

Lock_tables_priv	enum('N','Y')	NO		N	Acquire explicit locks using the <a href="#">LOCK TABLES</a> statement; user also needs to have the SELECT privilege on a table in order to lock it.	
Execute_priv	enum('N','Y')	NO		N	Can execute <a href="#">stored procedure</a> or functions.	
Repl_slave_priv	enum('N','Y')	NO		N	Accounts used by slave servers on the master need this privilege. This is needed to get the updates made on the master.	
Repl_client_priv	enum('N','Y')	NO		N	Can execute <a href="#">SHOW MASTER STATUS</a> and <a href="#">SHOW SLAVE STATUS</a> statements.	
Create_view_priv	enum('N','Y')	NO		N	Can create a view using the <a href="#">CREATE_VIEW</a> statement.	
Show_view_priv	enum('N','Y')	NO		N	Can show the <a href="#">CREATE VIEW</a> statement to create a view using the <a href="#">SHOW CREATE VIEW</a> statement.	
Create_routine_priv	enum('N','Y')	NO		N	Can create stored programs using the <a href="#">CREATE PROCEDURE</a> and <a href="#">CREATE FUNCTION</a> statements.	
Alter_routine_priv	enum('N','Y')	NO		N	Can change the characteristics of a stored function using the <a href="#">ALTER FUNCTION</a> statement.	
Create_user_priv	enum('N','Y')	NO		N	Can create a user using the <a href="#">CREATE USER</a> statement, or implicitly create a user with the <a href="#">GRANT</a> statement.	
Event_priv	enum('N','Y')	NO		N	Create, drop and alter <a href="#">events</a> .	
Trigger_priv	enum('N','Y')	NO		N	Can execute <a href="#">triggers</a> associated with tables the user updates, execute the <a href="#">CREATE TRIGGER</a> and <a href="#">DROP TRIGGER</a> statements.	
Create_tablespace_priv	enum('N','Y')	NO		N		
Delete_history_priv	enum('N','Y')	NO		N	Can delete rows created through <a href="#">system versioning</a> .	MariaDB 10.3.5
ssl_type	enum('', 'ANY', 'X509', 'SPECIFIED')	NO			TLS type - see <a href="#">TLS options</a> .	

Field	Type	Null	Key	Default	Description	Introduced
ssl_cipher	blob	NO		NULL	TLS cipher - see <a href="#">TLS options</a> .	
x509_issuer	blob	NO		NULL	X509 cipher - see <a href="#">TLS options</a> .	
x509_subject	blob	NO		NULL	SSL subject - see <a href="#">TLS options</a> .	
max_questions	int(11) unsigned	NO		0	Number of queries the user can perform per hour. Zero is unlimited. See <a href="#">per-account resource limits</a> .	
max_updates	int(11) unsigned	NO		0	Number of updates the user can perform per hour. Zero is unlimited. See <a href="#">per-account resource limits</a> .	
max_connections	int(11) unsigned	NO		0	Number of connections the account can start per hour. Zero is unlimited. See <a href="#">per-account resource limits</a> .	
max_user_connections	int(11)	NO		0	Number of simultaneous connections the account can have. Zero is unlimited. See <a href="#">per-account resource limits</a> .	
plugin	char(64)	NO			Authentication plugin used on connection. If empty, uses the <a href="#">default</a> .	<a href="#">MariaDB 5.5</a>
authentication_string	text	NO		NULL	Authentication string for the authentication plugin.	<a href="#">MariaDB 5.5</a>
password_expired	enum('N','Y')	NO		N	MySQL-compatibility option, not implemented in MariaDB.	
is_role	enum('N','Y')	NO		N	Whether the user is a <a href="#">role</a> .	<a href="#">MariaDB 10.0.5</a>
default_role	char(80)	NO		N	Role which will be enabled on user login automatically.	<a href="#">MariaDB 10.1.1</a>
max_statement_time	decimal(12,6)	NO		0.000000	If non-zero, how long queries can run before being killed automatically.	<a href="#">MariaDB 10.1.1</a>

The [Acl\\_roles](#) status variable, added in [MariaDB 10.1.4](#), indicates how many rows the

```
mysql.user
table contains where
is_role='Y'
```

The [Acl\\_users](#) status variable, added in [MariaDB 10.1.4](#), indicates how many rows the

```
mysql.user
```

table contains where

is\_role='N'

.

## Authentication Plugin

When the

plugin

column is empty, MariaDB defaults to authenticating accounts with either the

`mysql_native_password`

or the

`mysql_old_password`

plugins. It decides which based on the hash used in the value for the

Password

column. When there's no password set or when the 4.1 password hash is used, (which is 41 characters long), MariaDB uses the

`mysql_native_password`

plugin. The

`mysql_old_password`

plugin is used with pre-4.1 password hashes, (which are 16 characters long).

MariaDB also supports the use of alternative [authentication plugins](#). When the

plugin

column is not empty for the given account, MariaDB uses it to authenticate connection attempts. The specific plugin then uses the value of either the

Password

column or the

`authentication_string`

column to authenticate the user.

A specific authentication plugin can be used for an account by providing the

`IDENTIFIED VIA authentication_plugin`

clause with the

`CREATE USER`

,

`ALTER USER`

, or

`GRANT`

statements.

For example, the following statement would create an account that authenticates with the [PAM authentication plugin](#):

```
CREATE USER foo2@test IDENTIFIED VIA pam;
```

If the specific authentication plugin uses the

`authentication_string`

column, then this value for the account can be specified after a

`USING`

or

`AS`

keyword. For example, the [PAM authentication plugin](#) accepts a `service name` that would go into the

`authentication_string`

column for the account:

```
CREATE USER foo2@test IDENTIFIED VIA pam USING 'mariadb';
```

## 1.1.2.9.3.32 Spider mysql Database Tables

### 1.1.2.9.3.32.1 mysql.spider\_link\_failed\_log Table

The

`mysql.spider_link_failed_log`  
table is installed by the [Spider storage engine](#).

MariaDB starting with [10.4](#)

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO			
table_name	char(199)	NO			
link_id	char(64)	NO			
failed_time	timestamp	NO		current_timestamp()	

### 1.1.2.9.3.32.2 mysql.spider\_link\_mon\_servers Table

The

`mysql.spider_link_mon_servers`  
table is installed by the [Spider storage engine](#).

MariaDB starting with [10.4](#)

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
link_id	char(64)	NO	PRI		
sid	int(10) unsigned	NO	PRI	0	
server	char(64)	YES		NULL	
scheme	char(64)	YES		NULL	
host	char(64)	YES		NULL	
port	char(5)	YES		NULL	
socket	text	YES		NULL	
username	char(64)	YES		NULL	
password	char(64)	YES		NULL	
ssl_ca	text	YES		NULL	
ssl_capath	text	YES		NULL	
ssl_cert	text	YES		NULL	
ssl_cipher	char(64)	YES		NULL	
ssl_key	text	YES		NULL	

ssl_verify_server_cert	tinyint(4)	NO	0	
default_file	text	YES	NULL	
default_group	char(64)	YES	NULL	
dsn	char(64)	YES	NULL	
filedsn	text	YES	NULL	
driver	char(64)	YES	NULL	

## 1.1.2.9.3.32.3 mysql.spider\_tables Table

The

`mysql.spider_tables`  
table is installed by the [Spider storage engine](#).

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM](#) storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
link_id	int(11)	NO	PRI	0	
priority	bigint(20)	NO	MUL	0	
server	char(64)	YES		NULL	
schema	char(64)	YES		NULL	
host	char(64)	YES		NULL	
port	char(5)	YES		NULL	
socket	text	YES		NULL	
username	char(64)	YES		NULL	
password	char(64)	YES		NULL	
ssl_ca	text	YES		NULL	
ssl_capath	text	YES		NULL	
ssl_cert	text	YES		NULL	
ssl_cipher	char(64)	YES		NULL	
ssl_key	text	YES		NULL	
ssl_verify_server_cert	tinyint(4)	NO		0	
monitoring_binlog_pos_at_failing	tinyint(4)	NO		0	
default_file	text	YES		NULL	
default_group	char(64)	YES		NULL	
dsn	char(64)	YES		NULL	
filedsn	text	YES		NULL	
driver	char(64)	YES		NULL	
tgt_db_name	char(64)	YES		NULL	
tgt_table_name	char(64)	YES		NULL	
link_status	tinyint(4)	NO		1	
block_status	tinyint(4)	NO		0	

static_link_id	char(64)	YES	NULL	
----------------	----------	-----	------	--

## 1.1.2.9.3.32.4 mysql.spider\_table\_crd Table

The

`mysql.spider_table_crd`  
table is installed by the [Spider storage engine](#).

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
key_seq	int(10) unsigned	NO	PRI	0	
cardinality	bigint(20)	NO		0	

## 1.1.2.9.3.32.5 mysql.spider\_table\_position\_for\_recovery Table

The

`mysql.spider_table_position_for_recovery`  
table is installed by the [Spider storage engine](#).

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
failed_link_id	int(11)	NO	PRI	0	
source_link_id	int(11)	NO	PRI	0	
file	text	YES		NULL	
position	text	YES		NULL	
gtid	text	YES		NULL	

## 1.1.2.9.3.32.6 mysql.spider\_table\_sts Table

The

`mysql.spider_table_sts`  
table is installed by the [Spider storage engine](#).

MariaDB starting with 10.4

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until 10.3

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
db_name	char(64)	NO	PRI		
table_name	char(199)	NO	PRI		
data_file_length	bigint(20) unsigned	NO		0	
max_data_file_length	bigint(20) unsigned	NO		0	
index_file_length	bigint(20) unsigned	NO		0	
records	bigint(20) unsigned	NO		0	
mean_rec_length	bigint(20) unsigned	NO		0	
check_time	datetime	NO		0000-00-00 00:00:00	
create_time	datetime	NO		0000-00-00 00:00:00	
update_time	datetime	NO		0000-00-00 00:00:00	
checksum	bigint(20) unsigned	YES		NULL	

## 1.1.2.9.3.32.7 mysql.spider\_xa Table

The

`mysql.spider_xa`  
table is installed by the [Spider storage engine](#).

MariaDB starting with [10.4](#)

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
format_id	int(11)	NO	PRI	0	
gtrid_length	int(11)	NO	PRI	0	
bqual_length	int(11)	NO		0	
data	binary(128)	NO	PRI		
status	char(8)	NO	MUL		

## 1.1.2.9.3.32.8 mysql.spider\_xa\_failed\_log Table

The

`mysql.spider_xa_failed_log`  
table is installed by the [Spider storage engine](#).

MariaDB starting with [10.4](#)

In MariaDB 10.4 and later, this table uses the [Aria](#) storage engine.

MariaDB until [10.3](#)

In MariaDB 10.3 and before, this table uses the [MyISAM](#) storage engine.

It contains the following fields:

Field	Type	Null	Key	Default	Description
format_id	int(11)	NO		0	
gtrid_length	int(11)	NO		0	

bqual_length	int(11)	NO		0	
data	binary(128)	NO	MUL		
scheme	char(64)	NO			
host	char(64)	NO			
port	char(5)	NO			
socket	text	NO		NULL	
username	char(64)	NO			
password	char(64)	NO			
ssl_ca	text	YES		NULL	
ssl_capath	text	YES		NULL	
ssl_cert	text	YES		NULL	
ssl_cipher	char(64)	YES		NULL	
ssl_key	text	YES		NULL	
ssl_verify_server_cert	tinyint(4)	NO		0	
default_file	text	YES		NULL	
default_group	char(64)	YES		NULL	
dsn	char(64)	YES		NULL	
filedsn	text	YES		NULL	
driver	char(64)	YES		NULL	
thread_id	int(11)	YES		NULL	
status	char(8)	NO			
failed_time	timestamp	NO		current_timestamp()	

## 1.1.2.9.3.32.9 mysql.spider\_xa\_member Table

The

`mysql.spider_xa_member`  
table is installed by the [Spider storage engine](#).

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, this table uses the [Aria storage engine](#).

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, this table uses the [MyISAM storage engine](#).

It contains the following fields:

Field	Type	Null	Key	Default	Description
format_id	int(11)	NO		0	
gtrid_length	int(11)	NO		0	
bqual_length	int(11)	NO		0	
data	binary(128)	NO	MUL		
scheme	char(64)	NO			
host	char(64)	NO			
port	char(5)	NO			
socket	text	NO		NULL	
username	char(64)	NO			
password	char(64)	NO			
ssl_ca	text	YES		NULL	

ssl_capath	text	YES		NULL	
ssl_cert	text	YES		NULL	
ssl_cipher	char(64)	YES		NULL	
ssl_key	text	YES		NULL	
ssl_verify_server_cert	tinyint(4)	NO		0	
default_file	text	YES		NULL	
default_group	char(64)	YES		NULL	
dsn	char(64)	YES		NULL	
filedsn	text	YES		NULL	
driver	char(64)	YES		NULL	

## 1.1.2.9.4 Sys Schema

### 1.1.2.9.4.1 Sys Schema sys\_config Table

MariaDB starting with 10.6.0

The sys schema `sys_config` table was added in [MariaDB 10.6.0](#).

The `sys_config` table holds configuration options for the [sys schema](#).

This is a persistent table (using the `InnoDB` storage engine), with the configuration persisting across upgrades (new options are added with [INSERT IGNORE](#)).

The table also has two related triggers, which maintain the user that INSERTs or UPDATEs the configuration - `sys_config_insert_set_user` and `sys_config_update_set_user` respectively.

Its structure is as follows:

Field	Type	Null	Key	Default	Extra
variable	varchar(128)	NO	PRI	NULL	
value	varchar(128)	YES		NULL	
set_time	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
set_by	varchar(128)	YES		NULL	

Note, when functions check for configuration options, they first check whether a similar named user variable exists with a value, and if this is not set then pull the configuration option from this table in to that named user variable. This is done for performance reasons (to not continually SELECT from the table), however this comes with the side effect that once init'd, the values last with the session, somewhat like how session variables are init'd from global variables. If the values within this table are changed, they will not take effect until the user logs in again.

## Options Included

Variable	Default Value	Description
statement_truncate_len	64	Sets the size to truncate statements to, for the <code>format_statement()</code> function.
statement_performance_analyzer.limit	100	The maximum number of rows to include for the views that does not have a built-in limit (e.g. the 95th percentile view). If not set the limit is 100.
statement_performance_analyzer.view	NULL	Used together with the 'custom' view. If the value contains a space, it is considered a query, otherwise it must be an existing view querying the <code>performance_schema.events_statements_summary_by_digest</code> table.
diagnostics.allow_i_s_tables	OFF	Specifies whether it is allowed to do table scan queries on <code>information_schema.TABLES</code> for the diagnostics procedure.
diagnostics.include_raw	OFF	Set to 'ON' to include the raw data (e.g. the original output of "SELECT * FROM sys.metrics") for the diagnostics procedure.
ps_thread_trx_info.max_length	65535	Sets the maximum output length for JSON object output by the <code>ps_thread_trx_info()</code> function.

## 1.1.2.9.5 mariadb\_schema

## Contents

### 1. History

mariadb\_schema

is a data type qualifier that allows one to create MariaDB native date types in an [SQL\\_MODE](#) that has conflicting data type translations.

mariadb\_schema

was introduced in [MariaDB 10.3.24](#), [MariaDB 10.4.14](#) and [MariaDB 10.5.5](#).

For example, in [SQL\\_MODE=ORACLE](#), if one creates a table with the [DATE](#) type, it will actually create a [DATETIME](#) column to match what an Oracle user is expecting. To be able to create a MariaDB DATE in Oracle mode one would have to use

mariadb\_schema

:

```
CREATE TABLE t1 (d mariadb_schema.DATE);
```

mariadb\_schema

is also shown if one creates a table with

DATE

in MariaDB native mode and then does a [SHOW CREATE TABLE](#) in

ORACLE

mode:

```
SET sql_mode=DEFAULT;
CREATE OR REPLACE TABLE t1 (
  d DATE
);
SET SQL_mode=ORACLE;
SHOW CREATE TABLE t1;
+-----+
| Table | Create Table
+-----+
| t1    | CREATE TABLE "t1" (
  "d" mariadb_schema.date DEFAULT NULL
) |
+-----+
```

When the server sees the

mariadb\_schema

qualifier, it disables sql\_mode-specific data type translation and interprets the data type literally, so for example

mariadb\_schema.DATE

is interpreted as the traditional MariaDB

DATE

data type, no matter what the current sql\_mode is.

The

mariadb\_schema

prefix is displayed only when the data type name would be ambiguous otherwise. The prefix is displayed together with MariaDB

DATE

when [SHOW CREATE TABLE](#) is executed in [SQL\\_MODE=ORACLE](#). The prefix is not displayed when [SHOW CREATE TABLE](#) is executed in [SQL\\_MODE=DEFAULT](#), or when a non-ambiguous data type is displayed.

Note, the

mariadb\_schema

prefix can be used with any data type, including non-ambiguous ones:

```
CREATE OR REPLACE TABLE t1 (a mariadb_schema.INT);
SHOW CREATE TABLE t1;
+-----+
| Table | Create Table
+-----+
| t1    | CREATE TABLE "t1" (
  "a" int(11) DEFAULT NULL
) |
+-----+
```

Currently the

mariadb\_schema

prefix is only used in the following case:

- For a MariaDB native DATE type when running SHOW CREATE TABLE in Oracle mode .

## History

When running with `SQL_MODE=ORACLE` , MariaDB server translates the data type

```
DATE
to
DATETIME
, for better Oracle compatibility:
```

```
SET SQL_mode=ORACLE;
CREATE OR REPLACE TABLE t1 (
  d DATE
);
SHOW CREATE TABLE t1;
+-----+
| Table | Create Table |
+-----+
| t1    | CREATE TABLE "t1" (
  "d" datetime DEFAULT NULL
) |
```

Notice,

```
DATE
was translated to
DATETIME
.
```

This translation may cause some ambiguity. Suppose a user creates a table with a column of the traditional MariaDB

```
DATE
data type using the default sql_mode, but then switches to SQL_MODE=ORACLE and runs a SHOW CREATE TABLE statement:
```

```
SET sql_mode=DEFAULT;
CREATE OR REPLACE TABLE t1 (
  d DATE
);
SET SQL_mode=ORACLE;
SHOW CREATE TABLE t1;
```

Before

```
mariadb_schema
was introduced, the above script displayed:
```

```
CREATE TABLE "t1" (
  "d" date DEFAULT NULL
);
```

which had two problems:

- It was confusing for the reader: its not clear if it is the traditional MariaDB  
DATE  
, or is it Oracle-alike date (which is actually  
DATETIME  
);
- It broke replication and caused data type mismatch on the master and on the slave (see [MDEV-19632](#) ).

To address this problem, starting from the mentioned versions, MariaDB uses the idea of qualified data types:

```
SET sql_mode=DEFAULT;
CREATE OR REPLACE TABLE t1 (
  d DATE
);
SET SQL_mode=ORACLE;
SHOW CREATE TABLE t1;
+-----+
| Table | Create Table |
+-----+
| t1    | CREATE TABLE "t1" (
  "d" mariadb_schema.date DEFAULT NULL
) |
```

## 1.1.2.9.6 Writing Logs Into Tables

By default, all logs are disabled or written into files. The [general query log](#) and the [slow query log](#) can also be written to special tables in the `mysql` database. During the startup, entries will always be written into files.

Note that [EXPLAIN output](#) will only be recorded if the slow query log is written to a file and not to a table.

To write logs into tables, the [log\\_output](#) server system variable is used. Allowed values are

```
FILE  
,  
TABLE  
and  
NONE  
. It is possible to specify multiple values, separated with commas, to write the logs into both tables and files.  
NONE  
disables logging and has precedence over the other values.
```

So, to write logs into tables, one of the following settings can be used:

```
SET GLOBAL log_output = 'TABLE';  
SET GLOBAL log_output = 'FILE, TABLE';
```

The general log will be written into the [general\\_log](#) table, and the slow query log will be written into the [slow\\_log](#) table. Only a limited set of operations are supported for those special tables. For example, direct DML statements (like

```
INSERT  
) on those tables will fail with an error similar to the following:
```

```
ERROR 1556 (HY000): You can't use locks with log tables.
```

To flush data to the tables, use [FLUSH TABLES](#) instead of [FLUSH LOGS](#).

To empty the contents of the log tables, [TRUNCATE TABLE](#) can be used.

The log tables use the [CSV](#) storage engine by default. This allows an external program to read the files if needed: normal CSV files are stored in the

```
mysql  
subdirectory, in the data dir. However that engine is slow because it does not support indexes, so you can convert the tables to MyISAM (but not other storage engines). To do so, first temporarily disable logging:
```

```
SET GLOBAL general_log = 'OFF';  
ALTER TABLE mysql.general_log ENGINE = MyISAM;  
ALTER TABLE mysql.slow_log ENGINE = MyISAM;  
SET GLOBAL general_log = @old_log_state;
```

[CHECK TABLE](#) and [CHECKSUM TABLE](#) are supported.

[CREATE TABLE](#) is supported. [ALTER TABLE](#), [RENAME TABLE](#) and [DROP TABLE](#) are supported when logging is disabled, but log tables cannot be partitioned.

The contents of the log tables is not logged in the [binary log](#) thus cannot be replicated.

## 1.1.2.10 BINLOG

### Syntax

```
BINLOG 'str'
```

### Description

`BINLOG` is an internal-use statement. It is generated by the [mariadb-binlog/mysqlbinlog](#) program as the printable representation of certain events in [binary log](#) files. The `'str'` value is a base 64-encoded string that the server decodes to determine the data change indicated by the corresponding event. This statement requires the [SUPER](#) privilege (<= MariaDB 10.5.1) or the [BINLOG REPLAY](#) privilege (>= MariaDB 10.5.2).

### See also

- [MariaDB replication](#)

## 1.1.2.11 PURGE BINARY LOGS

### Syntax

```
PURGE { BINARY | MASTER } LOGS  
      { TO 'log_name' | BEFORE datetime_expr }
```

### Description

The

PURGE BINARY LOGS

statement deletes all the [binary log](#) files listed in the log index file prior to the specified log file name or date.

BINARY

and

MASTER

are synonyms. Deleted log files also are removed from the list recorded in the index file, so that the given log file becomes the first in the list.

The datetime expression is in the format 'YYYY-MM-DD hh:mm:ss'.

If a replica is active but has yet to read from a binary log file you attempt to delete, the statement will fail with an error. However, if the replica is not connected and has yet to read from a log file you delete, the file will be deleted, but the replica will be unable to continue replicating once it connects again.

This statement has no effect if the server was not started with the [--log-bin](#) option to enable binary logging.

To list the binary log files on the server, use [SHOW BINARY LOGS](#). To see which files they are reading, use [SHOW SLAVE STATUS](#) (or [SHOW REPLICAS STATUS](#) from [MariaDB 10.5.1](#)). You can only delete the files that are older than the oldest file that is used by the slaves.

To delete all binary log files, use [RESET MASTER](#). To move to a new log file (for example if you want to remove the current log file), use [FLUSH LOGS](#) before you execute

```
PURGE LOGS
```

If the [expire\\_logs\\_days](#) server system variable is not set to 0, the server automatically deletes binary log files after the given number of days. From [MariaDB 10.6](#), the [binlog\\_expire\\_logs\\_seconds](#) variable allows more precise control over binlog deletion, and takes precedence if both are non-zero.

Requires the [SUPER](#) privilege or, from [MariaDB 10.5.2](#), the [BINLOG ADMIN](#) privilege, to run.

### Examples

```
PURGE BINARY LOGS TO 'mariadb-bin.000063';
```

```
PURGE BINARY LOGS BEFORE '2013-04-21';
```

```
PURGE BINARY LOGS BEFORE '2013-04-22 09:55:22';
```

### See Also

- [Using and Maintaining the Binary Log](#)
- [FLUSH LOGS](#).

## 1.1.2.12 CACHE INDEX

### Syntax

```
CACHE INDEX  
tbl_index_list [, tbl_index_list] ...  
IN key_cache_name  
  
tbl_index_list:  
tbl_name [[INDEX|KEY] (index_name[, index_name] ...)]
```

# Description

The

```
CACHE INDEX  
statement assigns table indexes to a specific key cache. It is used only for MyISAM tables.
```

A default key cache exists and cannot be destroyed. To create more key caches, the [key\\_buffer\\_size](#) server system variable.

The associations between tables indexes and key caches are lost on server restart. To recreate them automatically, it is necessary to configure caches in a [configuration file](#) and include some

```
CACHE INDEX  
(and optionally
```

```
LOAD INDEX
```

) statements in the init file.

## Examples

The following statement assigns indexes from the tables t1, t2, and t3 to the key cache named hot\_cache:

```
CACHE INDEX t1, t2, t3 IN hot_cache;  
+-----+-----+-----+  
| Table | Op          | Msg_type | Msg_text |  
+-----+-----+-----+  
| test.t1 | assign_to_keycache | status    | OK        |  
| test.t2 | assign_to_keycache | status    | OK        |  
| test.t3 | assign_to_keycache | status    | OK        |  
+-----+-----+-----+
```

## Implementation (for MyISAM)

Normally CACHE INDEX should not take a long time to execute. Internally it's implemented the following way:

- Find the right key cache (under LOCK\_global\_system\_variables)
- Open the table with a TL\_READ\_NO\_INSERT lock.
- Flush the original key cache for the given file (under key cache lock)
- Flush the new key cache for the given file (safety)
- Move the file to the new key cache (under file share lock)

The only possible long operations are getting the locks for the table and flushing the original key cache, if there were many key blocks for the file in it.

We plan to also add CACHE INDEX for Aria tables if there is a need for this.

### 1.1.2.13 DESCRIBE

## Syntax

```
{DESCRIBE | DESC} tbl_name [col_name | wild]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

## Description

DESCRIBE provides information about the columns in a table. It is a shortcut for

```
SHOW COLUMNS FROM
```

. These statements also display information for [views](#).

col\_name can be a column name, or a string containing the SQL "%

" and "

— wildcard characters to obtain output only for the columns with names matching the string. There is no need to enclose the string within quotes unless it contains spaces or other special characters.

```
DESCRIBE city;
+-----+-----+-----+-----+
| Field | Type   | Null | Key  | Default | Extra      |
+-----+-----+-----+-----+
| Id    | int(11) | NO   | PRI   | NULL    | auto_increment |
| Name  | char(35) | YES  |       | NULL    |             |
| Country | char(3) | NO   | UNI   |          |             |
| District | char(20) | YES  | MUL   |          |             |
| Population | int(11) | YES  |       | NULL    |             |
+-----+-----+-----+-----+
```

The description for

```
SHOW COLUMNS
```

provides more information about the output columns.

## See Also

- [SHOW COLUMNS](#)
- [INFORMATION\\_SCHEMA.COLUMNS Table](#)
- [mysqlshow](#)

## 1.1.2.14 EXECUTE Statement

### Syntax

```
EXECUTE stmt_name
      [USING expression[, expression] ...]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [See Also](#)

MariaDB starting with 10.2.3

EXECUTE  
with expression as parameters was introduced in [MariaDB 10.2.3](#). Before that one could only use variables (@var\_name) as parameters.

## Description

After preparing a statement with

```
PREPARE
```

, you execute it with an  
EXECUTE  
statement that refers to the prepared statement name. If the prepared statement contains any parameter markers, you must supply a  
USING  
clause that lists user variables containing the values to be bound to the parameters. Parameter values can be supplied only by user variables,  
and the  
USING  
clause must name exactly as many variables as the number of parameter markers in the statement.

You can execute a given prepared statement multiple times, passing different variables to it or setting the variables to different values before each execution.

If the specified statement has not been PREPARED, an error similar to the following is produced:

ERROR 1243 (HY000): Unknown prepared statement handler (stmt\_name) given to EXECUTE

## Example

See [example in PREPARE](#).

## See Also

- [EXECUTE IMMEDIATE](#)

## 1.1.2.15 HELP Command

### Syntax

```
HELP search_string
```

### Description

The

HELP

command can be used in any MariaDB client, such as the [mysql](#) command-line client, to get basic syntax help and a short description for most commands and functions.

If you provide an argument to the

HELP

command, the [mysql](#) client uses it as a search string to access server-side help. The proper operation of this command requires that the help tables in the

mysql

database be initialized with help topic information.

If there is no match for the search string, the search fails. Use

HELP contents

to see a list of the help categories:

```
HELP contents
You asked for help about help category: "Contents"
For more information, type 'help <item>', where <item> is one of the following
categories:
  Account Management
  Administration
  Compound Statements
  Data Definition
  Data Manipulation
  Data Types
  Functions
  Functions and Modifiers for Use with GROUP BY
  Geographic Features
  Help Metadata
  Language Structure
  Plugins
  Procedures
  Sequences
  Table Maintenance
  Transactions
  User-Defined Functions
  Utility
```

If a search string matches multiple items, MariaDB shows a list of matching topics:

```
HELP drop
Many help items for your request exist.
To make a more specific request, please type 'help <item>',
where <item> is one of the following
topics:
ALTER TABLE
DROP DATABASE
DROP EVENT
DROP FUNCTION
DROP FUNCTION UDF
DROP INDEX
DROP PACKAGE
DROP PACKAGE BODY
DROP PROCEDURE
DROP ROLE
DROP SEQUENCE
DROP SERVER
DROP TABLE
DROP TRIGGER
DROP USER
DROP VIEW
```

Then you can enter a topic as the search string to see the help entry for that topic.

The help is provided with the MariaDB server and makes use of four help tables found in the

```
mysql
database: help\_relation , help\_topic , help\_category and help\_keyword . These tables are populated by the
mysql_install_db
or
fill\_help\_table.sql
scripts which, until MariaDB 10.4.7 , contain data generated from an old version of MySQL.
```

## 1.1.2.16 KILL [CONNECTION | QUERY]

### Syntax

```
KILL [HARD | SOFT] [CONNECTION | QUERY [ID] ] [thread_id | USER user_name | query_id]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

### Description

Each connection to mysqld runs in a separate thread. You can see which threads are running with the

```
SHOW PROCESSLIST
statement and kill a thread with the
KILL thread_id
statement.
KILL
allows the optional
CONNECTION
or
QUERY
modifier:
•
    KILL CONNECTION
    is the same as
    KILL
    with no modifier: It terminates the connection associated with the given thread or query id.
•
    KILL QUERY
    terminates the statement that the connection thread_id is currently executing, but leaves the connection itself intact.
•
    KILL QUERY ID
    (introduced in MariaDB 10.0.5 ) terminates the query by query_id, leaving the connection intact.
```

If a connection is terminated that has an active transaction, the transaction will be rolled back. If only a query is killed, the current transaction will stay

active. See also [idle\\_transaction\\_timeout](#).

If you have the [PROCESS](#) privilege, you can see all threads. If you have the [SUPER](#) privilege, or, from [MariaDB 10.5.2](#), the [CONNECTION ADMIN](#) privilege, you can kill all threads and statements. Otherwise, you can see and kill only your own threads and statements.

Killing queries that repair or create indexes on MyISAM and Aria tables may result in corrupted tables. Use the [SOFT](#) option to avoid this!

The

HARD  
option (default) kills a command as soon as possible. If you use SOFT, then critical operations that may leave a table in an inconsistent state will not be interrupted. Such operations include REPAIR and INDEX creation for MyISAM and Aria tables ([REPAIR TABLE](#), [OPTIMIZE TABLE](#)).

KILL ... USER username  
will kill all connections/queries for a given user.  
USER  
can be specified one of the following ways:

- username (Kill without regard to hostname)
- username@hostname
- 

[CURRENT\\_USER](#)

or

[CURRENT\\_USER\(\)](#)

If you specify a thread id and that thread does not exist, you get the following error:

ERROR 1094 (HY000): Unknown thread id: <thread\_id>

If you specify a query id that doesn't exist, you get the following error:

ERROR 1957 (HY000): Unknown query id: <query\_id>

However, if you specify a user name, no error is issued for non-connected (or even non-existing) users. To check if the connection/query has been killed, you can use the [ROW\\_COUNT\(\)](#) function.

A client whose connection is killed receives the following error:

ERROR 1317 (70100): Query execution was interrupted

To obtain a list of existing sessions, use the

[SHOW PROCESSLIST](#)

statement or query the [Information Schema](#)

[PROCESSLIST](#)

table.

**Note:** You cannot use

[KILL](#)

with the Embedded MySQL Server library because the embedded server merely runs inside the threads of the host application. It does not create any connection threads of its own.

**Note:** You can also use

[mysqladmin kill thread\\_id \[,thread\\_id...\]](#)

to kill connections. To get a list of running queries, use

[mysqladmin processlist](#)

. See [mysqladmin](#).

Percona Toolkit contains a program, [pt-kill](#) that can be used to automatically kill connections that match certain criteria. For example, it can be used to terminate idle connections, or connections that have been busy for more than 60 seconds.

## See Also

- [Query limits and timeouts](#)
- [Aborting statements that exceed a certain time to execute](#)
- [idle\\_transaction\\_timeout](#)

## 1.1.2.17 LOAD INDEX

### Syntax

```
LOAD INDEX INTO CACHE
tbl_index_list [, tbl_index_list] ...

tbl_index_list:
tbl_name
[[INDEX|KEY] (index_name[, index_name] ...)]
[IGNORE LEAVES]
```

### Description

The

```
LOAD INDEX INTO CACHE
statement preloads a table index into the key cache to which it has been assigned by an explicit
CACHE INDEX
statement, or into the default key cache otherwise.
LOAD INDEX INTO CACHE
is used only for MyISAM or Aria tables. Until MariaDB 5.3, it was not supported for tables having user-defined partitioning, but this limitation was
removed in MariaDB 5.5.
```

The

```
IGNORE LEAVES
modifier causes only blocks for the nonleaf nodes of the index to be preloaded.
```

## 1.1.2.18 RESET

### Syntax

```
RESET reset_option [, reset_option] ...
```

### Description

The

```
RESET
statement is used to clear the state of various server operations. You must have the
```

```
RELOAD privilege
```

```
to execute
```

```
RESET
```

```
.
```

```
RESET
acts as a stronger version of the FLUSH statement.
```

The different

```
RESET
```

```
options are:
```

Option	Description
SLAVE ["connection_name"] [ALL]	Deletes all <a href="#">relay logs</a> from the slave and reset the replication position in the master <a href="#">binary log</a> .

MASTER	Deletes all old binary logs, makes the binary index file ( <a href="#">--log-bin-index</a> ) empty and creates a new binary log file. This is useful when you want to reset the master to an initial state. If you want to just delete old, not used binary logs, you should use the <a href="#">PURGE BINARY LOGS</a> command.
QUERY CACHE	Removes all queries from <a href="#">the query cache</a> . See also <a href="#">FLUSH QUERY CACHE</a> .

## 1.1.2.19 SHUTDOWN

### Contents

1. [Syntax](#)
2. [Description](#)
3. [WAIT FOR ALL SLAVES](#)
4. [Required Permissions](#)
5. [Shutdown for Upgrades](#)
6. [Example](#)
7. [Other Ways to Stop mysqld](#)
8. [See Also](#)

### Syntax

```
SHUTDOWN [WAIT FOR ALL { SLAVES | REPLICAS } ]
```

### Description

The

`SHUTDOWN`

command shuts the server down.

### WAIT FOR ALL SLAVES

MariaDB starting with [10.4.4](#)

The

`WAIT FOR ALL SLAVES`

option was first added in [MariaDB 10.4.4](#).

`WAIT FOR ALL REPLICAS`

has been a synonym since [MariaDB 10.5.1](#).

When a master server is shutdown and it goes through the normal shutdown process, the master kills client threads in random order. By default, the master also considers its binary log dump threads to be regular client threads. As a consequence, the binary log dump threads can be killed while client threads still exist, and this means that data can be written on the master during a normal shutdown that won't be replicated. This is true even if [semi-synchronous replication](#) is being used.

In [MariaDB 10.4](#) and later, this problem can be solved by shutting down the server with the `SHUTDOWN` command and by providing the

`WAIT FOR ALL SLAVES`

option to the command. For example:

```
SHUTDOWN WAIT FOR ALL SLAVES;
```

When the

`WAIT FOR ALL SLAVES`

option is provided, the server only kills its binary log dump threads after all client threads have been killed, and it only completes the shutdown after the last `binary log` has been sent to all connected replicas.

See [Replication Threads: Binary Log Dump Threads and the Shutdown Process](#) for more information.

### Required Permissions

One must have a

`SHUTDOWN`

privilege (see [GRANT](#)) to use this command. It is the same privilege one needs to use the `mariadb-admin/mysqladmin shutdown` command.

### Shutdown for Upgrades

If you are doing a shutdown to [migrate to another major version of MariaDB](#), please ensure that the `innodb_fast_shutdown` variable is not 2 (fast crash shutdown). The default of this variable is 1.

## Example

The following example shows how to create an [event](#) which turns off the server at a certain time:

```
CREATE EVENT `test`.`shutd`
  ON SCHEDULE
    EVERY 1 DAY
    STARTS '2014-01-01 20:00:00'
    COMMENT 'Shutdown Maria when the office is closed'
DO BEGIN
  SHUTDOWN;
END;
```

## Other Ways to Stop mysqld

You can use the [mariadb-admin/mysqladmin shutdown](#) command to take down mysqld cleanly.

You can also use the system kill command on Unix with signal SIGTERM (15)

```
kill -SIGTERM pid-of-mysqld-process
```

You can find the process number of the server process in the file that ends with

.pid  
in your data directory.

The above is identical to

```
mysqladmin shutdown
```

On windows you should use:

```
NET STOP MySQL
```

## See Also

- [mariadb-admin/mysqladmin shutdown](#) .
- [InnoDB fast shutdown option](#)

## 1.1.2.20 USE

### Syntax

```
USE db_name
```

### Description

The

'USE db\_name'  
statement tells MariaDB to use the  
db\_name  
database as the default (current) database for subsequent statements. The database remains the default until the end of the session or another  
USE  
statement is issued:

```
USE db1;
SELECT COUNT(*) FROM mytable;  # selects from db1.mytable
USE db2;
SELECT COUNT(*) FROM mytable;  # selects from db2.mytable
```

The [DATABASE\(\)](#) function ( [SCHEMA\(\)](#) is a synonym) returns the default database.

Another way to set the default database is specifying its name at [mysql](#) command line client startup.

## See Also

- [Identifier Qualifiers](#)

## 1.1.3 Data Definition

### 1.1.3.1 CREATE

#### 1.1.3.1.1 CREATE DATABASE

##### Syntax

```
CREATE [OR REPLACE] {DATABASE | SCHEMA} [IF NOT EXISTS] db_name  
[create_specification] ...  
  
create_specification:  
  [DEFAULT] CHARACTER SET [=] charset_name  
  | [DEFAULT] COLLATE [=] collation_name  
  | COMMENT [=] 'comment'
```

##### Contents

1. [Syntax](#)
2. [Description](#)
  1. [OR REPLACE](#)
  2. [IF NOT EXISTS](#)
  3. [COMMENT](#)
3. [Examples](#)
4. [See Also](#)

##### Description

CREATE DATABASE  
creates a database with the given name. To use this statement, you need the [CREATE privilege](#) for the database.  
CREATE SCHEMA  
is a synonym for  
CREATE DATABASE

For valid identifiers to use as database names, see [Identifier Names](#).

##### OR REPLACE

MariaDB starting with [10.1.3](#)

The

OR REPLACE  
clause was added in [MariaDB 10.1.3](#)

If the optional

OR REPLACE  
clause is used, it acts as a shortcut for:

```
DROP DATABASE IF EXISTS db_name;  
CREATE DATABASE db_name ...;
```

##### IF NOT EXISTS

When the

IF NOT EXISTS  
clause is used, MariaDB will return a warning instead of an error if the specified database already exists.

##### COMMENT

MariaDB starting with [10.5.0](#)

From [MariaDB 10.5.0](#), it is possible to add a comment of a maximum of 1024 bytes. If the comment length exceeds this length, an error/warning code 4144 is thrown. The database comment is also added to the db.opt file, as well as to the [information\\_schema.schemata](#) table.

## Examples

```
CREATE DATABASE db1;
Query OK, 1 row affected (0.18 sec)

CREATE DATABASE db1;
ERROR 1007 (HY000): Can't create database 'db1'; database exists

CREATE OR REPLACE DATABASE db1;
Query OK, 2 rows affected (0.00 sec)

CREATE DATABASE IF NOT EXISTS db1;
Query OK, 1 row affected, 1 warning (0.01 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1007 | Can't create database 'db1'; database exists |
+-----+-----+
```

Setting the [character sets and collation](#). See [Setting Character Sets and Collations](#) for more details.

```
CREATE DATABASE czech_slovak_names
  CHARACTER SET = 'keybcs2'
  COLLATE = 'keybcs2_bin';
```

Comments, from [MariaDB 10.5.0](#):

```
CREATE DATABASE presentations COMMENT 'Presentations for conferences';
```

## See Also

- [Identifier Names](#)
- [DROP DATABASE](#)
- [SHOW CREATE DATABASE](#)
- [ALTER DATABASE](#)
- [SHOW DATABASES](#)
- [Character Sets and Collations](#)
- [Information Schema SCHEMATA Table](#)

### 1.1.3.1.2 CREATE EVENT

## Syntax

```
CREATE [OR REPLACE]
  [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
  EVENT
  [IF NOT EXISTS]
  event_name
  ON SCHEDULE schedule
  [ON COMPLETION [NOT] PRESERVE]
  [ENABLE | DISABLE | DISABLE ON SLAVE]
  [COMMENT 'comment']
  DO sql_statement;

schedule:
  AT timestamp [+ INTERVAL interval] ...
  | EVERY interval
  [STARTS timestamp [+ INTERVAL interval] ...]
  [ENDS timestamp [+ INTERVAL interval] ...]

interval:
  quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
            WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
            DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [OR REPLACE](#)
  2. [IF NOT EXISTS](#)
  3. [ON SCHEDULE](#)
  4. [AT](#)
  5. [ON COMPLETION \[NOT\] PRESERVE](#)
  6. [ENABLE/DISABLE/DISABLE ON SLAVE](#)
  7. [COMMENT](#)
3. [Examples](#)
4. [See Also](#)

## Description

This statement creates and schedules a new [event](#). It requires the

`EVENT`

privilege for the schema in which the event is to be created.

The minimum requirements for a valid CREATE EVENT statement are as follows:

- The keywords

`CREATE EVENT`

plus an event name, which uniquely identifies the event in the current schema. (Prior to MySQL 5.1.12, the event name needed to be unique only among events created by the same user on a given database.)

- An

`ON SCHEDULE`

clause, which determines when and how often the event executes.

- A

`DO`

clause, which contains the SQL statement to be executed by an event.

Here is an example of a minimal

`CREATE EVENT`  
statement:

```
CREATE EVENT myevent
  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
  DO
    UPDATE myschema.mytable SET mycol = mycol + 1;
```

The previous statement creates an event named `myevent`. This event executes once — one hour following its creation — by running an SQL statement that increments the value of the `myschema.mytable` table's `mycol` column by 1.

The `event_name` must be a valid MariaDB identifier with a maximum length of 64 characters. It may be delimited using back ticks, and may be qualified with the name of a database schema. An event is associated with both a MariaDB user (the definer) and a schema, and its name must be unique among names of events within that schema. In general, the rules governing event names are the same as those for names of stored routines. See [Identifier Names](#).

If no schema is indicated as part of `event_name`, the default (current) schema is assumed.

For valid identifiers to use as event names, see [Identifier Names](#).

### OR REPLACE

The

`OR REPLACE`

clause was included in [MariaDB 10.1.4](#). If used and the event already exists, instead of an error being returned, the existing event will be dropped and replaced by the newly defined event.

### IF NOT EXISTS

If the

`IF NOT EXISTS`

clause is used, MariaDB will return a warning instead of an error if the event already exists. Cannot be used together with OR REPLACE.

### ON SCHEDULE

The

`ON SCHEDULE`

clause can be used to specify when the event must be triggered.

## AT

If you want to execute the event only once (one time event), you can use the

AT

keyword, followed by a timestamp. If you use

CURRENT\_TIMESTAMP

, the event acts as soon as it is created. As a convenience, you can add one or more intervals to that timestamp. You can also specify a timestamp in the past, so that the event is stored but not triggered, until you modify it via [ALTER EVENT](#).

The following example shows how to create an event that will be triggered tomorrow at a certain time:

```
CREATE EVENT example
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 DAY + INTERVAL 3 HOUR
DO something;
```

You can also specify that an event must be triggered at a regular interval (recurring event). In such cases, use the

EVERY

clause followed by the interval.

If an event is recurring, you can specify when the first execution must happen via the

STARTS

clause and a maximum time for the last execution via the

ENDS

clause.

STARTS

and

ENDS

clauses are followed by a timestamp and, optionally, one or more intervals. The

ENDS

clause can specify a timestamp in the past, so that the event is stored but not executed until you modify it via [ALTER EVENT](#).

In the following example, next month a recurring event will be triggered hourly for a week:

```
CREATE EVENT example
ON SCHEDULE EVERY 1 HOUR
STARTS CURRENT_TIMESTAMP + INTERVAL 1 MONTH
ENDS CURRENT_TIMESTAMP + INTERVAL 1 MONTH + INTERVAL 1 WEEK
DO some_task;
```

Intervals consist of a quantity and a time unit. The time units are the same used for other statements and time functions, except that you can't use microseconds for events. For simple time units, like

HOUR

or

MINUTE

, the quantity is an integer number, for example '10 MINUTE'. For composite time units, like

HOUR\_MINUTE

or

HOUR\_SECOND

, the quantity must be a string with all involved simple values and their separators, for example '2:30' or '2:30:30'.

## ON COMPLETION [NOT] PRESERVE

The

ON COMPLETION

clause can be used to specify if the event must be deleted after its last execution (that is, after its

AT

or

ENDS

timestamp is past). By default, events are dropped when they are expired. To explicitly state that this is the desired behaviour, you can use

ON COMPLETION NOT PRESERVE

. Instead, if you want the event to be preserved, you can use

ON COMPLETION PRESERVE

In you specify

ON COMPLETION NOT PRESERVE

, and you specify a timestamp in the past for

AT

```
or
ENDS
clause, the event will be immediately dropped. In such cases, you will get a Note 1558: "Event execution time is in the past and ON COMPLETION NOT PRESERVE is set. The event was dropped immediately after creation".
```

## ENABLE/DISABLE/DISABLE ON SLAVE

Events are

```
ENABLE
d by default. If you want to stop MariaDB from executing an event, you may specify
DISABLE
. When it is ready to be activated, you may enable it using
```

```
ALTER EVENT
```

```
. Another option is
DISABLE ON SLAVE
, which indicates that an event was created on a master and has been replicated to the slave, which is prevented from executing the event. If
DISABLE ON SLAVE
is specifically set, the event will not be executed.
```

## COMMENT

The

```
COMMENT
```

clause may be used to set a comment for the event. Maximum length for comments is 64 characters. The comment is a string, so it must be quoted. To see events comments, you can query the [INFORMATION\\_SCHEMA.EVENTS](#) table (the column is named `EVENT_COMMENT`).

## Examples

Minimal

```
CREATE EVENT
statement:
```

```
CREATE EVENT myevent
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
DO
    UPDATE myschema.mytable SET mycol = mycol + 1;
```

An event that will be triggered tomorrow at a certain time:

```
CREATE EVENT example
ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 DAY + INTERVAL 3 HOUR
DO something;
```

Next month a recurring event will be triggered hourly for a week:

```
CREATE EVENT example
ON SCHEDULE EVERY 1 HOUR
STARTS CURRENT_TIMESTAMP + INTERVAL 1 MONTH
ENDS CURRENT_TIMESTAMP + INTERVAL 1 MONTH + INTERVAL 1 WEEK
DO some_task;
```

OR REPLACE and IF NOT EXISTS:

```

CREATE EVENT myevent
  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
  DO
    UPDATE myschema.mytable SET mycol = mycol + 1;
ERROR 1537 (HY000): Event 'myevent' already exists

CREATE OR REPLACE EVENT myevent
  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
  DO
    UPDATE myschema.mytable SET mycol = mycol + 1;
Query OK, 0 rows affected (0.00 sec)

CREATE EVENT IF NOT EXISTS myevent
  ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 HOUR
  DO
    UPDATE myschema.mytable SET mycol = mycol + 1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1537 | Event 'myevent' already exists |
+-----+-----+

```

## See Also

- [Identifier Names](#)
- [Events Overview](#)
- [SHOW CREATE EVENT](#)
- [ALTER EVENT](#)
- [DROP EVENT](#)

### 1.1.3.1.3 CREATE FUNCTION

#### Syntax

```

CREATE [OR REPLACE]
  [DEFINER = {user | CURRENT_USER | role | CURRENT_ROLE}]
  [AGGREGATE] FUNCTION [IF NOT EXISTS] func_name ([func_parameter[,...]])
  RETURNS type
  [characteristic ...]
  RETURN func_body

func_parameter:
  [ IN | OUT | INOUT | IN OUT ] param_name type

type:
  Any valid MariaDB data type

characteristic:
  LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'

func_body:
  Valid SQL procedure statement

```

## Contents

- 1. [Syntax](#)
  - 2. [Description](#)
    - 1. [IN | OUT | INOUT | IN OUT](#)
    - 2. [AGGREGATE](#)
    - 3. [RETURNS](#)
    - 4. [LANGUAGE SQL](#)
    - 5. [OR REPLACE](#)
    - 6. [IF NOT EXISTS](#)
    - 7. [\[NOT\] DETERMINISTIC](#)
    - 8. [MODIFIES SQL DATA](#)
    - 9. [READS SQL DATA](#)
  - 10. [CONTAINS SQL](#)
  - 11. [NO SQL](#)
  - 12. [Oracle Mode](#)
- 3. [Security](#)
  - 4. [Character sets and collations](#)
  - 5. [Examples](#)
  - 6. [See Also](#)

## Description

Use the

```
CREATE FUNCTION
```

statement to create a new [stored function](#). You must have the [CREATE ROUTINE](#) database privilege to use

```
CREATE FUNCTION
```

. A function takes any number of arguments and returns a value from the function body. The function body can be any valid SQL expression as you would use, for example, in any select expression. If you have the appropriate privileges, you can call the function exactly as you would any built-in function. See [Security](#) below for details on privileges.

You can also use a variant of the

```
CREATE FUNCTION
```

statement to install a user-defined function (UDF) defined by a plugin. See [CREATE FUNCTION \(UDF\)](#) for details.

You can use a [SELECT](#) statement for the function body by enclosing it in parentheses, exactly as you would to use a subselect for any other expression.

The

```
SELECT
```

statement must return a single value. If more than one column is returned when the function is called, error 1241 results. If more than one row is returned when the function is called, error 1242 results. Use a

```
LIMIT
```

clause to ensure only one row is returned.

You can also replace the

```
RETURN
```

clause with a [BEGIN...END](#) compound statement. The compound statement must contain a

```
RETURN
```

statement. When the function is called, the

```
RETURN
```

statement immediately returns its result, and any statements after

```
RETURN
```

are effectively ignored.

By default, a function is associated with the current database. To associate the function explicitly with a given database, specify the fully-qualified name as

```
db_name
```

```
.
```

```
func_name
```

when you create it. If the function name is the same as the name of a built-in function, you must use the fully qualified name when you call it.

The parameter list enclosed within parentheses must always be present. If there are no parameters, an empty parameter list of () should be used. Parameter names are not case sensitive.

Each parameter can be declared to use any valid data type, except that the COLLATE attribute cannot be used.

For valid identifiers to use as function names, see [Identifier Names](#).

[IN | OUT | INOUT | IN OUT](#)

The function parameter qualifiers for

```
IN  
,  
OUT  
,  
INOUT  
, and  
IN OUT
```

were added in a 10.8.0 preview release. Prior to 10.8.0 quantifiers were supported only in procedures.

```
OUT  
,  
INOUT  
and its equivalent  
IN OUT  
, are only valid if called from  
SET  
and not  
SELECT
```

. These quantifiers are especially useful for creating functions with more than one return value. This allows functions to be more complex and nested.

```
DELIMITER $$  
CREATE FUNCTION add_func3(IN a INT, IN b INT, OUT c INT) RETURNS INT  
BEGIN  
    SET c = 100;  
    RETURN a + b;  
END;  
$$  
DELIMITER ;  
  
SET @a = 2;  
SET @b = 3;  
SET @c = 0;  
SET @res= add_func3(@a, @b, @c);  
  
SELECT add_func3(@a, @b, @c);  
ERROR 4186 (HY000): OUT or INOUT argument 3 for function add_func3 is not allowed here  
  
DELIMITER $$  
CREATE FUNCTION add_func4(IN a INT, IN b INT, d INT) RETURNS INT  
BEGIN  
    DECLARE c, res INT;  
    SET res = add_func3(a, b, c) + d;  
    if (c > 99) then  
        return 3;  
    else  
        return res;  
    end if;  
END;  
$$  
  
DELIMITER ;  
  
SELECT add_func4(1,2,3);  
+-----+  
| add_func4(1,2,3) |  
+-----+  
|            3 |  
+-----+
```

## AGGREGATE

MariaDB starting with 10.3.3

From [MariaDB 10.3.3](#), it is possible to create stored aggregate functions as well. See [Stored Aggregate Functions](#) for details.

## RETURNS

The

```
RETURNS
```

clause specifies the return type of the function.

NULL

values are permitted with all return types.

What happens if the

RETURN

clause returns a value of a different type? It depends on the `SQL_MODE` in effect at the moment of the function creation.

If the `SQL_MODE` is strict (`STRICT_ALL_TABLES` or `STRICT_TRANS_TABLES` flags are specified), a 1366 error will be produced.

Otherwise, the value is coerced to the proper type. For example, if a function specifies an

ENUM

or

SET

value in the

RETURNS

clause, but the

RETURN

clause returns an integer, the value returned from the function is the string for the corresponding

ENUM

member of set of

SET

members.

MariaDB stores the `SQL_MODE` system variable setting that is in effect at the time a routine is created, and always executes the routine with this setting in force, regardless of the server SQL mode in effect when the routine is invoked.

## LANGUAGE SQL

LANGUAGE SQL

is a standard SQL clause, and it can be used in MariaDB for portability. However that clause has no meaning, because SQL is the only supported language for stored functions.

A function is deterministic if it can produce only one result for a given list of parameters. If the result may be affected by stored data, server variables, random numbers or any value that is not explicitly passed, then the function is not deterministic. Also, a function is non-deterministic if it uses non-deterministic functions like `NOW()` or `CURRENT_TIMESTAMP()`. The optimizer may choose a faster execution plan if it known that the function is deterministic. In such cases, you should declare the routine using the

DETERMINISTIC

keyword. If you want to explicitly state that the function is not deterministic (which is the default) you can use the

NOT DETERMINISTIC

keywords.

If you declare a non-deterministic function as

DETERMINISTIC

, you may get incorrect results. If you declare a deterministic function as

NOT DETERMINISTIC

, in some cases the queries will be slower.

## OR REPLACE

MariaDB starting with 10.1.3

If the optional

OR REPLACE

clause is used, it acts as a shortcut for:

```
DROP FUNCTION IF EXISTS function_name;  
CREATE FUNCTION function_name ...;
```

with the exception that any existing privileges for the function are not dropped.

## IF NOT EXISTS

MariaDB starting with 10.1.3

If the IF NOT EXISTS clause is used, MariaDB will return a warning instead of an error if the function already exists. Cannot be used together with OR REPLACE.

## [NOT] DETERMINISTIC

The

[NOT] DETERMINISTIC  
clause also affects [binary logging](#), because the  
STATEMENT  
format can not be used to store or replicate non-deterministic statements.

CONTAINS SQL  
,  
NO SQL  
,  
READS SQL DATA  
, and  
MODIFIES SQL DATA  
are informative clauses that tell the server what the function does. MariaDB does not check in any way whether the specified clause is correct. If none of these clauses are specified,  
CONTAINS SQL  
is used by default.

## MODIFIES SQL DATA

MODIFIES SQL DATA  
means that the function contains statements that may modify data stored in databases. This happens if the function contains statements like [DELETE](#), [UPDATE](#), [INSERT](#), [REPLACE](#) or DDL.

## READS SQL DATA

READS SQL DATA  
means that the function reads data stored in databases, but does not modify any data. This happens if [SELECT](#) statements are used, but there no write operations are executed.

## CONTAINS SQL

CONTAINS SQL  
means that the function contains at least one SQL statement, but it does not read or write any data stored in a database. Examples include [SET](#) or [DO](#).

## NO SQL

NO SQL  
means nothing, because MariaDB does not currently support any language other than SQL.

## Oracle Mode

MariaDB starting with 10.3

From [MariaDB 10.3](#), a subset of Oracle's PL/SQL language has been supported in addition to the traditional SQL/PSM-based MariaDB syntax. See [Oracle mode from MariaDB 10.3](#) for details on changes when running Oracle mode.

# Security

You must have the [EXECUTE](#) privilege on a function to call it. MariaDB automatically grants the

EXECUTE  
and  
ALTER ROUTINE  
privileges to the account that called  
CREATE FUNCTION  
, even if the  
DEFINER  
clause was used.

Each function has an account associated as the definer. By default, the definer is the account that created the function. Use the  
DEFINER  
clause to specify a different account as the definer. You must have the [SUPER](#) privilege, or, from [MariaDB 10.5.2](#), the [SET USER](#) privilege, to  
use the  
DEFINER  
clause. See [Account Names](#) for details on specifying accounts.

The

```
SQL SECURITY
clause specifies what privileges are used when a function is called. If
SQL SECURITY
is
INVOKER
, the function body will be evaluated using the privileges of the user calling the function. If
SQL SECURITY
is
DEFINER
, the function body is always evaluated using the privileges of the definer account.
DEFINER
is the default.
```

This allows you to create functions that grant limited access to certain data. For example, say you have a table that stores some employee information, and that you've granted

```
SELECT
privileges only on certain columns to the user account
roger
.
```

```
CREATE TABLE employees (name TINYTEXT, dept TINYTEXT, salary INT);
GRANT SELECT (name, dept) ON employees TO roger;
```

To allow the user to get the maximum salary for a department, define a function and grant the

```
EXECUTE
privilege:
```

```
CREATE FUNCTION max_salary (dept TINYTEXT) RETURNS INT RETURN
(SELECT MAX(salary) FROM employees WHERE employees.dept = dept);
GRANT EXECUTE ON FUNCTION max_salary TO roger;
```

Since

```
SQL SECURITY
defaults to
DEFINER
, whenever the user
roger
```

calls this function, the subselect will execute with your privileges. As long as you have privileges to select the salary of each employee, the caller of the function will be able to get the maximum salary for each department without being able to see individual salaries.

## Character sets and collations

Function return types can be declared to use any valid [character set and collation](#). If used, the COLLATE attribute needs to be preceded by a CHARACTER SET attribute.

If the character set and collation are not specifically set in the statement, the database defaults at the time of creation will be used. If the database defaults change at a later stage, the stored function character set/collation will not be changed at the same time; the stored function needs to be dropped and recreated to ensure the same character set/collation as the database is used.

## Examples

The following example function takes a parameter, performs an operation using an SQL function, and returns the result.

```
CREATE FUNCTION hello (s CHAR(20))
RETURNS CHAR(50) DETERMINISTIC
RETURN CONCAT('Hello, ',s,'!');

SELECT hello('world');
+-----+
| hello('world') |
+-----+
| Hello, world! |
+-----+
```

You can use a compound statement in a function to manipulate data with statements like

```
INSERT
and
UPDATE
.
The following example creates a counter function that uses a temporary table to store the current value. Because the compound statement
```

contains statements terminated with semicolons, you have to first change the statement delimiter with the `DELIMITER` statement to allow the semicolon to be used in the function body. See [Delimiters in the mysql client](#) for more.

```
CREATE TEMPORARY TABLE counter (c INT);
INSERT INTO counter VALUES (0);
DELIMITER //
CREATE FUNCTION counter () RETURNS INT
BEGIN
    UPDATE counter SET c = c + 1;
    RETURN (SELECT c FROM counter LIMIT 1);
END //
DELIMITER ;
```

Character set and collation:

```
CREATE FUNCTION hello2 (s CHAR(20))
RETURNS CHAR(50) CHARACTER SET 'utf8' COLLATE 'utf8_bin' DETERMINISTIC
RETURN CONCAT('Hello, ',s,'!');
```

## See Also

- [Identifier Names](#)
- [Stored Aggregate Functions](#)
- [CREATE FUNCTION \(UDF\)](#)
- [SHOW CREATE FUNCTION](#)
- [ALTER FUNCTION](#)
- [DROP FUNCTION](#)
- [SHOW FUNCTION STATUS](#)
- [Stored Routine Privileges](#)
- [Information Schema ROUTINES Table](#)

### 1.1.3.1.4 CREATE FUNCTION UDF

## Syntax

```
CREATE [OR REPLACE] [AGGREGATE] FUNCTION [IF NOT EXISTS] function_name
RETURNS {STRING|INTEGER|REAL|DECIMAL}
SONAME shared_library_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [RETURNS](#)
  2. [shared\\_library\\_name](#)
  3. [AGGREGATE](#)
  4. [OR REPLACE](#)
  5. [IF NOT EXISTS](#)
  6. [Upgrading a UDF](#)
  7. [Examples](#)
3. [See Also](#)

## Description

A user-defined function (UDF) is a way to extend MariaDB with a new function that works like a native (built-in) MariaDB function such as `ABS()` or `CONCAT()`.

`function_name`  
is the name that should be used in SQL statements to invoke the function.

To create a function, you must have the `INSERT` privilege for the `mysql` database. This is necessary because

```
CREATE FUNCTION
adds a row to the mysql.func system table that records the function's name, type, and shared library name. If you do not have this table, you
should run the mysql_upgrade command to create it.
```

UDFs need to be written in C, C++ or another language that uses C calling conventions, MariaDB needs to have been dynamically compiled, and your operating system must support dynamic loading.

For an example, see

Statements making use of user-defined functions are not [safe for replication](#).

For creating a stored function as opposed to a user-defined function, see [CREATE FUNCTION](#).

For valid identifiers to use as function names, see [Identifier Names](#).

## RETURNS

The

```
RETURNS
clause indicates the type of the function's return value, and can be one of STRING , INTEGER , REAL or DECIMAL .
DECIMAL
functions currently return string values and should be written like STRING functions.
```

## shared\_library\_name

```
shared_library_name
```

is the basename of the shared object file that contains the code that implements the function. The file must be located in the plugin directory. This directory is given by the value of the [plugin\\_dir](#) system variable. Note that before MariaDB/MySQL 5.1, the shared object could be located in any directory that was searched by your system's dynamic linker.

## AGGREGATE

Aggregate functions are summary functions such as [SUM\(\)](#) and [AVG\(\)](#).

MariaDB starting with [10.4](#)

Aggregate UDF functions can be used as [window functions](#).

## OR REPLACE

MariaDB starting with [10.1.3](#)

The

```
OR REPLACE
clause was added in MariaDB 10.1.3
```

If the optional

```
OR REPLACE
clause is used, it acts as a shortcut for:
```

```
DROP FUNCTION IF EXISTS function_name;
CREATE FUNCTION name ...;
```

## IF NOT EXISTS

MariaDB starting with [10.1.3](#)

The

```
IF NOT EXISTS
clause was added in MariaDB 10.1.3
```

When the IF NOT EXISTS clause is used, MariaDB will return a warning instead of an error if the specified function already exists. Cannot be used together with OR REPLACE.

## Upgrading a UDF

To upgrade the UDF's shared library, first run a [DROP FUNCTION](#) statement, then upgrade the shared library and finally run the [CREATE FUNCTION](#) statement. If you upgrade without following this process, you may crash the server.

## Examples

```
CREATE FUNCTION jsoncontains_path RETURNS integer SONAME 'ha_connect.so';
Query OK, 0 rows affected (0.00 sec)
```

OR REPLACE and IF NOT EXISTS:

```
CREATE FUNCTION jsoncontains_path RETURNS integer SONAME 'ha_connect.so';
ERROR 1125 (HY000): Function 'jsoncontains_path' already exists

CREATE OR REPLACE FUNCTION jsoncontains_path RETURNS integer SONAME 'ha_connect.so';
Query OK, 0 rows affected (0.00 sec)

CREATE FUNCTION IF NOT EXISTS jsoncontains_path RETURNS integer SONAME 'ha_connect.so';
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1125 | Function 'jsoncontains_path' already exists |
+-----+-----+
```

## See Also

- Identifier Names
- DROP FUNCTION
- CREATE FUNCTION

### 1.1.3.1.5 CREATE INDEX

#### Syntax

```
CREATE [OR REPLACE] [UNIQUE|FULLTEXT|SPATIAL] INDEX
[IF NOT EXISTS] index_name
[index_type]
ON tbl_name (index_col_name,...)
[WAIT n | NOWAIT]
[index_option]
[algorithm_option | lock_option] ...

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH | RTREE}

index_option:
    [ KEY_BLOCK_SIZE [=] value
    | index_type
    | WITH PARSER parser_name
    | COMMENT 'string'
    | CLUSTERING={YES| NO} ]
    [ IGNORED | NOT IGNORED ]

algorithm_option:
    ALGORITHM [=] {DEFAULT|INPLACE|COPY|NOCOPY|INSTANT}

lock_option:
    LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Privileges](#)
4. [Online DDL](#)
5. [CREATE OR REPLACE INDEX](#)
6. [CREATE INDEX IF NOT EXISTS](#)
7. [Index Definitions](#)
8. [WAIT/NOWAIT](#)
9. [ALGORITHM](#)
10. [LOCK](#)
11. [Progress Reporting](#)
12. [WITHOUT OVERLAPS](#)
13. [Examples](#)
14. [See Also](#)

## Description

CREATE INDEX is mapped to an ALTER TABLE statement to create indexes . See [ALTER TABLE](#) . CREATE INDEX cannot be used to create a PRIMARY KEY; use ALTER TABLE instead.

If another connection is using the table, a [metadata lock](#) is active, and this statement will wait until the lock is released. This is also true for non-transactional tables.

Another shortcut, [DROP INDEX](#) , allows the removal of an index.

For valid identifiers to use as index names, see [Identifier Names](#) .

Note that `KEY_BLOCK_SIZE` is currently ignored in CREATE INDEX, although it is included in the output of [SHOW CREATE TABLE](#) .

## Privileges

Executing the

`CREATE INDEX`  
statement requires the

`INDEX`

privilege for the table or the database.

## Online DDL

Online DDL is supported with the [ALGORITHM](#) and [LOCK](#) clauses.

See [InnoDB Online DDL Overview](#) for more information on online DDL with InnoDB .

## CREATE OR REPLACE INDEX

MariaDB starting with [10.1.4](#)

The

`OR REPLACE`  
clause was added in [MariaDB 10.1.4](#) .

If the

`OR REPLACE`  
clause is used and if the index already exists, then instead of returning an error, the server will drop the existing index and replace it with the newly defined index.

## CREATE INDEX IF NOT EXISTS

If the

`IF NOT EXISTS`  
clause is used, then the index will only be created if an index with the same name does not already exist. If the index already exists, then a warning will be triggered by default.

## Index Definitions

See [CREATE TABLE: Index Definitions](#) for information about index definitions.

## WAIT/NOWAIT

MariaDB starting with [10.3.0](#)

Set the lock wait timeout. See [WAIT](#) and [NOWAIT](#).

## ALGORITHM

See [ALTER TABLE: ALGORITHM](#) for more information.

## LOCK

See [ALTER TABLE: LOCK](#) for more information.

## Progress Reporting

MariaDB provides progress reporting for

`CREATE INDEX` statement for clients that support the new progress reporting protocol. For example, if you were using the

`mysql`

client, then the progress report might look like this::

```
CREATE INDEX ON tab (num);;
Stage: 1 of 2 'copy to tmp table'    46% of stage
```

The progress report is also shown in the output of the

`SHOW PROCESSLIST`

statement and in the contents of the

`information_schema.PROCESSLIST`

table.

See [Progress Reporting](#) for more information.

## WITHOUT OVERLAPS

MariaDB starting with [10.5.3](#)

The `WITHOUT OVERLAPS` clause allows one to constrain a primary or unique index such that [application-time periods](#) cannot overlap.

## Examples

Creating a unique index:

```
CREATE UNIQUE INDEX HomePhone ON Employees(Home_Phone);
```

OR REPLACE and IF NOT EXISTS:

```
CREATE INDEX xi ON xx5 (x);
Query OK, 0 rows affected (0.03 sec)

CREATE INDEX xi ON xx5 (x);
ERROR 1061 (42000): Duplicate key name 'xi'

CREATE OR REPLACE INDEX xi ON xx5 (x);
Query OK, 0 rows affected (0.03 sec)

CREATE INDEX IF NOT EXISTS xi ON xx5 (x);
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message           |
+-----+-----+
| Note  | 1061 | Duplicate key name 'xi' |
+-----+-----+
```

From MariaDB 10.5.3 , creating a unique index for an application-time period table with a WITHOUT OVERLAPS constraint:

```
CREATE UNIQUE INDEX u ON rooms (room_number, p WITHOUT OVERLAPS);
```

## See Also

- [Identifier Names](#)
- [Getting Started with Indexes](#)
- [What is an Index?](#)
- [ALTER TABLE](#)
- [DROP INDEX](#)
- [SHOW INDEX](#)
- [SPATIAL INDEX](#)
- [Full-text Indexes](#)
- [WITHOUT OVERLAPS](#)
- [Ignored Indexes](#)

### 1.1.3.1.6 CREATE LOGFILE GROUP

The

CREATE LOGFILE GROUP

statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. See [MDEV-19295](#) for more information.

### 1.1.3.1.7 CREATE PACKAGE

MariaDB starting with 10.3.5

Oracle-style packages were introduced in [MariaDB 10.3.5](#).

## Syntax

```

CREATE
  [ OR REPLACE]
  [DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
PACKAGE [ IF NOT EXISTS ]
[ db_name . ] package_name
[ package_characteristic ... ]
{ AS | IS }
[ package_specification_element ... ]
END [ package_name ]

package_characteristic:
  COMMENT 'string'
| SQL SECURITY { DEFINER | INVOKER }

package_specification_element:
  FUNCTION_SYM package_specification_function ;
| PROCEDURE_SYM package_specification_procedure ;

package_specification_function:
  func_name [ ( func_param [, func_param]... ) ]
RETURNS func_return_type
[ package_routine_characteristic... ]

package_specification_procedure:
  proc_name [ ( proc_param [, proc_param]... ) ]
[ package_routine_characteristic... ]

func_return_type:
  type

func_param:
  param_name [ IN | OUT | INOUT | IN OUT ] type

proc_param:
  param_name [ IN | OUT | INOUT | IN OUT ] type

type:
  Any valid MariaDB explicit or anchored data type

package_routine_characteristic:
  COMMENT 'string'
| LANGUAGE SQL
| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
| SQL SECURITY { DEFINER | INVOKER }

```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Function parameter quantifiers IN | OUT |  
INOUT | IN OUT](#)
4. [Examples](#)
5. [See Also](#)

## Description

The

```
CREATE PACKAGE
```

statement can be used when [Oracle SQL\\_MODE](#) is set.

The

```
CREATE PACKAGE
```

creates the specification for a stored package (a collection of logically related stored objects). A stored package specification declares public routines (procedures and functions) of the package, but does not implement these routines.

A package whose specification was created by the

```
CREATE PACKAGE
```

statement, should later be implemented using the [CREATE PACKAGE BODY](#) statement.

## Function parameter quantifiers IN | OUT | INOUT | IN OUT

MariaDB starting with 10.8.0

The function parameter quantifiers for

```
IN  
,  
OUT  
,  
INOUT  
, and  
IN OUT
```

where added in a 10.8.0 preview release. Prior to 10.8.0 quantifiers were supported only in procedures.

```
OUT  
,  
INOUT  
and its equivalent  
IN OUT  
, are only valid if called from  
SET  
and not  
SELECT
```

. These quantifiers are especially useful for creating functions and procedures with more than one return value. This allows functions and procedures to be more complex and nested.

## Examples

```
SET sql_mode=ORACLE;  
DELIMITER $$  
CREATE OR REPLACE PACKAGE employee_tools AS  
    FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2);  
    PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2));  
    PROCEDURE raiseSalaryStd(eid INT);  
    PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2));  
END;  
$$  
DELIMITER ;
```

## See Also

- [CREATE PACKAGE BODY](#)
- [SHOW CREATE PACKAGE](#)
- [DROP PACKAGE](#)
- [Oracle SQL\\_MODE](#)

### 1.1.3.1.8 CREATE PACKAGE BODY

MariaDB starting with 10.3.5

Oracle-style packages were introduced in [MariaDB 10.3.5](#).

## Syntax

```
CREATE [ OR REPLACE ]  
[DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]  
PACKAGE BODY  
[ IF NOT EXISTS ]  
[ db_name . ] package_name  
[ package_characteristic... ]  
{ AS | IS }  
    packageImplementation_declare_section  
    packageImplementation_executable_section  
END [ package_name ]  
  
packageImplementation_declare_section:  
    packageImplementation_item_declaration  
    [ packageImplementation_item_declaration... ]
```

```

[ packageImplementationRoutineDefinition... ]
| packageImplementationRoutineDefinition
  [ packageImplementationRoutineDefinition... ]

packageImplementationItemDeclaration:
  variableDeclaration ;

variableDeclaration:
  variableName[,...] type [:= expr] ;

packageImplementationRoutineDefinition:
  FUNCTION packageSpecificationFunction
    [ packageImplementationFunctionBody ] ;
  | PROCEDURE packageSpecificationProcedure
    [ packageImplementationProcedureBody ] ;

packageImplementationFunctionBody:
  { AS | IS } packageRoutineBody [func_name]

packageImplementationProcedureBody:
  { AS | IS } packageRoutineBody [proc_name]

packageRoutineBody:
  [ packageRoutineDeclarations ]
  BEGIN
    statements [ EXCEPTION exceptionHandlers ]
  END

packageRoutineDeclarations:
  packageRoutineDeclaration ';' [packageRoutineDeclaration ';' ]...

packageRoutineDeclaration:
  variableDeclaration
  | conditionName CONDITION FOR conditionValue
  | userExceptionName EXCEPTION
  | CURSOR_SYM cursorName
    [ ( cursorFormalParameters ) ]
  IS selectStatement
;

packageImplementationExecutableSection:
  END
  | BEGIN
    statement ; [statement ; ]...
    [EXCEPTION exceptionHandlers]
  END

exceptionHandlers:
  exceptionHandler [exceptionHandler...]

exceptionHandler:
  WHEN_SYM conditionValue [, conditionValue]...
  THEN_SYM statement ; [statement ; ]...

conditionValue:
  conditionName
  | userExceptionName
  | SQLWARNING
  | SQLEXCEPTION
  | NOT FOUND
  | OTHERS_SYM
  | SQLSTATE [VALUE] sqlstateValue
  | mariadb_errorCode

```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

# Description

The

```
CREATE PACKAGE BODY  
statement can be used when Oracle SQL_MODE is set.
```

The

```
CREATE PACKAGE BODY  
statement creates the package body for a stored package. The package specification must be previously created using the CREATE PACKAGE statement.
```

A package body provides implementations of the package public routines and can optionally have:

- package-wide private variables
- package private routines
- forward declarations for private routines
- an executable initialization section

## Examples

```

SET sql_mode=ORACLE;
DELIMITER $$

CREATE OR REPLACE PACKAGE employee_tools AS
  FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2);
  PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2));
  PROCEDURE raiseSalaryStd(eid INT);
  PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2));
END;
$$

CREATE PACKAGE BODY employee_tools AS
  -- package body variables
  stdRaiseAmount DECIMAL(10,2):=500;

  -- private routines
  PROCEDURE log (eid INT, ecmnt TEXT) AS
  BEGIN
    INSERT INTO employee_log (id, cmnt) VALUES (eid, ecmnt);
  END;

  -- public routines
  PROCEDURE hire(ename TEXT, esalary DECIMAL(10,2)) AS
    eid INT;
  BEGIN
    INSERT INTO employee (name, salary) VALUES (ename, esalary);
    eid:= last_insert_id();
    log(eid, 'hire ' || ename);
  END;

  FUNCTION getSalary(eid INT) RETURN DECIMAL(10,2) AS
    nSalary DECIMAL(10,2);
  BEGIN
    SELECT salary INTO nSalary FROM employee WHERE id=eid;
    log(eid, 'getSalary id=' || eid || ' salary=' || nSalary);
    RETURN nSalary;
  END;

  PROCEDURE raiseSalary(eid INT, amount DECIMAL(10,2)) AS
  BEGIN
    UPDATE employee SET salary=salary+amount WHERE id=eid;
    log(eid, 'raiseSalary id=' || eid || ' amount=' || amount);
  END;

  PROCEDURE raiseSalaryStd(eid INT) AS
  BEGIN
    raiseSalary(eid, stdRaiseAmount);
    log(eid, 'raiseSalaryStd id=' || eid);
  END;

BEGIN
  -- This code is executed when the current session
  -- accesses any of the package routines for the first time
  log(0, 'Session ' || connection_id() || ' ' || current_user || ' started');
END;
$$

DELIMITER ;

```

## See Also

- [CREATE PACKAGE](#)
- [SHOW CREATE PACKAGE BODY](#)
- [DROP PACKAGE BODY](#)
- [Oracle SQL\\_MODE](#)

## 1.1.3.1.9 CREATE PROCEDURE

### Syntax

```

CREATE
[OR REPLACE]
[DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
PROCEDURE sp_name ([proc_parameter[,...]])
[characteristic ...] routine_body

proc_parameter:
[ IN | OUT | INOUT ] param_name type

type:
Any valid MariaDB data type

characteristic:
LANGUAGE SQL
| [NOT] DETERMINISTIC
| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
| SQL SECURITY { DEFINER | INVOKER }
| COMMENT 'string'

routine_body:
Valid SQL procedure statement

```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [IN/OUT/INOUT](#)
  2. [DETERMINISTIC/NOT DETERMINISTIC](#)
  3. [CONTAINS SQL/NO SQL/READS SQL DATA/MODIFIES SQL DATA](#)
  4. [Invoking stored procedure from within programs](#)
  5. [OR REPLACE](#)
  6. [sql\\_mode](#)
  7. [Character Sets and Collations](#)
  8. [Oracle Mode](#)
3. [Examples](#)
4. [See Also](#)

## Description

Creates a [stored procedure](#). By default, a routine is associated with the default database. To associate the routine explicitly with a given database, specify the name as db\_name.sp\_name when you create it.

When the routine is invoked, an implicit USE db\_name is performed (and undone when the routine terminates). This causes the routine to have the given default database while it executes. USE statements within stored routines are disallowed.

When a stored procedure has been created, you invoke it by using the

```
CALL
statement (see CALL ).
```

To execute the

```
CREATE PROCEDURE
statement, it is necessary to have the
CREATE ROUTINE
privilege. By default, MariaDB automatically grants the
ALTER ROUTINE
and
EXECUTE
privileges to the routine creator. See also Stored Routine Privileges.
```

The

```
DEFINER
```

and SQL SECURITY clauses specify the security context to be used when checking access privileges at routine execution time, as described later. Requires the [SUPER](#) privilege, or, from [MariaDB 10.5.2](#), the [SET USER](#) privilege.

If the routine name is the same as the name of a built-in SQL function, you must use a space between the name and the following parenthesis when defining the routine, or a syntax error occurs. This is also true when you invoke the routine later. For this reason, we suggest that it is better to avoid re-using the names of existing SQL functions for your own stored routines.

The IGNORE\_SPACE SQL mode applies to built-in functions, not to stored routines. It is always allowable to have spaces after a routine name, regardless of whether IGNORE\_SPACE is enabled.

The parameter list enclosed within parentheses must always be present. If there are no parameters, an empty parameter list of () should be used.

Parameter names are not case sensitive.

Each parameter can be declared to use any valid data type, except that the COLLATE attribute cannot be used.

For valid identifiers to use as procedure names, see [Identifier Names](#).

## IN/OUT/INOUT

Each parameter is an

IN  
parameter by default. To specify otherwise for a parameter, use the keyword  
OUT  
or  
INOUT  
before the parameter name.

An

IN  
parameter passes a value into a procedure. The procedure might modify the value, but the modification is not visible to the caller when the procedure returns. An  
OUT  
parameter passes a value from the procedure back to the caller. Its initial value is NULL within the procedure, and its value is visible to the caller when the procedure returns. An  
INOUT  
parameter is initialized by the caller, can be modified by the procedure, and any change made by the procedure is visible to the caller when the procedure returns.

For each

OUT  
or  
INOUT  
parameter, pass a user-defined variable in the  
CALL  
statement that invokes the procedure so that you can obtain its value when the procedure returns. If you are calling the procedure from within another stored procedure or function, you can also pass a routine parameter or local routine variable as an  
IN  
or  
INOUT  
parameter.

## DETERMINISTIC/NOT DETERMINISTIC

DETERMINISTIC  
and  
NOT DETERMINISTIC  
apply only to [functions](#). Specifying  
DETERMINISTIC  
or  
NON-DETERMINISTIC  
in procedures has no effect. The default value is  
NOT DETERMINISTIC  
. Functions are  
DETERMINISTIC  
when they always return the same value for the same input. For example, a truncate or substring function. Any function involving data, therefore, is always  
NOT DETERMINISTIC

## CONTAINS SQL/NO SQL/READS SQL DATA/MODIFIES SQL DATA

CONTAINS SQL  
,  
NO SQL  
,  
READS SQL DATA  
, and  
MODIFIES SQL DATA  
are informative clauses that tell the server what the function does. MariaDB does not check in any way whether the specified clause is correct. If none of these clauses are specified,

`CONTAINS SQL`  
is used by default.

`MODIFIES SQL DATA`  
means that the function contains statements that may modify data stored in databases. This happens if the function contains statements like `DELETE`, `UPDATE`, `INSERT`, `REPLACE` or DDL.

`READS SQL DATA`  
means that the function reads data stored in databases, but does not modify any data. This happens if `SELECT` statements are used, but there no write operations are executed.

`CONTAINS SQL`  
means that the function contains at least one SQL statement, but it does not read or write any data stored in a database. Examples include `SET` or `DO`.

`NO SQL`  
means nothing, because MariaDB does not currently support any language other than SQL.

The routine \_body consists of a valid SQL procedure statement. This can be a simple statement such as `SELECT` or `INSERT`, or it can be a compound statement written using `BEGIN` and `END`. Compound statements can contain declarations, loops, and other control structure statements. See [Programmatic and Compound Statements](#) for syntax details.

MariaDB allows routines to contain DDL statements, such as

`CREATE`  
and `DROP`. MariaDB also allows `stored procedures` (but not `stored functions`) to contain SQL transaction statements such as  
`COMMIT`

For additional information about statements that are not allowed in stored routines, see [Stored Routine Limitations](#).

## Invoking stored procedure from within programs

For information about invoking `stored procedures` from within programs written in a language that has a MariaDB/MySQL interface, see [CALL](#).

## OR REPLACE

MariaDB starting with 10.1.3

If the optional

`OR REPLACE`

clause is used, it acts as a shortcut for:

```
DROP PROCEDURE IF EXISTS name;
CREATE PROCEDURE name ...;
```

with the exception that any existing `privileges` for the procedure are not dropped.

## sql\_mode

MariaDB stores the `sql_mode` system variable setting that is in effect at the time a routine is created, and always executes the routine with this setting in force, regardless of the server `SQL mode` in effect when the routine is invoked.

## Character Sets and Collations

Procedure parameters can be declared with any character set/collation. If the character set and collation are not specifically set, the database defaults at the time of creation will be used. If the database defaults change at a later stage, the stored procedure character set/collation will not be changed at the same time; the stored procedure needs to be dropped and recreated to ensure the same character set/collation as the database is used.

## Oracle Mode

MariaDB starting with 10.3

From [MariaDB 10.3](#), a subset of Oracle's PL/SQL language has been supported in addition to the traditional SQL/PSM-based MariaDB syntax. See [Oracle mode from MariaDB 10.3](#) for details on changes when running Oracle mode.

## Examples

The following example shows a simple stored procedure that uses an

OUT

parameter. It uses the DELIMITER command to set a new delimiter for the duration of the process — see [Delimiters in the mysql client](#).

```
DELIMITER //  
  
CREATE PROCEDURE simpleproc (OUT param1 INT)  
BEGIN  
    SELECT COUNT(*) INTO param1 FROM t;  
END;  
//  
  
DELIMITER ;  
  
CALL simpleproc(@a);  
  
SELECT @a;  
+-----+  
| @a   |  
+-----+  
|     1 |  
+-----+
```

Character set and collation:

```
DELIMITER //  
  
CREATE PROCEDURE simpleproc2 (  
    OUT param1 CHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_bin'  
)  
BEGIN  
    SELECT CONCAT('a'),f1 INTO param1 FROM t;  
END;  
//  
  
DELIMITER ;
```

CREATE OR REPLACE:

```
DELIMITER //  
  
CREATE PROCEDURE simpleproc2 (  
    OUT param1 CHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_bin'  
)  
BEGIN  
    SELECT CONCAT('a'),f1 INTO param1 FROM t;  
END;  
//  
ERROR 1304 (42000): PROCEDURE simpleproc2 already exists  
  
DELIMITER ;  
  
DELIMITER //  
  
CREATE OR REPLACE PROCEDURE simpleproc2 (  
    OUT param1 CHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8_bin'  
)  
BEGIN  
    SELECT CONCAT('a'),f1 INTO param1 FROM t;  
END;  
//  
ERROR 1304 (42000): PROCEDURE simpleproc2 already exists  
  
DELIMITER ;  
Query OK, 0 rows affected (0.03 sec)
```

## See Also

- [Identifier Names](#)
- [Stored Procedure Overview](#)
- [ALTER PROCEDURE](#)
- [DROP PROCEDURE](#)
- [SHOW CREATE PROCEDURE](#)

- SHOW PROCEDURE STATUS
- Stored Routine Privileges
- Information Schema ROUTINES Table

## 1.1.3.1.10 CREATE ROLE

## 1.1.3.1.11 CREATE SEQUENCE

MariaDB starting with 10.3

CREATE SEQUENCE was introduced in MariaDB 10.3 .

### Syntax

```
CREATE [OR REPLACE] [TEMPORARY] SEQUENCE [IF NOT EXISTS] sequence_name
[ INCREMENT [ BY | = ] increment ]
[ MINVALUE [=] minvalue | NO MINVALUE | NOMINVALUE ]
[ MAXVALUE [=] maxvalue | NO MAXVALUE | NOMAXVALUE ]
[ START [ WITH | = ] start ]
[ CACHE [=] cache | NOCACHE ] [ CYCLE | NOCYCLE]
[table_options]
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [Arguments to Create](#)
  2. [Constraints on Create Arguments](#)
  3. [Atomic DDL](#)
3. [Examples](#)
4. [See Also](#)

The options for

`CREATE SEQUENCE`  
can be given in any order, optionally followed by  
`table_options`

`table_options` can be any of the normal table options in [CREATE TABLE](#) but the most usable ones are

`ENGINE=...`  
and  
`COMMENT=`

`NOMAXVALUE`  
and  
`NOMINVALUE`  
are there to allow one to create SEQUENCES using the Oracle syntax.

### Description

`CREATE SEQUENCE` will create a sequence that generates new values when called with

`NEXT VALUE FOR sequence_name`

. It's an alternative to [AUTO INCREMENT](#) when one wants to have more control of how the numbers are generated. As the SEQUENCE caches values (up to

`CACHE`

) it can in some cases be much faster than [AUTO INCREMENT](#) . Another benefit is that one can access the last value generated by all used sequences, which solves one of the limitations with [LAST\\_INSERT\\_ID\(\)](#) .

`CREATE SEQUENCE` requires the [CREATE privilege](#) .

`DROP SEQUENCE` can be used to drop a sequence, and `ALTER SEQUENCE` to change it.

### Arguments to Create

The following options may be used:

Option	Default value	Description
INCREMENT	1	Increment to use for values. May be negative. Setting an increment of 0 causes the sequence to use the value of the <a href="#">auto_increment_increment</a> system variable at the time of creation, which is always a positive number. (see <a href="#">MDEV-16035</a> ).
MINVALUE	1 if INCREMENT > 0 and -9223372036854775807 if INCREMENT < 0	Minimum value for the sequence
MAXVALUE	9223372036854775806 if INCREMENT > 0 and -1 if INCREMENT < 0	Max value for sequence
START	MINVALUE if INCREMENT > 0 and MAX_VALUE if INCREMENT < 0	First value that the sequence will generate
CACHE	1000	Number of values that should be cached. 0 if no CACHE. The underlying table will be updated first time a new sequence number is generated and each time the cache runs out.

If

CYCLE

is used then the sequence should start again from

MINVALUE

after it has run out of values. Default value is

NOCYCLE

## Constraints on Create Arguments

To be able to create a legal sequence, the following must hold:

- MAXVALUE >= start
- MAXVALUE > MINVALUE
- START >= MINVALUE
- MAXVALUE <= 9223372036854775806 (LONGLONG\_MAX-1)
- MINVALUE >= -9223372036854775807 (LONGLONG\_MIN+1)

Note that sequences can't generate the maximum/minimum 64 bit number because of the constraint of

MINVALUE

and

MAXVALUE

## Atomic DDL

MariaDB starting with [10.6.1](#)

MariaDB [10.6.1](#) supports Atomic DDL and

```
CREATE SEQUENCE
    is atomic.
```

## Examples

```
CREATE SEQUENCE s START WITH 100 INCREMENT BY 10;
CREATE SEQUENCE s2 START WITH -100 INCREMENT BY -10;
```

The following statement fails, as the increment conflicts with the defaults

```
CREATE SEQUENCE s3 START WITH -100 INCREMENT BY 10;
ERROR 4082 (HY000): Sequence 'test.s3' values are conflicting
```

The sequence can be created by specifying workable minimum and maximum values:

```
CREATE SEQUENCE s3 START WITH -100 INCREMENT BY 10 MINVALUE=-100 MAXVALUE=1000;
```

## See Also

- Sequence Overview
- ALTER SEQUENCE
- DROP SEQUENCE
- NEXT VALUE FOR
- PREVIOUS VALUE FOR
- SETVAL() . Set next value for the sequence.
- AUTO INCREMENT
- SHOW CREATE SEQUENCE

## 1.1.3.1.12 CREATE SERVER

### Syntax

```
CREATE [OR REPLACE] SERVER [IF NOT EXISTS] server_name
    FOREIGN DATA WRAPPER wrapper_name
    OPTIONS (option [, option] ...)

option:
{ HOST character-literal
| DATABASE character-literal
| USER character-literal
| PASSWORD character-literal
| SOCKET character-literal
| OWNER character-literal
| PORT numeric-literal }
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [OR REPLACE](#)
  2. [IF NOT EXISTS](#)
3. [Examples](#)
4. [See Also](#)

### Description

This statement creates the definition of a server for use with the [Spider](#) , [Connect](#) , [FEDERATED](#) or [FederatedX](#) storage engine. The CREATE SERVER statement creates a new row within the `servers` table within the `mysql` database. This statement requires the [SUPER](#) privilege or, from [MariaDB 10.5.2](#) , the [FEDERATED ADMIN](#) privilege.

The `server_name` should be a unique reference to the server. Server definitions are global within the scope of the server, it is not possible to qualify the server definition to a specific database. `server_name` has a maximum length of 64 characters (names longer than 64 characters are silently truncated), and is case insensitive. You may specify the name as a quoted string.

The `wrapper_name` may be quoted with single quotes. Supported values are:

- mysql
- mariadb  
(in [MariaDB 10.3](#) and later)

For each option you must specify either a character literal or numeric literal. Character literals are UTF-8, support a maximum length of 64 characters and default to a blank (empty) string. String literals are silently truncated to 64 characters. Numeric literals must be a number between 0 and 9999, default value is 0.

**Note :** The

OWNER  
option is currently not applied, and has no effect on the ownership or operation of the server connection that is created.

The CREATE SERVER statement creates an entry in the `mysql.servers` table that can later be used with the CREATE TABLE statement when creating a [Spider](#) , [Connect](#) , [FederatedX](#) or [FEDERATED](#) table. The options that you specify will be used to populate the columns in the `mysql.servers` table. The table columns are `Server_name`, `Host`, `Db`, `Username`, `Password`, `Port` and `Socket`.

[DROP SERVER](#) removes a previously created server definition.

CREATE SERVER is not written to the [binary log](#) , irrespective of the [binary log format](#) being used. From [MariaDB 10.1.13](#) , [Galera](#) replicates the CREATE SERVER, [ALTER SERVER](#) and [DROP SERVER](#) statements.

For valid identifiers to use as server names, see [Identifier Names](#) .

### OR REPLACE

If the optional  
OR REPLACE  
clause is used, it acts as a shortcut for:

```
DROP SERVER IF EXISTS name;  
CREATE SERVER server_name ...;
```

## IF NOT EXISTS

If the IF NOT EXISTS clause is used, MariaDB will return a warning instead of an error if the server already exists. Cannot be used together with OR REPLACE.

## Examples

```
CREATE SERVER s  
FOREIGN DATA WRAPPER mysql  
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');
```

OR REPLACE and IF NOT EXISTS:

```
CREATE SERVER s  
FOREIGN DATA WRAPPER mysql  
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');  
ERROR 1476 (HY000): The foreign server, s, you are trying to create already exists  
  
CREATE OR REPLACE SERVER s  
FOREIGN DATA WRAPPER mysql  
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');  
Query OK, 0 rows affected (0.00 sec)  
  
CREATE SERVER IF NOT EXISTS s  
FOREIGN DATA WRAPPER mysql  
OPTIONS (USER 'Remote', HOST '192.168.1.106', DATABASE 'test');  
Query OK, 0 rows affected, 1 warning (0.00 sec)  
  
SHOW WARNINGS;  
+-----+-----+  
| Level | Code | Message |  
+-----+-----+  
| Note  | 1476 | The foreign server, s, you are trying to create already exists |  
+-----+-----+
```

## See Also

- Identifier Names
- ALTER SERVER
- DROP SERVER
- Spider Storage Engine
- Connect Storage Engine

## 1.1.3.1.13 CREATE TABLE

## 1.1.3.1.14 CREATE TABLESPACE

The

```
CREATE TABLESPACE
```

statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. In MySQL 5.7 and later, the statement is also supported for InnoDB. However, MariaDB has chosen not to include that specific feature. See [MDEV-19294](#) for more information.

## 1.1.3.1.15 CREATE TRIGGER

## Syntax

```
CREATE [OR REPLACE]
[DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
TRIGGER [IF NOT EXISTS] trigger_name trigger_time trigger_event
ON tbl_name FOR EACH ROW
[ { FOLLOWES | PRECEDES } other_trigger_name ]
trigger_stmt;
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [OR REPLACE](#)
  2. [DEFINER](#)
  3. [IF NOT EXISTS](#)
  4. [trigger\\_time](#)
  5. [trigger\\_event](#)
    1. [FOLLOWES/PRECEDES](#)  
[other\\_trigger\\_name](#)
  6. [Atomic DDL](#)
3. [Examples](#)
4. [See Also](#)

## Description

This statement creates a new [trigger](#). A trigger is a named database object that is associated with a table, and that activates when a particular event occurs for the table. The trigger becomes associated with the table named

tbl\_name  
, which must refer to a permanent table. You cannot associate a trigger with a TEMPORARY table or a view.

CREATE TRIGGER  
requires the [TRIGGER](#) privilege for the table associated with the trigger.

MariaDB starting with [10.2.3](#)

You can have multiple triggers for the same

trigger\_time  
and  
trigger\_event

For valid identifiers to use as trigger names, see [Identifier Names](#).

## OR REPLACE

MariaDB starting with [10.1.4](#)

If used and the trigger already exists, instead of an error being returned, the existing trigger will be dropped and replaced by the newly defined trigger.

## DEFINER

The

DEFINER  
clause determines the security context to be used when checking access privileges at trigger activation time. Usage requires the [SUPER](#) privilege, or, from [MariaDB 10.5.2](#), the [SET USER](#) privilege.

## IF NOT EXISTS

MariaDB starting with [10.1.4](#)

If the

IF NOT EXISTS  
clause is used, the trigger will only be created if a trigger of the same name does not exist. If the trigger already exists, by default a warning will be returned.

## trigger\_time

`trigger_time`  
is the trigger action time. It can be  
`BEFORE`  
or  
`AFTER`  
to indicate that the trigger activates before or after each row to be modified.

## trigger\_event

`trigger_event`  
indicates the kind of statement that activates the trigger. The  
`trigger_event`  
can be one of the following:

- `INSERT`  
: The trigger is activated whenever a new row is inserted into the table; for example, through `INSERT`, `LOAD DATA`, and `REPLACE` statements.
- `UPDATE`  
: The trigger is activated whenever a row is modified; for example, through `UPDATE` statements.
- `DELETE`  
: The trigger is activated whenever a row is deleted from the table; for example, through `DELETE` and `REPLACE` statements. However,  
`DROP TABLE`  
and  
`TRUNCATE`  
statements on the table do not activate this trigger, because they do not use  
`DELETE`  
. Dropping a partition does not activate  
`DELETE`  
triggers, either.

## FOLLOWNS/PRECEDES other\_trigger\_name

MariaDB starting with [10.2.3](#)

The

`FOLLOWNS other_trigger_name`  
and  
`PRECEDES other_trigger_name`  
options were added in [MariaDB 10.2.3](#) as part of supporting multiple triggers per action time. This is the same syntax used by MySQL 5.7, although MySQL 5.7 does not have multi-trigger support.

`FOLLOWNS`  
adds the new trigger after another trigger while  
`PRECEDES`  
adds the new trigger before another trigger. If neither option is used, the new trigger is added last for the given action and time.

`FOLLOWNS`  
and  
`PRECEDES`  
are not stored in the trigger definition. However the trigger order is guaranteed to not change over time. `mariadb-dump/mysqldump` and other backup methods will not change trigger order. You can verify the trigger order from the  
`ACTION_ORDER`  
column in `INFORMATION_SCHEMA.TRIGGERS` table.

```
SELECT trigger_name, action_order FROM information_schema.triggers
WHERE event_object_table='t1';
```

## Atomic DDL

MariaDB starting with [10.6.1](#)

[MariaDB 10.6.1](#) supports **Atomic DDL** and  
`CREATE TRIGGER`

is atomic.

## Examples

```
CREATE DEFINER=`root`@`localhost` TRIGGER increment_animal
AFTER INSERT ON animals FOR EACH ROW
UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
```

OR REPLACE and IF NOT EXISTS

```
CREATE DEFINER=`root`@`localhost` TRIGGER increment_animal
AFTER INSERT ON animals FOR EACH ROW
UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
ERROR 1359 (HY000): Trigger already exists

CREATE OR REPLACE DEFINER=`root`@`localhost` TRIGGER increment_animal
AFTER INSERT ON animals FOR EACH ROW
UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
Query OK, 0 rows affected (0.12 sec)

CREATE DEFINER=`root`@`localhost` TRIGGER IF NOT EXISTS increment_animal
AFTER INSERT ON animals FOR EACH ROW
UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message      |
+-----+-----+
| Note  | 1359 | Trigger already exists |
+-----+-----+
1 row in set (0.00 sec)
```

## See Also

- Identifier Names
- Trigger Overview
- DROP TRIGGER
- Information Schema TRIGGERS Table
- SHOW TRIGGERS
- SHOW CREATE TRIGGER
- Trigger Limitations

### 1.1.3.1.16 CREATE USER

### 1.1.3.1.17 CREATE VIEW

## Syntax

```
CREATE
[OR REPLACE]
[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
[DEFINER = { user | CURRENT_USER | role | CURRENT_ROLE }]
[SQL SECURITY { DEFINER | INVOKER }]
VIEW [IF NOT EXISTS] view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

## Contents

- 1. [Syntax](#)
- 2. [Description](#)
  - 1. [WITH CHECK OPTION](#)
  - 2. [IF NOT EXISTS](#)
  - 3. [Atomic DDL](#)
- 3. [Examples](#)
- 4. [See Also](#)

## Description

The CREATE VIEW statement creates a new [view](#), or replaces an existing one if the OR REPLACE clause is given. If the view does not exist, CREATE OR REPLACE VIEW is the same as CREATE VIEW. If the view does exist, CREATE OR REPLACE VIEW is the same as [ALTER VIEW](#).

The select\_statement is a [SELECT](#) statement that provides the definition of the view. (When you select from the view, you select in effect using the SELECT statement.) select\_statement can select from base tables or other views.

The view definition is "frozen" at creation time, so changes to the underlying tables afterwards do not affect the view definition. For example, if a view is defined as SELECT \* on a table, new columns added to the table later do not become part of the view. A [SHOW CREATE VIEW](#) shows that such queries are rewritten and column names are included in the view definition.

The view definition must be a query that does not return errors at view creation times. However, the base tables used by the views might be altered later and the query may not be valid anymore. In this case, querying the view will result in an error. [CHECK TABLE](#) helps in finding this kind of problems.

The [ALGORITHM clause](#) affects how MariaDB processes the view. The DEFINER and SQL SECURITY clauses specify the security context to be used when checking access privileges at view invocation time. The WITH CHECK OPTION clause can be given to constrain inserts or updates to rows in tables referenced by the view. These clauses are described later in this section.

The CREATE VIEW statement requires the CREATE VIEW privilege for the view, and some privilege for each column selected by the SELECT statement. For columns used elsewhere in the SELECT statement you must have the SELECT privilege. If the OR REPLACE clause is present, you must also have the DROP privilege for the view.

A view belongs to a database. By default, a new view is created in the default database. To create the view explicitly in a given database, specify the name as db\_name.view\_name when you create it.

```
CREATE VIEW test.v AS SELECT * FROM t;
```

Base tables and views share the same namespace within a database, so a database cannot contain a base table and a view that have the same name.

Views must have unique column names with no duplicates, just like base tables. By default, the names of the columns retrieved by the SELECT statement are used for the view column names. To define explicit names for the view columns, the optional column\_list clause can be given as a list of comma-separated identifiers. The number of names in column\_list must be the same as the number of columns retrieved by the SELECT statement.

### MySQL until 5.1.28

Prior to MySQL 5.1.29, When you modify an existing view, the current view definition is backed up and saved. It is stored in that table's database directory, in a subdirectory named arc. The backup file for a view v is named v.frm-00001. If you alter the view again, the next backup is named v.frm-00002. The three latest view backup definitions are stored. Backed up view definitions are not preserved by [mysqldump](#), or any other such programs, but you can retain them using a file copy operation. However, they are not needed for anything but to provide you with a backup of your previous view definition. It is safe to remove these backup definitions, but only while mysqld is not running. If you delete the arc subdirectory or its files while mysqld is running, you will receive an error the next time you try to alter the view:

```
MariaDB [test]> ALTER VIEW v AS SELECT * FROM t;
ERROR 6 (HY000): Error on delete of '.\test\arc\v.frm-0004' (Errcode: 2)
```

Columns retrieved by the SELECT statement can be simple references to table columns. They can also be expressions that use functions, constant values, operators, and so forth.

Unqualified table or view names in the SELECT statement are interpreted with respect to the default database. A view can refer to tables or views in other databases by qualifying the table or view name with the proper database name.

A view can be created from many kinds of SELECT statements. It can refer to base tables or other views. It can use joins, UNION, and subqueries. The SELECT need not even refer to any tables. The following example defines a view that selects two columns from another table, as well as an expression calculated from those columns:

```

CREATE TABLE t (qty INT, price INT);

INSERT INTO t VALUES(3, 50);

CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;

SELECT * FROM v;
+-----+-----+
| qty | price | value |
+-----+-----+
|    3 |     50 |    150 |
+-----+-----+

```

A view definition is subject to the following restrictions:

- The SELECT statement cannot contain a subquery in the FROM clause.
- The SELECT statement cannot refer to system or user variables.
- Within a stored program, the definition cannot refer to program parameters or local variables.
- The SELECT statement cannot refer to prepared statement parameters.
- Any table or view referred to in the definition must exist. However, after a view has been created, it is possible to drop a table or view that the definition refers to. In this case, use of the view results in an error. To check a view definition for problems of this kind, use the CHECK TABLE statement.
- The definition cannot refer to a TEMPORARY table, and you cannot create a TEMPORARY view.
- Any tables named in the view definition must exist at definition time.
- You cannot associate a trigger with a view.
- For valid identifiers to use as view names, see [Identifier Names](#).

ORDER BY is allowed in a view definition, but it is ignored if you select from a view using a statement that has its own ORDER BY.

For other options or clauses in the definition, they are added to the options or clauses of the statement that references the view, but the effect is undefined. For example, if a view definition includes a LIMIT clause, and you select from the view using a statement that has its own LIMIT clause, it is undefined which limit applies. This same principle applies to options such as ALL, DISTINCT, or SQL\_SMALL\_RESULT that follow the SELECT keyword, and to clauses such as INTO, FOR UPDATE, and LOCK IN SHARE MODE.

The PROCEDURE clause cannot be used in a view definition, and it cannot be used if a view is referenced in the FROM clause.

If you create a view and then change the query processing environment by changing system variables, that may affect the results that you get from the view:

```

CREATE VIEW v (mycol) AS SELECT 'abc';

SET sql_mode = '';

SELECT "mycol" FROM v;
+-----+
| mycol |
+-----+
| mycol |
+-----+

SET sql_mode = 'ANSI_QUOTES';

SELECT "mycol" FROM v;
+-----+
| mycol |
+-----+
| abc   |
+-----+

```

The DEFINER and SQL SECURITY clauses determine which MariaDB account to use when checking access privileges for the view when a statement is executed that references the view. They were added in MySQL 5.1.2. The legal SQL SECURITY characteristic values are DEFINER and INVOKER. These indicate that the required privileges must be held by the user who defined or invoked the view, respectively. The default SQL SECURITY value is DEFINER.

If a user value is given for the DEFINER clause, it should be a MariaDB account in 'user\_name'@'host\_name' format (the same format used in the GRANT statement). The user\_name and host\_name values both are required. The definer can also be given as CURRENT\_USER or CURRENT\_USER(). The default DEFINER value is the user who executes the CREATE VIEW statement. This is the same as specifying DEFINER = CURRENT\_USER explicitly.

If you specify the DEFINER clause, these rules determine the legal DEFINER user values:

- If you do not have the [SUPER](#) privilege, or, from [MariaDB 10.5.2](#), the [SET USER](#) privilege, the only legal user value is your own account, either specified literally or by using CURRENT\_USER. You cannot set the definer to some other account.
- If you have the [SUPER](#) privilege, or, from [MariaDB 10.5.2](#), the [SET USER](#) privilege, you can specify any syntactically legal account name. If the account does not actually exist, a warning is generated.
- If the SQL SECURITY value is DEFINER but the definer account does not exist when the view is referenced, an error occurs.

Within a view definition, CURRENT\_USER returns the view's DEFINER value by default. For views defined with the SQL SECURITY INVOKER characteristic, CURRENT\_USER returns the account for the view's invoker. For information about user auditing within views, see <http://dev.mysql.com/doc/refman/5.1/en/account-activity-auditing.html>.

Within a stored routine that is defined with the SQL SECURITY DEFINER characteristic, CURRENT\_USER returns the routine's DEFINER value. This also affects a view defined within such a program, if the view definition contains a DEFINER value of CURRENT\_USER.

View privileges are checked like this:

- At view definition time, the view creator must have the privileges needed to use the top-level objects accessed by the view. For example, if the view definition refers to table columns, the creator must have privileges for the columns, as described previously. If the definition refers to a stored function, only the privileges needed to invoke the function can be checked. The privileges required when the function runs can be checked only as it executes: For different invocations of the function, different execution paths within the function might be taken.
- When a view is referenced, privileges for objects accessed by the view are checked against the privileges held by the view creator or invoker, depending on whether the SQL SECURITY characteristic is DEFINER or INVOKER, respectively.
- If reference to a view causes execution of a stored function, privilege checking for statements executed within the function depend on whether the function is defined with a SQL SECURITY characteristic of DEFINER or INVOKER. If the security characteristic is DEFINER, the function runs with the privileges of its creator. If the characteristic is INVOKER, the function runs with the privileges determined by the view's SQL SECURITY characteristic.

Example: A view might depend on a stored function, and that function might invoke other stored routines. For example, the following view invokes a stored function f():

```
CREATE VIEW v AS SELECT * FROM t WHERE t.id = f(t.name);
```

Suppose that f() contains a statement such as this:

```
IF name IS NULL then
  CALL p1();
ELSE
  CALL p2();
END IF;
```

The privileges required for executing statements within f() need to be checked when f() executes. This might mean that privileges are needed for p1() or p2(), depending on the execution path within f(). Those privileges must be checked at runtime, and the user who must possess the privileges is determined by the SQL SECURITY values of the view v and the function f().

The DEFINER and SQL SECURITY clauses for views are extensions to standard SQL. In standard SQL, views are handled using the rules for SQL SECURITY INVOKER.

If you invoke a view that was created before MySQL 5.1.2, it is treated as though it was created with a SQL SECURITY DEFINER clause and with a DEFINER value that is the same as your account. However, because the actual definer is unknown, MySQL issues a warning. To make the warning go away, it is sufficient to re-create the view so that the view definition includes a DEFINER clause.

The optional ALGORITHM clause is an extension to standard SQL. It affects how MariaDB processes the view. ALGORITHM takes three values: MERGE, TEMPTABLE, or UNDEFINED. The default algorithm is UNDEFINED if no ALGORITHM clause is present. See [View Algorithms](#) for more information.

Some views are updatable. That is, you can use them in statements such as UPDATE, DELETE, or INSERT to update the contents of the underlying table. For a view to be updatable, there must be a one-to-one relationship between the rows in the view and the rows in the underlying table. There are also certain other constructs that make a view non-updatable. See [Inserting and Updating with Views](#).

## WITH CHECK OPTION

The WITH CHECK OPTION clause can be given for an updatable view to prevent inserts or updates to rows except those for which the WHERE clause in the select\_statement is true.

In a WITH CHECK OPTION clause for an updatable view, the LOCAL and CASCDED keywords determine the scope of check testing when the view is defined in terms of another view. The LOCAL keyword restricts the CHECK OPTION only to the view being defined. CASCDED causes the checks for underlying views to be evaluated as well. When neither keyword is given, the default is CASCDED.

For more information about updatable views and the WITH CHECK OPTION clause, see [Inserting and Updating with Views](#).

## IF NOT EXISTS

MariaDB starting with 10.1.3

The

```
IF NOT EXISTS
clause was added in MariaDB 10.1.3
```

When the IF NOT EXISTS clause is used, MariaDB will return a warning instead of an error if the specified view already exists. Cannot be used together with the

```
OR REPLACE
clause.
```

## Atomic DDL

MariaDB starting with 10.6.1

MariaDB 10.6.1 supports Atomic DDL and  
CREATE VIEW  
is atomic.

## Examples

```
CREATE TABLE t (a INT, b INT) ENGINE = InnoDB;  
  
INSERT INTO t VALUES (1,1), (2,2), (3,3);  
  
CREATE VIEW v AS SELECT a, a*2 AS a2 FROM t;  
  
SELECT * FROM v;  
+-----+-----+  
| a    | a2   |  
+-----+-----+  
| 1    | 2    |  
| 2    | 4    |  
| 3    | 6    |  
+-----+-----+
```

OR REPLACE and IF NOT EXISTS:

```
CREATE VIEW v AS SELECT a, a*2 AS a2 FROM t;  
ERROR 1050 (42S01): Table 'v' already exists  
  
CREATE OR REPLACE VIEW v AS SELECT a, a*2 AS a2 FROM t;  
Query OK, 0 rows affected (0.04 sec)  
  
CREATE VIEW IF NOT EXISTS v AS SELECT a, a*2 AS a2 FROM t;  
Query OK, 0 rows affected, 1 warning (0.01 sec)  
  
SHOW WARNINGS;  
+-----+-----+  
| Level | Code | Message          |  
+-----+-----+  
| Note  | 1050 | Table 'v' already exists |  
+-----+-----+
```

## See Also

- [Identifier Names](#)
- [ALTER VIEW](#)
- [DROP VIEW](#)
- [SHOW CREATE VIEWS](#)
- [INFORMATION SCHEMA VIEWS Table](#)

### 1.1.3.1.18 Silent Column Changes

When a [CREATE TABLE](#) or [ALTER TABLE](#) command is issued, MariaDB will silently change a column specification in the following cases:

- [PRIMARY KEY](#) columns are always NOT NULL.
- Any trailing spaces from [SET](#) and [ENUM](#) values are discarded.
- [TIMESTAMP](#) columns are always NOT NULL, and display sizes are discarded
- A row-size limit of 65535 bytes applies
- If [strict SQL mode](#) is not enabled (it is enabled by default from [MariaDB 10.2](#)), a [VARCHAR](#) column longer than 65535 becomes [TEXT](#), and a [VARBINARY](#) columns longer than 65535 becomes a [BLOB](#). If strict mode is enabled the silent changes will not be made, and an error will occur.
- If a [USING](#) clause specifies an index that's not permitted by the storage engine, the engine will instead use another available index type that can be applied without affecting results.
- If the CHARACTER SET binary attribute is specified, the column is created as the matching binary data type. A [TEXT](#) becomes a [BLOB](#), [CHAR](#) a [BINARY](#) and [VARCHAR](#) a [VARBINARY](#). [ENUMs](#) and [SETs](#) are created as defined.

To ease imports from other RDBMSs, MariaDB will also silently map the following data types:

Other Vendor Type	MariaDB Type
BOOL	TINYINT

BOOLEAN	TINYINT
CHARACTER VARYING(M)	VARCHAR (M)
FIXED	DECIMAL
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB
LONG VARCHAR	MEDIUMTEXT
LONG	MEDIUMTEXT
MIDDLEINT	MEDIUMINT
NUMERIC	DECIMAL

Currently, all MySQL types are supported in MariaDB.

For type mapping between Cassandra and MariaDB, see [Cassandra storage engine](#).

## Example

Silent changes in action:

```
CREATE TABLE SilenceIsGolden
(
  f1 TEXT CHARACTER SET binary,
  f2 VARCHAR(15) CHARACTER SET binary,
  f3 CHAR CHARACTER SET binary,
  f4 ENUM('x','y','z') CHARACTER SET binary,
  f5 VARCHAR (65536),
  f6 VARBINARY (65536),
  f7 INT1
);
Query OK, 0 rows affected, 2 warnings (0.31 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1246 | Converting column 'f5' from VARCHAR to TEXT |
| Note  | 1246 | Converting column 'f6' from VARBINARY to BLOB |
+-----+-----+

DESCRIBE SilenceIsGolden;
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| f1    | blob          | YES  |     | NULL    |       |
| f2    | varbinary(15) | YES  |     | NULL    |       |
| f3    | binary(1)     | YES  |     | NULL    |       |
| f4    | enum('x','y','z') | YES  |     | NULL    |       |
| f5    | mediumtext    | YES  |     | NULL    |       |
| f6    | mediumblob   | YES  |     | NULL    |       |
| f7    | tinyint(4)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
```

### 1.1.3.1.19 Generated (Virtual and Persistent/Stored) Columns

#### Syntax

```
<type> [GENERATED ALWAYS] AS ( <expression> )
[VIRTUAL | PERSISTENT | STORED] [UNIQUE] [UNIQUE KEY] [COMMENT <text>]
```

## Contents

- 1. [Syntax](#)
- 2. [Description](#)
- 3. [Supported Features](#)
  - 1. [Storage Engine Support](#)
  - 2. [Data Type Support](#)
  - 3. [Index Support](#)
  - 4. [Statement Support](#)
  - 5. [Expression Support](#)
  - 6. [Making Stored Values Consistent](#)
  - 7. [MySQL Compatibility Support](#)
- 4. [Implementation Differences](#)
  - 1. [Implementation Differences Compared to Microsoft SQL Server](#)
- 5. [Development History](#)
- 6. [Examples](#)
- 7. [See Also](#)

MariaDB's generated columns syntax is designed to be similar to the syntax for [Microsoft SQL Server's computed columns](#) and [Oracle Database's virtual columns](#). In [MariaDB 10.2](#) and later, the syntax is also compatible with the syntax for [MySQL's generated columns](#).

## Description

A generated column is a column in a table that cannot explicitly be set to a specific value in a [DML query](#). Instead, its value is automatically generated based on an expression. This expression might generate the value based on the values of other columns in the table, or it might generate the value by calling [built-in functions](#) or [user-defined functions \(UDFs\)](#).

There are two types of generated columns:

- - PERSISTENT
    - (a.k.a.
    - STORED
  - ): This type's value is actually stored in the table.
- - VIRTUAL
    - : This type's value is not stored at all. Instead, the value is generated dynamically when the table is queried. This type is the default.

Generated columns are also sometimes called computed columns or virtual columns.

## Supported Features

### Storage Engine Support

- Generated columns can only be used with storage engines which support them. If you try to use a storage engine that does not support them, then you will see an error similar to the following:

```
ERROR 1910 (HY000): TokuDB storage engine does not support computed columns
```

- [InnoDB](#), [Aria](#), [MyISAM](#) and [CONNECT](#) support generated columns.
- A column in a [MERGE](#) table can be built on a
  - PERSISTENT
    - generated column.
  - However, a column in a MERGE table can **not** be defined as a
    - VIRTUAL
    - and
    - PERSISTENT
    - generated column.

### Data Type Support

- All data types are supported when defining generated columns.
- Using the

[ZEROFILL](#)

column option is supported when defining generated columns.

#### MariaDB starting with 10.2.6

In MariaDB 10.2.6 and later, the following statements apply to data types for generated columns:

- Using the

AUTO\_INCREMENT

column option is **not** supported when defining generated columns. Previously, it was supported, but this support was removed, because it would not work correctly. See [MDEV-11117](#).

## Index Support

- Using a generated column as a table's primary key is **not** supported. See [MDEV-5590](#) for more information. If you try to use one as a primary key, then you will see an error similar to the following:

ERROR 1903 (HY000): Primary key cannot be defined upon a computed column

- Using

PERSISTENT

generated columns as part of a [foreign key](#) is supported.

- Referencing

PERSISTENT

generated columns as part of a [foreign key](#) is also supported.

- However, using the

ON UPDATE CASCADE

,

ON UPDATE SET NULL

, or

ON DELETE SET NULL

clauses is **not** supported. If you try to use an unsupported clause, then you will see an error similar to the following:

ERROR 1905 (HY000): Cannot define foreign key with ON UPDATE SET NULL clause on a computed column

#### MariaDB starting with 10.2.3

In MariaDB 10.2.3 and later, the following statements apply to indexes for generated columns:

- Defining indexes on both

VIRTUAL

and

PERSISTENT

generated columns is supported.

- If an index is defined on a generated column, then the optimizer considers using it in the same way as indexes based on "real" columns.

#### MariaDB until 10.2.2

In MariaDB 10.2.2 and before, the following statements apply to indexes for generated columns:

- Defining indexes on

VIRTUAL

generated columns is **not** supported.

- Defining indexes on

PERSISTENT

generated columns is supported.

- If an index is defined on a generated column, then the optimizer considers using it in the same way as indexes based on "real" columns.

## Statement Support

- Generated columns are used in [DML queries](#) just as if they were "real" columns.

- However,

VIRTUAL

and

PERSISTENT

generated columns differ in how their data is stored.

- Values for

PERSISTENT

generated columns are generated whenever a [DML queries](#) inserts or updates the row with the special

DEFAULT

value. This generates the columns value, and it is stored in the table like the other "real" columns. This value can be read by other [DML queries](#) just like the other "real" columns.

▪ Values for

VIRTUAL

generated columns are not stored in the table. Instead, the value is generated dynamically whenever the column is queried. If other columns in a row are queried, but the

VIRTUAL

generated column is not one of the queried columns, then the column's value is not generated.

- The

[SELECT](#)

statement supports generated columns.

- Generated columns can be referenced in the

[INSERT](#)

,

[UPDATE](#)

, and

[DELETE](#)

statements.

- However,

VIRTUAL

or

PERSISTENT

generated columns cannot be explicitly set to any other values than

NULL

or

[DEFAULT](#)

. If a generated column is explicitly set to any other value, then the outcome depends on whether [strict mode](#) is enabled in

[sql\\_mode](#)

. If it is not enabled, then a warning will be raised and the default generated value will be used instead. If it is enabled, then an error will be raised instead.

- The

[CREATE TABLE](#)

statement has limited support for generated columns.

- It supports defining generated columns in a new table.
- It supports using generated columns to [partition tables](#).
- It does **not** support using the [versioning clauses](#) with generated columns.

- The

[ALTER TABLE](#)

statement has limited support for generated columns.

- It supports the

MODIFY

and

CHANGE

clauses for

PERSISTENT

generated columns.

- It does **not** support the

MODIFY

clause for

VIRTUAL

generated columns if

#### ALGORITHM

is not set to  
COPY  
. See [MDEV-15476](#) for more information.

- It does **not** support the  
CHANGE  
clause for  
VIRTUAL  
generated columns if

#### ALGORITHM

is not set to  
COPY  
. See [MDEV-17035](#) for more information.

- It does **not** support altering a table if

#### ALGORITHM

is not set to  
COPY  
if the table has a  
VIRTUAL  
generated column that is indexed. See [MDEV-14046](#) for more information.

- It does **not** support adding a  
VIRTUAL  
generated column with the  
ADD  
clause if the same statement is also adding other columns if

#### ALGORITHM

is not set to  
COPY  
. See [MDEV-17468](#) for more information.

- It also does **not** support altering an existing column into a  
VIRTUAL  
generated column.
- It supports using generated columns to [partition tables](#) .
- It does **not** support using the [versioning clauses](#) with generated columns.

- The

#### SHOW CREATE TABLE

statement supports generated columns.

- The

#### DESCRIBE

statement can be used to check whether a table has generated columns.

- You can tell which columns are generated by looking for the ones where the  
Extra  
column is set to either  
VIRTUAL  
or  
PERSISTENT  
. For example:

```
DESCRIBE table1;
+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+
| a     | int(11)   | NO   |     | NULL    |            |
| b     | varchar(32)| YES  |     | NULL    |            |
| c     | int(11)   | YES  |     | NULL    | VIRTUAL    |
| d     | varchar(5) | YES  |     | NULL    | PERSISTENT |
+-----+-----+-----+-----+
```

- Generated columns can be properly referenced in the

NEW  
and  
OLD  
rows in triggers .

- Stored procedures support generated columns.
- The

HANDLER

statement supports generated columns.

## Expression Support

- Most legal, deterministic expressions which can be calculated are supported in expressions for generated columns.
- Most [built-in functions](#) are supported in expressions for generated columns.
  - However, some [built-in functions](#) can't be supported for technical reasons. For example, If you try to use an unsupported function in an expression, an error is generated similar to the following:

```
ERROR 1901 (HY000): Function or expression 'dayname()' cannot be used in the GENERATED ALWAYS AS clause of `v`
```

- [Subqueries](#) are **not** supported in expressions for generated columns because the underlying data can change.
- Using anything that depends on data outside the row is **not** supported in expressions for generated columns.
- [Stored functions](#) are **not** supported in expressions for generated columns. See [MDEV-17587](#) for more information.

### MariaDB starting with 10.2.1

In MariaDB 10.2.1 and later, the following statements apply to expressions for generated columns:

- Non-deterministic [built-in functions](#) are supported in expressions for not indexed  
VIRTUAL  
generated columns.
- Non-deterministic [built-in functions](#) are **not** supported in expressions for  
PERSISTENT  
or indexed  
VIRTUAL  
generated columns.
- [User-defined functions \(UDFs\)](#) are supported in expressions for generated columns.
  - However, MariaDB can't check whether a UDF is deterministic, so it is up to the user to be sure that they do not use non-deterministic UDFs with  
VIRTUAL  
generated columns.
- Defining a generated column based on other generated columns defined before it in the table definition is supported. For example:

```
CREATE TABLE t1 (a int as (1), b int as (a));
```

- However, defining a generated column based on other generated columns defined after in the table definition is **not** supported in expressions for generation columns because generated columns are calculated in the order they are defined.
- Using an expression that exceeds 255 characters in length is supported in expressions for generated columns. The new limit for the entire table definition, including all expressions for generated columns, is 65,535 bytes.
- Using constant expressions is supported in expressions for generated columns. For example:

```
CREATE TABLE t1 (a int as (1));
```

### MariaDB until 10.2.0

In MariaDB 10.2.0 and before, the following statements apply to expressions for generated columns:

- Non-deterministic [built-in functions](#) are **not** supported in expressions for generated columns.
- [User-defined functions \(UDFs\)](#) are **not** supported in expressions for generated columns.
- Defining a generated column based on other generated columns defined in the table is **not** supported. Otherwise, it would generate errors like this:

```
ERROR 1900 (HY000): A computed column cannot be based on a computed column
```

- Using an expression that exceeds 255 characters in length is **not** supported in expressions for generated columns.
- Using constant expressions is **not** supported in expressions for generated columns. Otherwise, it would generate errors like this:

```
ERROR 1908 (HY000): Constant expression in computed column function is not allowed
```

## Making Stored Values Consistent

When a generated column is

PERSISTENT

or indexed, the value of the expression needs to be consistent regardless of the

[SQL Mode](#)

flags in the current session. If it is not, then the table will be seen as corrupted when the value that should actually be returned by the computed expression and the value that was previously stored and/or indexed using a different

[sql\\_mode](#)

setting disagree.

There are currently two affected classes of inconsistencies: character padding and unsigned subtraction:

- For a

VARCHAR

or

TEXT

generated column the length of the value returned can vary depending on the PAD\_CHAR\_TO\_FULL\_LENGTH

[sql\\_mode](#)

flag. To make the value consistent, create the generated column using an RTRIM() or RPAD() function. Alternately, create the generated column as a

CHAR

column so that its data is always fully padded.

- If a

SIGNED

generated column is based on the subtraction of an

UNSIGNED

value, the resulting value can vary depending on how large the value is and the NO\_UNSIGNED\_SUBTRACTION

[sql\\_mode](#)

flag. To make the value consistent, use

[CAST\(\)](#)

to ensure that each

UNSIGNED

operand is

SIGNED

before the subtraction.

MariaDB starting with [10.5](#)

Beginning in [MariaDB 10.5](#), there is a fatal error generated when trying to create a generated column whose value can change depending on the

[SQL Mode](#)

when its data is

PERSISTENT

or indexed.

For an existing generated column that has a potentially inconsistent value, a warning about a bad expression is generated the first time it is used (if warnings are enabled).

Beginning in [MariaDB 10.4.8](#), [MariaDB 10.3.18](#), and [MariaDB 10.2.27](#) a potentially inconsistent generated column outputs a warning when created or first used (without restricting their creation).

Here is an example of two tables that would be rejected in [MariaDB 10.5](#) and warned about in the other listed versions:

```

CREATE TABLE bad_pad (
    txt CHAR(5),
    -- CHAR -> VARCHAR or CHAR -> TEXT can't be persistent or indexed:
    vtxt VARCHAR(5) AS (txt) PERSISTENT
);

CREATE TABLE bad_sub (
    num1 BIGINT UNSIGNED,
    num2 BIGINT UNSIGNED,
    -- The resulting value can vary for some large values
    vnum BIGINT AS (num1 - num2) VIRTUAL,
    KEY(vnum)
);

```

The warnings for the above tables look like this:

```

Warning (Code 1901): Function or expression ``txt`` cannot be used in the GENERATED ALWAYS AS clause of `vtxt`
Warning (Code 1105): Expression depends on the @@sql_mode value PAD_CHAR_TO_FULL_LENGTH

Warning (Code 1901): Function or expression ``num1`` - ``num2`` cannot be used in the GENERATED ALWAYS AS clause of `vnum`
Warning (Code 1105): Expression depends on the @@sql_mode value NO_UNSIGNED_SUBTRACTION

```

To work around the issue, force the padding or type to make the generated column's expression return a consistent value. For example:

```

CREATE TABLE good_pad (
    txt CHAR(5),
    -- Using RTRIM() or RPAD() makes the value consistent:
    vtxt VARCHAR(5) AS (RTRIM(txt)) PERSISTENT,
    -- When not persistent or indexed, it is OK for the value to vary by mode:
    vtxt2 VARCHAR(5) AS (txt) VIRTUAL,
    -- CHAR -> CHAR is always OK:
    txt2 CHAR(5) AS (txt) PERSISTENT
);

CREATE TABLE good_sub (
    num1 BIGINT UNSIGNED,
    num2 BIGINT UNSIGNED,
    -- The indexed value will always be consistent in this expression:
    vnum BIGINT AS (CAST(num1 AS SIGNED) - CAST(num2 AS SIGNED)) VIRTUAL,
    KEY(vnum)
);

```

## MySQL Compatibility Support

MariaDB starting with 10.2.1

In MariaDB 10.2.1 and later, the following statements apply to MySQL compatibility for generated columns:

- The **STORED**  
keyword is supported as an alias for the  
**PERSISTENT**  
keyword.
- Tables created with MySQL 5.7 or later that contain [MySQL's generated columns](#) can be imported into MariaDB without a dump and restore.

MariaDB until 10.2.0

In MariaDB 10.2.0 and before, the following statements apply to MySQL compatibility for generated columns:

- The **STORED**  
keyword is **not** supported as an alias for the  
**PERSISTENT**  
keyword.
- Tables created with MySQL 5.7 or later that contain [MySQL's generated columns](#) can **not** be imported into MariaDB without a dump and restore.

## Implementation Differences

Generated columns are subject to various constraints in other DBMSs that are not present in MariaDB's implementation. Generated columns may also be called computed columns or virtual columns in different implementations. The various details for a specific implementation can be found in the

documentation for each specific DBMS.

## Implementation Differences Compared to Microsoft SQL Server

MariaDB's generated columns implementation does not enforce the following restrictions that are present in Microsoft SQL Server's computed columns implementation:

- MariaDB allows [server variables](#) in generated column expressions, including those that change dynamically, such as

`warning_count`

- MariaDB allows the

`CONVERT_TZ()`

function to be called with a named [time zone](#) as an argument, even though time zone names and time offsets are configurable.

- MariaDB allows the

`CAST()`

function to be used with non-unicode [character sets](#), even though character sets are configurable and differ between binaries/versions.

- MariaDB allows

`FLOAT`

expressions to be used in generated columns. Microsoft SQL Server considers these expressions to be "imprecise" due to potential cross-platform differences in floating-point implementations and precision.

- Microsoft SQL Server requires the

`ARITHABORT`

mode to be set, so that division by zero returns an error, and not a NULL.

- Microsoft SQL Server requires

`QUOTED_IDENTIFIER`

to be set in

`sql_mode`

. In MariaDB, if data is inserted without

`ANSI_QUOTES`

set in

`sql_mode`

, then it will be processed and stored differently in a generated column that contains quoted identifiers.

- In [MariaDB 10.2.0](#) and before, it does not allow [user-defined functions \(UDFs\)](#) to be used in expressions for generated columns.

Microsoft SQL Server enforces the above restrictions by doing one of the following things:

- Refusing to create computed columns.
- Refusing to allow updates to a table containing them.
- Refusing to use an index over such a column if it can not be guaranteed that the expression is fully deterministic.

In MariaDB, as long as the

`sql_mode`

, language, and other settings that were in effect during the CREATE TABLE remain unchanged, the generated column expression will always be evaluated the same. If any of these things change, then please be aware that the generated column expression might not be evaluated the same way as it previously was.

In [MariaDB 5.2](#) , you will get a warning if you try to update a virtual column. In [MariaDB 5.3](#) and later, this warning will be converted to an error if [strict mode](#) is enabled in

`sql_mode`

## Development History

Generated columns was originally developed by Andrey Zhakov. It was then modified by Sanja Byelkin and Igor Babaev at Monty Program for inclusion in MariaDB. Monty did the work on [MariaDB 10.2](#) to lift a some of the old limitations.

# Examples

Here is an example table that uses both

VIRTUAL  
and  
PERSISTENT  
virtual columns:

```
USE TEST;

CREATE TABLE table1 (
    a INT NOT NULL,
    b VARCHAR(32),
    c INT AS (a mod 10) VIRTUAL,
    d VARCHAR(5) AS (left(b,5)) PERSISTENT);
```

If you describe the table, you can easily see which columns are virtual by looking in the "Extra" column:

```
DESCRIBE table1;
+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra       |
+-----+-----+-----+-----+
| a     | int(11)   | NO   |     | NULL    |             |
| b     | varchar(32) | YES  |     | NULL    |             |
| c     | int(11)   | YES  |     | NULL    | VIRTUAL     |
| d     | varchar(5) | YES  |     | NULL    | PERSISTENT |
+-----+-----+-----+-----+
```

To find out what function(s) generate the value of the virtual column you can use

```
SHOW CREATE TABLE
:
```

```
SHOW CREATE TABLE table1;

| table1 | CREATE TABLE `table1` (
  `a` int(11) NOT NULL,
  `b` varchar(32) DEFAULT NULL,
  `c` int(11) AS (a mod 10) VIRTUAL,
  `d` varchar(5) AS (left(b,5)) PERSISTENT
) ENGINE=MyISAM DEFAULT CHARSET=latin1 |
```

If you try to insert non-default values into a virtual column, you will receive a warning and what you tried to insert will be ignored and the derived value inserted instead:

```
WARNINGS;
Show warnings enabled.

INSERT INTO table1 VALUES (1, 'some text', default, default);
Query OK, 1 row affected (0.00 sec)

INSERT INTO table1 VALUES (2, 'more text', 5, default);
Query OK, 1 row affected, 1 warning (0.00 sec)

Warning (Code 1645): The value specified for computed column 'c' in table 'table1' has been ignored.

INSERT INTO table1 VALUES (123, 'even more text', default, 'something');
Query OK, 1 row affected, 2 warnings (0.00 sec)

Warning (Code 1645): The value specified for computed column 'd' in table 'table1' has been ignored.
Warning (Code 1265): Data truncated for column 'd' at row 1

SELECT * FROM table1;
+-----+-----+-----+
| a   | b       | c     | d     |
+-----+-----+-----+
| 1   | some text | 1   | some  |
| 2   | more text  | 2   | more  |
| 123 | even more text | 3   | even  |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

If the

ZEROFILL

clause is specified, it should be placed directly after the type definition, before the  
AS (<expression>  
:  
:

```
CREATE TABLE table2 (a INT, b INT ZEROFILL AS (a*2) VIRTUAL);
INSERT INTO table2 (a) VALUES (1);

SELECT * FROM table2;
+-----+
| a    | b   |
+-----+
| 1   | 000000002 |
+-----+
1 row in set (0.00 sec)
```

You can also use virtual columns to implement a "poor man's partial index". See example at the end of [Unique Index](#).

## See Also

- [Putting Virtual Columns to good use](#) on the mariadb.com blog.

### 1.1.3.1.20 Invisible Columns

MariaDB starting with [10.3.3](#)

Invisible columns (sometimes also called hidden columns) first appeared in [MariaDB 10.3.3](#).

Columns can be given an

INVISIBLE

attribute in a [CREATE TABLE](#) or [ALTER TABLE](#) statement. These columns will then not be listed in the results of a [SELECT \\*](#) statement, nor do they need to be assigned a value in an [INSERT](#) statement, unless INSERT explicitly mentions them by name.

Since

[SELECT \\*](#)

does not return the invisible columns, new tables or views created in this manner will have no trace of the invisible columns. If specifically referenced in the [SELECT](#) statement, the columns will be brought into the view/new table, but the INVISIBLE attribute will not.

Invisible columns can be declared as

[NOT NULL](#)

, but then require a

[DEFAULT](#)

value.

It is not possible for all columns in a table to be invisible.

## Examples

```

CREATE TABLE t (x INT INVISIBLE);
ERROR 1113 (42000): A table must have at least 1 column

CREATE TABLE t (x INT, y INT INVISIBLE, z INT INVISIBLE NOT NULL);
ERROR 4106 (HY000): Invisible column `z` must have a default value

CREATE TABLE t (x INT, y INT INVISIBLE, z INT INVISIBLE NOT NULL DEFAULT 4);

INSERT INTO t VALUES (1),(2);

INSERT INTO t (x,y) VALUES (3,33);

SELECT * FROM t;
+---+
| x |
+---+
| 1 |
| 2 |
| 3 |
+---+

SELECT x,y,z FROM t;
+---+---+---+
| x | y | z |
+---+---+---+
| 1 | NULL | 4 |
| 2 | NULL | 4 |
| 3 | 33 | 4 |
+---+---+---+

DESC t;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| x | int(11) | YES | | NULL | |
| y | int(11) | YES | | NULL | INVISIBLE |
| z | int(11) | NO | | 4 | INVISIBLE |
+-----+-----+-----+-----+-----+


ALTER TABLE t MODIFY x INT INVISIBLE, MODIFY y INT, MODIFY z INT NOT NULL DEFAULT 4;

DESC t;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| x | int(11) | YES | | NULL | INVISIBLE |
| y | int(11) | YES | | NULL | |
| z | int(11) | NO | | 4 | |
+-----+-----+-----+-----+-----+

```

Creating a view from a table with hidden columns:

```

CREATE VIEW v1 AS SELECT * FROM t;

DESC v1;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| y | int(11) | YES | | NULL | |
| z | int(11) | NO | | 4 | |
+-----+-----+-----+-----+-----+


CREATE VIEW v2 AS SELECT x,y,z FROM t;

DESC v2;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| x | int(11) | YES | | NULL | |
| y | int(11) | YES | | NULL | |
| z | int(11) | NO | | 4 | |
+-----+-----+-----+-----+-----+

```

## 1.1.3.2 ALTER

## 1.1.3.2.1 ALTER TABLE

## 1.1.3.2.2 ALTER DATABASE

## 1.1.3.2.3 ALTER EVENT

## 1.1.3.2.4 ALTER FUNCTION

## 1.1.3.2.5 ALTER LOGFILE GROUP

## 1.1.3.2.6 ALTER PROCEDURE

## 1.1.3.2.7 ALTER SEQUENCE

## 1.1.3.2.8 ALTER SERVER

## 1.1.3.2.9 ALTER TABLESPACE

## 1.1.3.2.10 ALTER USER

## 1.1.3.2.11 ALTER VIEW

## 1.1.3.3 DROP

### 1.1.3.3.1 DROP DATABASE

## Syntax

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

## Contents

- 1. [Syntax](#)
- 2. [Description](#)
  - 1. [IF EXISTS](#)
  - 2. [Atomic DDL](#)
- 3. [Examples](#)
- 4. [See Also](#)

## Description

`DROP DATABASE`

drops all tables in the database and deletes the database. Be very careful with this statement! To use `DROP DATABASE`, you need the [DROP privilege](#) on the database.

`DROP SCHEMA`

is a synonym for

`DROP DATABASE`

**Important:** When a database is dropped, user privileges on the database are not automatically dropped. See [GRANT](#).

### IF EXISTS

Use

`IF EXISTS`

to prevent an error from occurring for databases that do not exist. A

`NOTE`

is generated for each non-existent database when using  
IF EXISTS  
. See [SHOW WARNINGS](#).

## Atomic DDL

MariaDB starting with [10.6.1](#)

MariaDB [10.6.1](#) supports [Atomic DDL](#).

DROP DATABASE  
is implemented as

```
loop over all tables
DROP TABLE table
```

Each individual [DROP TABLE](#) is atomic while

DROP DATABASE  
as a whole is crash-safe.

## Examples

```
DROP DATABASE bufg;
Query OK, 0 rows affected (0.39 sec)

DROP DATABASE bufg;
ERROR 1008 (HY000): Can't drop database 'bufg'; database doesn't exist

\W
Show warnings enabled.

DROP DATABASE IF EXISTS bufg;
Query OK, 0 rows affected, 1 warning (0.00 sec)
Note (Code 1008): Can't drop database 'bufg'; database doesn't exist
```

## See Also

- [CREATE DATABASE](#)
- [ALTER DATABASE](#)
- [SHOW DATABASES](#)
- [Information Schema SCHEMATA Table](#)
- [SHOW CREATE DATABASE](#)

### 1.1.3.3.2 DROP EVENT

#### Syntax

```
DROP EVENT [IF EXISTS] event_name
```

#### Description

This statement drops the [event](#) named

[event\\_name](#)  
. The event immediately ceases being active, and is deleted completely from the server.

If the event does not exist, the error

ERROR 1517 (HY000): Unknown event 'event\_name'  
results. You can override this and cause the statement to generate a  
NOTE  
for non-existent events instead by using  
IF EXISTS  
. See

[SHOW WARNINGS](#)

This statement requires the

EVENT

privilege. In MySQL 5.1.11 and earlier, an event could be dropped only by its definer, or by a user having the

SUPER

privilege.

## Examples

```
DROP EVENT myevent3;
```

Using the IF EXISTS clause:

```
DROP EVENT IF EXISTS myevent3;
Query OK, 0 rows affected, 1 warning (0.01 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1305 | Event myevent3 does not exist |
+-----+-----+
```

## See also

- [Events Overview](#)
- [CREATE EVENT](#)
- [SHOW CREATE EVENT](#)
- [ALTER EVENT](#)

### 1.1.3.3.3 DROP FUNCTION

## Syntax

```
DROP FUNCTION [IF EXISTS] f_name
```

### Contents

1. [Syntax](#)
2. [Description](#)
  - 1. [IF EXISTS](#)
3. [Examples](#)
4. [See Also](#)

## Description

The DROP FUNCTION statement is used to drop a [stored function](#) or a user-defined function (UDF). That is, the specified routine is removed from the server, along with all privileges specific to the function. You must have the

ALTER ROUTINE  
privilege for the routine in order to drop it. If the [automatic\\_sp\\_privileges](#) server system variable is set, both the  
ALTER ROUTINE  
and  
EXECUTE  
privileges are granted automatically to the routine creator - see [Stored Routine Privileges](#).

### IF EXISTS

The

IF EXISTS  
clause is a MySQL/MariaDB extension. It prevents an error from occurring if the function does not exist. A  
NOTE  
is produced that can be viewed with [SHOW WARNINGS](#).

For dropping a [user-defined functions](#) (UDF), see [DROP FUNCTION UDF](#).

## Examples

```
DROP FUNCTION hello;
Query OK, 0 rows affected (0.042 sec)

DROP FUNCTION hello;
ERROR 1305 (42000): FUNCTION test.hello does not exist

DROP FUNCTION IF EXISTS hello;
Query OK, 0 rows affected, 1 warning (0.000 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message           |
+-----+-----+
| Note  | 1305 | FUNCTION test.hello does not exist |
+-----+-----+
```

## See Also

- [DROP PROCEDURE](#)
- [Stored Function Overview](#)
- [CREATE FUNCTION](#)
- [CREATE FUNCTION UDF](#)
- [ALTER FUNCTION](#)
- [SHOW CREATE FUNCTION](#)
- [SHOW FUNCTION STATUS](#)
- [Stored Routine Privileges](#)
- [INFORMATION\\_SCHEMA ROUTINES Table](#)

### 1.1.3.3.4 DROP FUNCTION UDF

## Syntax

```
DROP FUNCTION [IF EXISTS] function_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [Upgrading a UDF](#)
3. [Examples](#)

## Description

This statement drops the user-defined function (UDF) named  
function\_name

To drop a function, you must have the

```
DELETE privilege
```

for the mysql database. This is because

```
DROP FUNCTION
```

removes the row from the `mysql.func` system table that records the function's name, type and shared library name.

For dropping a stored function, see [DROP FUNCTION](#).

## Upgrading a UDF

To upgrade the UDF's shared library, first run a [DROP FUNCTION](#) statement, then upgrade the shared library and finally run the [CREATE FUNCTION](#) statement. If you upgrade without following this process, you may crash the server.

## Examples

```
DROP FUNCTION jsoncontains_path;
```

IF EXISTS:

```
DROP FUNCTION jsoncontains_path;
ERROR 1305 (42000): FUNCTION test.jsoncontains_path does not exist

DROP FUNCTION IF EXISTS jsoncontains_path;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1305 | FUNCTION test.jsoncontains_path does not exist |
+-----+-----+
```

## 1.1.3.3.5 DROP INDEX

### Syntax

```
DROP INDEX [IF EXISTS] index_name ON tbl_name
    [WAIT n | NOWAIT]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Privileges](#)
4. [Online DDL](#)
5. [DROP INDEX IF EXISTS ...](#)
6. [WAIT/NOWAIT](#)
7. [Progress Reporting](#)
8. [See Also](#)

### Description

```
DROP INDEX
drops the index named
index_name
from the table
tbl_name
. This statement is mapped to an
ALTER TABLE
statement to drop the index.
```

If another connection is using the table, a [metadata lock](#) is active, and this statement will wait until the lock is released. This is also true for non-transactional tables.

See

[ALTER TABLE](#)

Another shortcut,

[CREATE INDEX](#)

, allows the creation of an index.

To remove the primary key,

`PRIMARY`  
must be specified as index\_name. Note that [the quotes](#) are necessary, because  
PRIMARY  
is a keyword.

### Privileges

Executing the

[DROP INDEX](#)

statement requires the

#### INDEX

privilege for the table or the database.

## Online DDL

Online DDL is used by default with InnoDB, when the drop index operation supports it.

See [InnoDB Online DDL Overview](#) for more information on online DDL with InnoDB .

### DROP INDEX IF EXISTS ...

If the

IF EXISTS

clause is used, then MariaDB will return a warning instead of an error if the index does not exist.

### WAIT/NOWAIT

MariaDB starting with 10.3.0

Set the lock wait timeout. See [WAIT](#) and [NOWAIT](#) .

## Progress Reporting

MariaDB provides progress reporting for

DROP INDEX

statement for clients that support the new progress reporting protocol. For example, if you were using the

mysql

client, then the progress report might look like this::

## See Also

- [Getting Started with Indexes](#)
- [CREATE INDEX](#)
- [ALTER TABLE](#)

### 1.1.3.3.6 DROP LOGFILE GROUP

The

DROP LOGFILE GROUP

statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. See [MDEV-19295](#) for more information.

### 1.1.3.3.7 DROP PACKAGE

MariaDB starting with 10.3.5

Oracle-style packages were introduced in [MariaDB 10.3.5](#) .

## Syntax

```
DROP PACKAGE [IF EXISTS] [ db_name . ] package_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

## Description

The

```
DROP PACKAGE  
statement can be used when Oracle SQL_MODE is set.
```

The

```
DROP PACKAGE  
statement drops a stored package entirely.
```

- Drops the package specification (earlier created using the [CREATE PACKAGE](#) statement).
- Drops the package implementation, if the implementation was already created using the [CREATE PACKAGE BODY](#) statement.

## See Also

- [SHOW CREATE PACKAGE](#)
- [CREATE PACKAGE](#)
- [CREATE PACKAGE BODY](#)
- [DROP PACKAGE BODY](#)
- [Oracle SQL\\_MODE](#)

## 1.1.3.3.8 DROP PACKAGE BODY

MariaDB starting with [10.3.5](#)

Oracle-style packages were introduced in [MariaDB 10.3.5](#).

## Syntax

```
DROP PACKAGE BODY [IF EXISTS] [ db_name . ] package_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [See also](#)

## Description

The

```
DROP PACKAGE BODY  
statement can be used when Oracle SQL_MODE is set.
```

The

```
DROP PACKAGE BODY  
statement drops the package body (i.e the implementation), previously created using the CREATE PACKAGE BODY statement.
```

Note,

```
DROP PACKAGE BODY  
drops only the package implementation, but does not drop the package specification. Use DROP PACKAGE to drop the package entirely (i.e. both implementation and specification).
```

## See also

- [CREATE PACKAGE](#)
- [SHOW CREATE PACKAGE](#)
- [DROP PACKAGE](#)
- [CREATE PACKAGE BODY](#)
- [SHOW CREATE PACKAGE BODY](#)
- [Oracle SQL\\_MODE](#)

## 1.1.3.3.9 DROP PROCEDURE

# Syntax

```
DROP PROCEDURE [IF EXISTS] sp_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

This statement is used to drop a [stored procedure](#). That is, the specified routine is removed from the server along with all privileges specific to the [procedure](#). You must have the

ALTER ROUTINE  
privilege for the routine. If the

`automatic_sp_privileges`

server system variable is set, that privilege and  
`EXECUTE`  
are granted automatically to the routine creator - see [Stored Routine Privileges](#).

The

`IF EXISTS`  
clause is a MySQL/MariaDB extension. It prevents an error from occurring if the procedure or function does not exist. A  
`NOTE`  
is produced that can be viewed with

`SHOW WARNINGS`

While this statement takes effect immediately, threads which are executing a procedure can continue execution.

## Examples

```
DROP PROCEDURE simpleproc;
```

IF EXISTS:

```
DROP PROCEDURE simpleproc;
ERROR 1305 (42000): PROCEDURE test.simpleproc does not exist

DROP PROCEDURE IF EXISTS simpleproc;
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Note  | 1305 | PROCEDURE test.simpleproc does not exist |
+-----+-----+
```

## See Also

- [DROP FUNCTION](#)
- [Stored Procedure Overview](#)
- [CREATE PROCEDURE](#)
- [ALTER PROCEDURE](#)
- [SHOW CREATE PROCEDURE](#)
- [SHOW PROCEDURE STATUS](#)
- [Information Schema ROUTINES Table](#)

## 1.1.3.3.10 DROP ROLE

## 1.1.3.3.11 DROP SEQUENCE

MariaDB starting with [10.3](#)

DROP SEQUENCE was introduced in [MariaDB 10.3](#).

### Syntax

```
DROP [TEMPORARY] SEQUENCE [IF EXISTS] /*COMMENT TO SAVE*/
sequence_name [, sequence_name] ...
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Notes](#)
4. [See Also](#)

### Description

`DROP SEQUENCE`

removes one or more sequences created with [CREATE SEQUENCE](#). You must have the

`DROP`

privilege for each sequence. MariaDB returns an error indicating by name which non-existing tables it was unable to drop, but it also drops all of the tables in the list that do exist.

Important: When a table is dropped, user privileges on the table are not automatically dropped. See [GRANT](#).

If another connection is using the sequence, a metadata lock is active, and this statement will wait until the lock is released. This is also true for non-transactional tables.

For each referenced sequence, `DROP SEQUENCE` drops a temporary sequence with that name, if it exists. If it does not exist, and the

`TEMPORARY`

keyword is not used, it drops a non-temporary sequence with the same name, if it exists. The

`TEMPORARY`

keyword ensures that a non-temporary sequence will not accidentally be dropped.

Use

`IF EXISTS`

to prevent an error from occurring for sequences that do not exist. A NOTE is generated for each non-existent sequence when using

`IF EXISTS`

. See [SHOW WARNINGS](#).

`DROP SEQUENCE` requires the [DROP privilege](#).

### Notes

`DROP SEQUENCE` only removes sequences, not tables. However, [DROP TABLE](#) can remove both sequences and tables.

### See Also

- [Sequence Overview](#)
- [CREATE SEQUENCE](#)
- [ALTER SEQUENCE](#)
- [DROP TABLE](#)

## 1.1.3.3.12 DROP SERVER

### Syntax

```
DROP SERVER [ IF EXISTS ] server_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
  - 1. [IF EXISTS](#)
3. [Examples](#)
4. [See Also](#)

## Description

Drops the server definition for the server named `server_name`. The corresponding row within the [mysql.servers table](#) will be deleted. This statement requires the [SUPER](#) privilege or, from MariaDB 10.5.2, the [FEDERATED ADMIN](#) privilege.

Dropping a server for a table does not affect any [FederatedX](#), [FEDERATED](#), [Connect](#) or [Spider](#) tables that used this connection information when they were created.

`DROP SERVER` is not written to the [binary log](#), irrespective of the [binary log format](#) being used. From MariaDB 10.1.13, Galera replicates the [CREATE SERVER](#), [ALTER SERVER](#) and `DROP SERVER` statements.

### IF EXISTS

If the IF EXISTS clause is used, MariaDB will not return an error if the server does not exist. Unlike all other statements, `DROP SERVER` IF EXISTS does not issue a note if the server does not exist. See [MDEV-9400](#).

## Examples

```
DROP SERVER s;
```

IF EXISTS:

```
DROP SERVER s;
ERROR 1477 (HY000): The foreign server name you are trying to reference
does not exist. Data source error: s
```

```
DROP SERVER IF EXISTS s;
Query OK, 0 rows affected (0.00 sec)
```

## See Also

- [CREATE SERVER](#)
- [ALTER SERVER](#)
- [Spider Storage Engine](#)
- [FederatedX Storage Engine](#)
- [Connect Storage Engine](#)

### 1.1.3.3.13 DROP TABLE

### 1.1.3.3.14 DROP TABLESPACE

The

```
DROP TABLESPACE
```

statement is not supported by MariaDB. It was originally inherited from MySQL NDB Cluster. In MySQL 5.7 and later, the statement is also supported for InnoDB. However, MariaDB has chosen not to include that specific feature. See [MDEV-19294](#) for more information.

### 1.1.3.3.15 DROP TRIGGER

## Syntax

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name
```

## Contents

1. [Syntax](#)
2. [Description](#)
  - 1. [Atomic DDL](#)
3. [Examples](#)
4. [See Also](#)

## Description

This statement drops a [trigger](#). The schema (database) name is optional. If the schema is omitted, the trigger is dropped from the default schema. Its use requires the

TRIGGER  
privilege for the table associated with the trigger.

Use

IF EXISTS  
to prevent an error from occurring for a trigger that does not exist. A  
NOTE  
is generated for a non-existent trigger when using  
IF EXISTS  
. See [SHOW WARNINGS](#).

**Note:** Triggers for a table are also dropped if you drop the table.

## Atomic DDL

MariaDB starting with 10.6.1

MariaDB 10.6.1 supports [Atomic DDL](#) and

DROP TRIGGER  
is atomic.

## Examples

```
DROP TRIGGER test.example_trigger;
```

Using the IF EXISTS clause:

```
DROP TRIGGER IF EXISTS test.example_trigger;  
Query OK, 0 rows affected, 1 warning (0.01 sec)  
  
SHOW WARNINGS;  
+-----+-----+  
| Level | Code | Message           |  
+-----+-----+  
| Note  | 1360 | Trigger does not exist |  
+-----+-----+
```

## See Also

- [Trigger Overview](#)
- [CREATE TRIGGER](#)
- [Information Schema TRIGGERS Table](#)
- [SHOW TRIGGERS](#)
- [SHOW CREATE TRIGGER](#)
- [Trigger Limitations](#)

### 1.1.3.3.16 DROP USER

### 1.1.3.3.17 DROP VIEW

## Syntax

```
DROP VIEW [IF EXISTS]
view_name [, view_name] ...
[RESTRICT | CASCADE]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Atomic DDL](#)
4. [Examples](#)
5. [See Also](#)

## Description

DROP VIEW

removes one or more [views](#). You must have the DROP privilege for each view. If any of the views named in the argument list do not exist, MariaDB returns an error indicating by name which non-existing views it was unable to drop, but it also drops all of the views in the list that do exist.

The

IF EXISTS

clause prevents an error from occurring for views that don't exist. When this clause is given, a

NOTE

is generated for each non-existent view. See [SHOW WARNINGS](#).

RESTRICT

and

CASCADE

, if given, are parsed and ignored.

It is possible to specify view names as

db\_name

.

view\_name

. This is useful to delete views from multiple databases with one statement. See [Identifier Qualifiers](#) for details.

The [DROP privilege](#) is required to use

DROP TABLE

on non-temporary tables. For temporary tables, no privilege is required, because such tables are only visible for the current session.

If a view references another view, it will be possible to drop the referenced view. However, the other view will reference a view which does not exist any more. Thus, querying it will produce an error similar to the following:

```
ERROR 1356 (HY000): View 'db_name.view_name' references invalid table(s) or
column(s) or function(s) or definer/invoker of view lack rights to use them
```

This problem is reported in the output of [CHECK TABLE](#).

Note that it is not necessary to use

DROP VIEW

to replace an existing view, because

[CREATE VIEW](#)

has an

OR REPLACE

clause.

## Atomic DDL

MariaDB starting with [10.6.1](#)

[MariaDB 10.6.1](#) supports [Atomic DDL](#) and

DROP VIEW

for a singular view is atomic. Dropping multiple views is crash-safe.

## Examples

```
DROP VIEW v,v2;
```

Given views

v  
and  
v2  
, but no view  
v3

```
DROP VIEW v,v2,v3;  
ERROR 1051 (42S02): Unknown table 'v3'
```

```
DROP VIEW IF EXISTS v,v2,v3;  
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

```
SHOW WARNINGS;  
+-----+-----+  
| Level | Code | Message |  
+-----+-----+  
| Note | 1051 | Unknown table 'test.v3' |  
+-----+-----+
```

## See Also

- [CREATE VIEW](#)
- [ALTER VIEW](#)
- [SHOW CREATE VIEWS](#)
- [INFORMATION SCHEMA VIEWS Table](#)

### 1.1.3.4

### 1.1.3.5 CONSTRAINT

MariaDB supports the implementation of constraints at the table-level using either [CREATE TABLE](#) or [ALTER TABLE](#) statements. A table constraint restricts the data you can add to the table. If you attempt to insert invalid data on a column, MariaDB throws an error.

## Syntax

```
[CONSTRAINT [symbol]] constraint_expression  
  
constraint_expression:  
| PRIMARY KEY [index_type] (index_col_name, ...) [index_option] ...  
| FOREIGN KEY [index_name] (index_col_name, ...) REFERENCES tbl_name (index_col_name, ...)  
  [ON DELETE reference_option]  
  [ON UPDATE reference_option]  
| UNIQUE [INDEX|KEY] [index_name] [index_type] (index_col_name, ...) [index_option] ...  
| CHECK (check_constraints)  
  
index_type:  
  USING {BTREE | HASH | RTREE}  
  
index_col_name:  
  col_name [(length)] [ASC | DESC]  
  
index_option:  
  | KEY_BLOCK_SIZE [=] value  
  | index_type  
  | WITH PARSER parser_name  
  | COMMENT 'string'  
  | CLUSTERING={YES|NO}  
  
reference_option:  
  RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [FOREIGN KEY Constraints](#)
  2. [CHECK Constraints](#)
  3. [Replication](#)
  4. [Auto\\_increment](#)
3. [Examples](#)
4. [See Also](#)

## Description

Constraints provide restrictions on the data you can add to a table. This allows you to enforce data integrity from MariaDB, rather than through application logic. When a statement violates a constraint, MariaDB throws an error.

There are four types of table constraints:

Constraint	Description
PRIMARY KEY	Sets the column for referencing rows. Values must be unique and not null.
FOREIGN KEY	Sets the column to reference the primary key on another table.
UNIQUE	Requires values in column or columns only occur once in the table.
CHECK	Checks whether the data meets the given condition.

The [Information Schema TABLE\\_CONSTRAINTS Table](#) contains information about tables that have constraints.

## FOREIGN KEY Constraints

InnoDB supports [foreign key](#) constraints. The syntax for a foreign key constraint definition in InnoDB looks like this:

```
[CONSTRAINT [symbol]] FOREIGN KEY  
    [index_name] (index_col_name, ...)  
    REFERENCES tbl_name (index_col_name,...)  
    [ON DELETE reference_option]  
    [ON UPDATE reference_option]  
  
reference_option:  
    RESTRICT | CASCADE | SET NULL | NO ACTION
```

The [Information Schema REFERENTIAL\\_CONSTRAINTS](#) table has more information about foreign keys.

## CHECK Constraints

MariaDB starting with [10.2.1](#)

From [MariaDB 10.2.1](#), constraints are enforced. Before [MariaDB 10.2.1](#) constraint expressions were accepted in the syntax but ignored.

In [MariaDB 10.2.1](#) you can define constraints in 2 different ways:

- - ```
CHECK(expression)
```

given as part of a column definition.
- ```
CONSTRAINT [constraint_name] CHECK (expression)
```

Before a row is inserted or updated, all constraints are evaluated in the order they are defined. If any constraint expression returns false, then the row will not be inserted or updated. One can use most deterministic functions in a constraint, including [UDFs](#).

```
CREATE TABLE t1 (a INT CHECK (a>2), b INT CHECK (b>2), CONSTRAINT a_greater CHECK (a>b));
```

If you use the second format and you don't give a name to the constraint, then the constraint will get an automatically generated name. This is done so

that you can later delete the constraint with `ALTER TABLE DROP constraint_name`.

One can disable all constraint expression checks by setting the `check_constraint_checks` variable to

`OFF`

. This is useful for example when loading a table that violates some constraints that you want to later find and fix in SQL.

## Replication

In [row-based replication](#), only the master checks constraints, and failed statements will not be replicated. In [statement-based replication](#), the slaves will also check constraints. Constraints should therefore be identical, as well as deterministic, in a replication environment.

## Auto\_increment

MariaDB starting with 10.2.6

- From MariaDB 10.2.6, `auto_increment` columns are no longer permitted in check constraints. Previously they were permitted, but would not work correctly. See [MDEV-11117](#).

## Examples

```
CREATE TABLE product (category INT NOT NULL, id INT NOT NULL,
                      price DECIMAL,
                      PRIMARY KEY(category, id)) ENGINE=INNODB;
CREATE TABLE customer (id INT NOT NULL,
                      PRIMARY KEY (id)) ENGINE=INNODB;
CREATE TABLE product_order (no INT NOT NULL AUTO_INCREMENT,
                           product_category INT NOT NULL,
                           product_id INT NOT NULL,
                           customer_id INT NOT NULL,
                           PRIMARY KEY(no),
                           INDEX (product_category, product_id),
                           FOREIGN KEY (product_category, product_id)
                             REFERENCES product(category, id)
                             ON UPDATE CASCADE ON DELETE RESTRICT,
                           INDEX (customer_id),
                           FOREIGN KEY (customer_id)
                             REFERENCES customer(id)) ENGINE=INNODB;
```

MariaDB starting with 10.2.1

The following examples will work from MariaDB 10.2.1 onwards.

Numeric constraints and comparisons:

```
CREATE TABLE t1 (a INT CHECK (a>2), b INT CHECK (b>2), CONSTRAINT a_greater CHECK (a>b));

INSERT INTO t1(a) VALUES (1);
ERROR 4022 (23000): CONSTRAINT `a` failed for `test`.`t1`

INSERT INTO t1(a,b) VALUES (3,4);
ERROR 4022 (23000): CONSTRAINT `a_greater` failed for `test`.`t1`

INSERT INTO t1(a,b) VALUES (4,3);
Query OK, 1 row affected (0.04 sec)
```

Dropping a constraint:

```
ALTER TABLE t1 DROP CONSTRAINT a_greater;
```

Adding a constraint:

```
ALTER TABLE t1 ADD CONSTRAINT a_greater CHECK (a>b);
```

Date comparisons and character length:

```

CREATE TABLE t2 (name VARCHAR(30) CHECK (CHAR_LENGTH(name)>2), start_date DATE,
end_date DATE CHECK (start_date IS NULL OR end_date IS NULL OR start_date<end_date));

INSERT INTO t2(name, start_date, end_date) VALUES('Ione', '2003-12-15', '2014-11-09');
Query OK, 1 row affected (0.04 sec)

INSERT INTO t2(name, start_date, end_date) VALUES('Io', '2003-12-15', '2014-11-09');
ERROR 4222 (23000): CONSTRAINT `name` failed for `test`.`t2`

INSERT INTO t2(name, start_date, end_date) VALUES('Ione', NULL, '2014-11-09');
Query OK, 1 row affected (0.04 sec)

INSERT INTO t2(name, start_date, end_date) VALUES('Ione', '2015-12-15', '2014-11-09');
ERROR 4222 (23000): CONSTRAINT `end_date` failed for `test`.`t2`

```

A misplaced parenthesis:

```

CREATE TABLE t3 (name VARCHAR(30) CHECK (CHAR_LENGTH(name)>2), start_date DATE,
end_date DATE CHECK (start_date IS NULL OR end_date IS NULL OR start_date<end_date));
Query OK, 0 rows affected (0.32 sec)

INSERT INTO t3(name, start_date, end_date) VALUES('Io', '2003-12-15', '2014-11-09');
Query OK, 1 row affected, 1 warning (0.04 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message           |
+-----+-----+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'Io' |
+-----+-----+

```

Compare the definition of table *t2* to table *t3*.

CHAR\_LENGTH(name)>2

is very different to

CHAR\_LENGTH(name)>2)

as the latter mistakenly performs a numeric comparison on the *name* field, leading to unexpected results.

## See Also

- [Foreign Keys](#)

### 1.1.3.6 MERGE

#### Contents

1. [Description](#)
2. [Examples](#)

## Description

The MERGE storage engine, also known as the MRG\_MyISAM engine, is a collection of identical MyISAM tables that can be used as one. "Identical" means that all tables have identical column and index information. You cannot merge MyISAM tables in which the columns are listed in a different order, do not have exactly the same columns, or have the indexes in different order. However, any or all of the MyISAM tables can be compressed with `myisampack`. Columns names and indexes names can be different, as long as data types and NULL/NOT NULL clauses are the same. Differences in table options such as AVG\_ROW\_LENGTH, MAX\_ROWS, or PACK\_KEYS do not matter.

Each index in a MERGE table must match an index in underlying MyISAM tables, but the opposite is not true. Also, a MERGE table cannot have a PRIMARY KEY or UNIQUE indexes, because it cannot enforce uniqueness over all underlying tables.

The following options are meaningful for MERGE tables:

- - UNION
    - . This option specifies the list of the underlying MyISAM tables. The list is enclosed between parentheses and separated with commas.
- - INSERT\_METHOD
    - . This option specifies whether, and how, INSERTs are allowed for the table. Allowed values are:
      - NO
      - (INSERTs are not allowed),
      - FIRST
      - (new rows will be written into the first table specified in the
      - UNION
      - list),

```

LAST
(new rows will be written into the last table specified in the
UNION
list). The default value is
NO
.

```

If you define a MERGE table with a definition which is different from the underlying MyISAM tables, or one of the underlying tables is not MyISAM, the CREATE TABLE statement will not return any error. But any statement which involves the table will produce an error like the following:

```
ERROR 1168 (HY000): Unable to open underlying table which is differently defined
or of non-MyISAM type or doesn't exist
```

A

[CHECK TABLE](#)

will show more information about the problem.

The error is also produced if the table is properly define, but an underlying table's definition changes at some point in time.

If you try to insert a new row into a MERGE table with INSERT\_METHOD=NO, you will get an error like the following:

```
ERROR 1036 (HY000): Table 'tbl_name' is read only
```

It is possible to build a MERGE table on MyISAM tables which have one or more [virtual columns](#). MERGE itself does not support virtual columns, thus such columns will be seen as regular columns. The data types and sizes will still need to be identical, and they cannot be NOT NULL.

## Examples

```

CREATE TABLE t1 (
    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    message CHAR(20) ENGINE=MyISAM;

CREATE TABLE t2 (
    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    message CHAR(20) ENGINE=MyISAM;

INSERT INTO t1 (message) VALUES ('Testing'),('table'),('t1');

INSERT INTO t2 (message) VALUES ('Testing'),('table'),('t2');

CREATE TABLE total (
    a INT NOT NULL AUTO_INCREMENT,
    message CHAR(20), INDEX(a))
ENGINE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;

SELECT * FROM total;
+-----+
| a | message |
+-----+
| 1 | Testing |
| 2 | table   |
| 3 | t1      |
| 1 | Testing |
| 2 | table   |
| 3 | t2      |
+-----+

```

In the following example, we'll create three MyISAM tables, and then a MERGE table on them. However, one of them uses a different data type for the column b, so a SELECT will produce an error:

```

CREATE TABLE t1 (
  a INT,
  b INT
) ENGINE = MyISAM;

CREATE TABLE t2 (
  a INT,
  b INT
) ENGINE = MyISAM;

CREATE TABLE t3 (
  a INT,
  b TINYINT
) ENGINE = MyISAM;

CREATE TABLE t_mrg (
  a INT,
  b INT
) ENGINE = MERGE,UNION=(t1,t2,t3);

SELECT * FROM t_mrg;
ERROR 1168 (HY000): Unable to open underlying table which is differently defined
or of non-MyISAM type or doesn't exist

```

To find out what's wrong, we'll use a CHECK TABLE:

Table	Op	Msg_type	Msg_text
test.t_mrg	check	Error	Table 'test.t3' is differently defined or of non-MyISAM type or doesn't exist
test.t_mrg	check	Error	Unable to open underlying table which is differently defined or of non-MyISAM type or doesn't
test.t_mrg	check	error	Corrupt

Now, we know that the problem is in

t3  
's definition.

## 1.1.3.7 RENAME TABLE

## 1.1.3.8 TRUNCATE TABLE

## 1.1.4 Data Manipulation

### 1.1.4.1 Selecting Data

#### 1.1.4.1.1 SELECT

## Syntax

```

SELECT
  [ALL | DISTINCT | DISTINCTROW]
  [HIGH_PRIORITY]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
  [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
  select_expr [, select_expr ...]
  [ FROM table_references
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset [ROWS EXAMINED rows_limit]} | 
      [OFFSET start { ROW | ROWS }]
      [FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } { ONLY | WITH TIES } ] ]
    procedure|[PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] [export_options] |
      INTO DUMPFILE 'file_name' | INTO var_name [, var_name] ]
    [FOR UPDATE lock_option | LOCK IN SHARE MODE lock_option]

export_options:
  [{FIELDS | COLUMNS}
    [TERMINATED BY 'string']
    [[OPTIONALLY] ENCLOSED BY 'char']
    [ESCAPED BY 'char']
  ]
  [LINES
    [STARTING BY 'string']
    [TERMINATED BY 'string']
  ]
]

lock_option:
  [WAIT n | NOWAIT | SKIP LOCKED]

```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [Select Expressions](#)
  2. [DISTINCT](#)
  3. [INTO](#)
  4. [LIMIT](#)
  5. [LOCK IN SHARE MODE/FOR UPDATE](#)
  6. [OFFSET ... FETCH](#)
  7. [ORDER BY](#)
  8. [PARTITION](#)
  9. [PROCEDURE](#)
  10. [SKIP LOCKED](#)
  11. [SQL\\_CALC\\_FOUND\\_ROWS](#)
  12. [max\\_statement\\_time clause](#)
  13. [WAIT/NOWAIT](#)
3. [Examples](#)
4. [See Also](#)

## Description

`SELECT` is used to retrieve rows selected from one or more tables, and can include [UNION](#) statements and [subqueries](#).

- Each `select_expr` expression indicates a column or data that you want to retrieve. You must have at least one select expression. See [Select Expressions](#) below.
- The
 

```
FROM
clause indicates the table or tables from which to retrieve rows. Use either a single table name or a
JOIN
expression. See JOIN for details. If no table is involved, FROM DUAL can be specified.
```
- Each table can also be specified as
 

```
db_name
```

```

tbl_name
. Each column can also be specified as
tbl_name

.
col_name
or even
db_name

.
tbl_name

.
col_name
. This allows one to write queries which involve multiple databases. See Identifier Qualifiers for syntax details.

```

- The

WHERE

clause, if given, indicates the condition or conditions that rows must satisfy to be selected.

where\_condition

is an expression that evaluates to true for each row to be selected. The statement selects all rows if there is no WHERE clause.

- In the

WHERE

clause, you can use any of the functions and operators that MariaDB supports, except for aggregate (summary) functions. See [Functions and Operators](#) and [Functions and Modifiers for use with GROUP BY](#) (aggregate).

- Use the [ORDER BY](#) clause to order the results.

- Use the [LIMIT](#) clause allows you to restrict the results to only a certain number of rows, optionally with an offset.

- Use the [GROUP BY](#) and

HAVING

clauses to group rows together when they have columns or computed values in common.

SELECT can also be used to retrieve rows computed without reference to any table.

## Select Expressions

A

SELECT

statement must contain one or more select expressions, separated by commas. Each select expression can be one of the following:

- The name of a column.
- Any expression using [functions and operators](#).
- 

\*

to select all columns from all tables in the

FROM

clause.

- 

tbl\_name.\*

to select all columns from just the table *tbl\_name*.

When specifying a column, you can either use just the column name or qualify the column name with the name of the table using

tbl\_name.col\_name

. The qualified form is useful if you are joining multiple tables in the

FROM

clause. If you do not qualify the column names when selecting from multiple tables, MariaDB will try to find the column in each table. It is an error if that column name exists in multiple tables.

You can quote column names using backticks. If you are qualifying column names with table names, quote each part separately as

`tbl\_name`.`col\_name`

If you use any [grouping functions](#) in any of the select expressions, all rows in your results will be implicitly grouped, as if you had used

GROUP BY NULL

## DISTINCT

A query may produce some identical rows. By default, all rows are retrieved, even when their values are the same. To explicitly specify that you want to retrieve identical rows, use the

ALL

option. If you want duplicates to be removed from the resultset, use the

DISTINCT

option.

DISTINCTROW

is a synonym for

DISTINCT  
. See also [COUNT DISTINCT](#) and [SELECT UNIQUE](#) in Oracle mode .

## INTO

The

INTO  
clause is used to specify that the query results should be written to a file or variable.

- [SELECT INTO OUTFILE](#) - formatting and writing the result to an external file.
- [SELECT INTO DUMPFILE](#) - binary-safe writing of the unformatted results to an external file.
- [SELECT INTO Variable](#) - selecting and setting variables.

The reverse of

SELECT INTO OUTFILE  
is [LOAD DATA](#) .

## LIMIT

Restricts the number of returned rows. See [LIMIT](#) and [LIMIT ROWS EXAMINED](#) for details.

## LOCK IN SHARE MODE/FOR UPDATE

See [LOCK IN SHARE MODE](#) and [FOR UPDATE](#) for details on the respective locking clauses.

## OFFSET ... FETCH

MariaDB starting with 10.6

See [SELECT ... OFFSET ... FETCH](#) .

## ORDER BY

Order a resultset. See [ORDER BY](#) for details.

## PARTITION

Specifies to the optimizer which partitions are relevant for the query. Other partitions will not be read. See [Partition Pruning and Selection](#) for details.

## PROCEDURE

Passes the whole result set to a C Procedure. See [PROCEDURE](#) and [PROCEDURE ANALYSE](#) (the only built-in procedure not requiring the server to be recompiled).

## SKIP LOCKED

MariaDB starting with 10.6

The SKIP LOCKED clause was introduced in [MariaDB 10.6.0](#) .

This causes those rows that couldn't be locked ( [LOCK IN SHARE MODE](#) or [FOR UPDATE](#) ) to be excluded from the result set. An explicit  
NOWAIT  
is implied here. This is only implemented on [InnoDB](#) tables and ignored otherwise.

## SQL\_CALC\_FOUND\_ROWS

When

`SQL_CALC_FOUND_ROWS`

is used, then MariaDB will calculate how many rows would have been in the result, if there would be no [LIMIT](#) clause. The result can be found by calling the function [FOUND\\_ROWS\(\)](#) in your next sql statement.

## max\_statement\_time clause

By using [max\\_statement\\_time](#) in conjunction with [SET STATEMENT](#) , it is possible to limit the execution time of individual queries. For example:

```
SET STATEMENT max_statement_time=100 FOR
SELECT field1 FROM table_name ORDER BY field1;
```

## WAIT/NOWAIT

Set the lock wait timeout. See [WAIT](#) and [NOWAIT](#).

## Examples

```
SELECT f1,f2 FROM t1 WHERE (f3<=10) AND (f4='y');
```

See [Getting Data from MariaDB](#) (Beginner tutorial), or the various sub-articles, for more examples.

## See Also

- [Getting Data from MariaDB](#) (Beginner tutorial)
- [Joins and Subqueries](#)
- [LIMIT](#)
- [ORDER BY](#)
- [GROUP BY](#)
- [Common Table Expressions](#)
- [SELECT WITH ROLLUP](#)
- [SELECT INTO OUTFILE](#)
- [SELECT INTO DUMPFILE](#)
- [FOR UPDATE](#)
- [LOCK IN SHARE MODE](#)
- [Optimizer Hints](#)
- Oracle mode from MariaDB 10.3

### 1.1.4.1.2 Joins and Subqueries

#### 1.1.4.1.2.1 Joins

##### 1.1.4.1.2.1.1 Joining Tables with JOIN Clauses

In the absence of a more tutorial-level document, here is a simple example of three basic JOIN types, which you can experiment with in order to see what the different joins accomplish:

```
CREATE TABLE t1 ( a INT );
CREATE TABLE t2 ( b INT );
INSERT INTO t1 VALUES (1), (2), (3);
INSERT INTO t2 VALUES (2), (4);
SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b;
SELECT * FROM t1 CROSS JOIN t2;
SELECT * FROM t1 LEFT JOIN t2 ON t1.a = t2.b;
SELECT * FROM t2 LEFT JOIN t1 ON t1.a = t2.b;
```

The first two SELECTs are (unfortunately) commonly written with an older form:

```
SELECT * FROM t1, t2 WHERE t1.a = t2.b;
SELECT * FROM t1, t2;
```

What you can see from this is that an **INNER JOIN** produces a result set containing only rows that have a match, in both tables (t1 and t2), for the specified join condition(s).

A **CROSS JOIN** produces a result set in which every row in each table is joined to every row in the other table; this is also called a **cartesian product**. In MariaDB the CROSS keyword can be omitted, as it does nothing. Any JOIN without an ON clause is a CROSS JOIN.

The **LEFT JOIN** is an **outer join**, which produces a result set with all rows from the table on the "left" (t1); the values for the columns in the other table (t2) depend on whether or not a match was found. If no match is found, all columns from that table are set to NULL for that row.

The **RIGHT JOIN** is similar to the LEFT JOIN, though its resultset contains all rows from the right table, and the left table's columns will be filled with NULLS when needed.

JOINS can be concatenated to read results from three or more tables.

Here is the output of the various SELECT statements listed above:

```
SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b;
-----
| a      | b      |
-----
| 2      | 2      |
-----
1 row in set (0.00 sec)
```

```
SELECT * FROM t1 CROSS JOIN t2;
-----
| a      | b      |
-----
| 1      | 2      |
| 2      | 2      |
| 3      | 2      |
| 1      | 4      |
| 2      | 4      |
| 3      | 4      |
-----
6 rows in set (0.00 sec)
```

```
SELECT * FROM t1 LEFT JOIN t2 ON t1.a = t2.b;
-----
| a      | b      |
-----
| 1      | NULL  |
| 2      | 2      |
| 3      | NULL  |
-----
3 rows in set (0.00 sec)
```

```
SELECT * FROM t2 LEFT JOIN t1 ON t1.a = t2.b;
-----
| b      | a      |
-----
| 2      | 2      |
| 4      | NULL  |
-----
2 rows in set (0.00 sec)
```

That should give you a bit more understanding of how JOINS work!

## Other References

### See Also

- [More Advanced JOINS](#)
- [Comma vs JOIN](#)
- [http://www.keithjbrown.co.uk/vworks/mysql/mysql\\_p5.shtml](http://www.keithjbrown.co.uk/vworks/mysql/mysql_p5.shtml) - Nice tutorial. Part 5 covers joins.

The initial version of this article was copied, with permission, from [http://hashmysql.org/wiki/Introduction\\_to\\_Joins](http://hashmysql.org/wiki/Introduction_to_Joins) on 2012-10-05.

## 1.1.4.1.2.1.2 More Advanced Joins

### Contents

1. [The Employee Database](#)
2. [Working with the Employee Database](#)
  1. [Filtering by Name](#)
  2. [Filtering by Name, Date and Time](#)
  3. [Displaying Total Work Hours per Day](#)
3. [See Also](#)

This article is a follow up to the [Introduction to JOINS](#) page. If you're just getting started with JOINS, go through that page first and then come back here.

## The Employee Database

Let us begin by using an example employee database of a fairly small family business, which does not anticipate expanding in the future.

First, we create the table that will hold all of the employees and their contact information:

```
CREATE TABLE `Employees` (
  `ID` TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `First_Name` VARCHAR(25) NOT NULL,
  `Last_Name` VARCHAR(25) NOT NULL,
  `Position` VARCHAR(25) NOT NULL,
  `Home_Address` VARCHAR(50) NOT NULL,
  `Home_Phone` VARCHAR(12) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM;
```

Next, we add a few employees to the table:

```
INSERT INTO `Employees` (`First_Name`, `Last_Name`, `Position`, `Home_Address`, `Home_Phone`)
VALUES
('Mustapha', 'Mond', 'Chief Executive Officer', '692 Promiscuous Plaza', '326-555-3492'),
('Henry', 'Foster', 'Store Manager', '314 Savage Circle', '326-555-3847'),
('Bernard', 'Marx', 'Cashier', '1240 Ambient Avenue', '326-555-8456'),
('Lenina', 'Crowne', 'Cashier', '281 Bumblepuppy Boulevard', '328-555-2349'),
('Fanny', 'Crowne', 'Restocker', '1023 Bokanovsky Lane', '326-555-6329'),
('Helmholtz', 'Watson', 'Janitor', '944 Soma Court', '329-555-2478');
```

Now, we create a second table, containing the hours which each employee clocked in and out during the week:

```
CREATE TABLE `Hours` (
  `ID` TINYINT(3) UNSIGNED NOT NULL,
  `Clock_In` DATETIME NOT NULL,
  `Clock_Out` DATETIME NOT NULL
) ENGINE=MyISAM;
```

Finally, although it is a lot of information, we add a full week of hours for each of the employees into the second table that we created:

```
INSERT INTO `Hours`
VALUES
('1', '2005-08-08 07:00:42', '2005-08-08 17:01:36'),
('1', '2005-08-09 07:01:34', '2005-08-09 17:10:11'),
('1', '2005-08-10 06:59:56', '2005-08-10 17:09:29'),
('1', '2005-08-11 07:00:17', '2005-08-11 17:00:47'),
('1', '2005-08-12 07:02:29', '2005-08-12 16:59:12'),
('2', '2005-08-08 07:00:25', '2005-08-08 17:03:13'),
('2', '2005-08-09 07:00:57', '2005-08-09 17:05:09'),
('2', '2005-08-10 06:58:43', '2005-08-10 16:58:24'),
('2', '2005-08-11 07:01:58', '2005-08-11 17:00:45'),
('2', '2005-08-12 07:02:12', '2005-08-12 16:58:57'),
('3', '2005-08-08 07:00:12', '2005-08-08 17:01:32'),
('3', '2005-08-09 07:01:10', '2005-08-09 17:00:26'),
('3', '2005-08-10 06:59:53', '2005-08-10 17:02:53'),
('3', '2005-08-11 07:01:15', '2005-08-11 17:04:23'),
('3', '2005-08-12 07:00:51', '2005-08-12 16:57:52'),
('4', '2005-08-08 06:54:37', '2005-08-08 17:01:23'),
('4', '2005-08-09 06:58:23', '2005-08-09 17:00:54'),
('4', '2005-08-10 06:59:14', '2005-08-10 17:00:12'),
('4', '2005-08-11 07:00:49', '2005-08-11 17:00:34'),
('4', '2005-08-12 07:01:09', '2005-08-12 16:58:29'),
('5', '2005-08-08 07:00:04', '2005-08-08 17:01:43'),
('5', '2005-08-09 07:02:12', '2005-08-09 17:02:13'),
('5', '2005-08-10 06:59:39', '2005-08-10 17:03:37'),
('5', '2005-08-11 07:01:26', '2005-08-11 17:00:03'),
('5', '2005-08-12 07:02:15', '2005-08-12 16:59:02'),
('6', '2005-08-08 07:00:12', '2005-08-08 17:01:02'),
('6', '2005-08-09 07:03:44', '2005-08-09 17:00:00'),
('6', '2005-08-10 06:54:19', '2005-08-10 17:03:31'),
('6', '2005-08-11 07:00:05', '2005-08-11 17:02:57'),
('6', '2005-08-12 07:02:07', '2005-08-12 16:58:23');
```

## Working with the Employee Database

Now that we have a cleanly structured database to work with, let us begin this tutorial by stepping up one notch from the last tutorial and filtering our information a little.

## Filtering by Name

Earlier in the week, an anonymous employee reported that Helmholtz came into work almost four minutes late; to verify this, we will begin our investigation by filtering out employees whose first names are "Helmholtz".

```
SELECT
`Employees`.`First_Name`,
`Employees`.`Last_Name`,
`Hours`.`Clock_In`,
`Hours`.`Clock_Out`
FROM `Employees`
INNER JOIN `Hours` ON `Employees`.`ID` = `Hours`.`ID`
WHERE `Employees`.`First_Name` = 'Helmholtz';
```

The result:

First_Name	Last_Name	Clock_In	Clock_Out
Helmholtz	Watson	2005-08-08 07:00:12	2005-08-08 17:01:02
Helmholtz	Watson	2005-08-09 07:03:44	2005-08-09 17:00:00
Helmholtz	Watson	2005-08-10 06:54:19	2005-08-10 17:03:31
Helmholtz	Watson	2005-08-11 07:00:05	2005-08-11 17:02:57
Helmholtz	Watson	2005-08-12 07:02:07	2005-08-12 16:58:23

5 rows in set (0.00 sec)

This is obviously more information than we care to trudge through, considering we only care about when he arrived past 7:00:59 on any given day within this week; thus, we need to add a couple more conditions to our WHERE clause.

## Filtering by Name, Date and Time

In the following example, we will filter out all of the times which Helmholtz clocked in that were before 7:01:00 and during the work week that lasted from the 8th to the 12th of August:

```
SELECT
`Employees`.`First_Name`,
`Employees`.`Last_Name`,
`Hours`.`Clock_In`,
`Hours`.`Clock_Out`
FROM `Employees`
INNER JOIN `Hours` ON `Employees`.`ID` = `Hours`.`ID`
WHERE `Employees`.`First_Name` = 'Helmholtz'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%Y-%m-%d') >= '2005-08-08'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%Y-%m-%d') <= '2005-08-12'
AND DATE_FORMAT(`Hours`.`Clock_In`, '%H:%i:%S') > '07:00:59';
```

The result:

First_Name	Last_Name	Clock_In	Clock_Out
Helmholtz	Watson	2005-08-09 07:03:44	2005-08-09 17:00:00
Helmholtz	Watson	2005-08-12 07:02:07	2005-08-12 16:58:23

2 rows in set (0.00 sec)

We have now, by merely adding a few more conditions, eliminated all of the irrelevant information; Helmholtz was late to work on the 9th and the 12th of August.

## Displaying Total Work Hours per Day

Suppose you would like to based on the information stored in both of our tables in the employee database develop a quick list of the total hours each employee has worked for each day recorded; a simple way to estimate the time each employee worked per day is exemplified below:

```

SELECT
`Employees`.`ID`,
`Employees`.`First_Name`,
`Employees`.`Last_Name`,
`Hours`.`Clock_In`,
`Hours`.`Clock_Out`,
DATE_FORMAT(`Hours`.`Clock_Out`, '%T')-DATE_FORMAT(`Hours`.`Clock_In`, '%T') AS 'Total_Hours'
FROM `Employees` INNER JOIN `Hours` ON `Employees`.`ID` = `Hours`.`ID`;

```

The result (limited by 10):

ID	First_Name	Last_Name	Clock_In	Clock_Out	Total_Hours
1	Mustapha	Mond	2005-08-08 07:00:42	2005-08-08 17:01:36	10
1	Mustapha	Mond	2005-08-09 07:01:34	2005-08-09 17:10:11	10
1	Mustapha	Mond	2005-08-10 06:59:56	2005-08-10 17:09:29	11
1	Mustapha	Mond	2005-08-11 07:00:17	2005-08-11 17:00:47	10
1	Mustapha	Mond	2005-08-12 07:02:29	2005-08-12 16:59:12	9
2	Henry	Foster	2005-08-08 07:00:25	2005-08-08 17:03:13	10
2	Henry	Foster	2005-08-09 07:00:57	2005-08-09 17:05:09	10
2	Henry	Foster	2005-08-10 06:58:43	2005-08-10 16:58:24	10
2	Henry	Foster	2005-08-11 07:01:58	2005-08-11 17:00:45	10
2	Henry	Foster	2005-08-12 07:02:12	2005-08-12 16:58:57	9

10 rows in set (0.00 sec)

## See Also

- [Introduction to JOINS](#)

The first version of this article was copied, with permission, from [http://hashmysql.org/wiki/More\\_Advanced\\_Joins](http://hashmysql.org/wiki/More_Advanced_Joins) on 2012-10-05.

### 1.1.4.1.2.1.3 JOIN Syntax

#### Description

MariaDB supports the following

JOIN  
syntaxes for the  
table\_references  
part of

[SELECT](#)

statements and multiple-table

[DELETE](#)

and

[UPDATE](#)

statements:

```

table_references:
  table_reference [, table_reference] ...

table_reference:
  table_factor
  | join_table

table_factor:
  tbl_name [PARTITION (partition_list)]
    [query_system_time_period_specification] [[AS] alias] [index_hint_list]
  | table_subquery [query_system_time_period_specification] [AS] alias
  | ( table_references )
  | { ON table_reference LEFT OUTER JOIN table_reference
      ON conditional_expr }

join_table:
  table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor

join_condition:
  ON conditional_expr
  | USING (column_list)

query_system_time_period_specification:
  FOR SYSTEM_TIME AS OF point_in_time
  | FOR SYSTEM_TIME BETWEEN point_in_time AND point_in_time
  | FOR SYSTEM_TIME FROM point_in_time TO point_in_time
  | FOR SYSTEM_TIME ALL

point_in_time:
  [TIMESTAMP] expression
  | TRANSACTION expression

index_hint_list:
  index_hint [, index_hint] ...

index_hint:
  USE {INDEX|KEY}
    [{FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
  | IGNORE {INDEX|KEY}
    [{FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
  | FORCE {INDEX|KEY}
    [{FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
  index_name [, index_name] ...

```

A table reference is also known as a join expression.

Each table can also be specified as

```

db_name
.
tbl_name
. This allows to write queries which involve multiple databases. See Identifier Qualifiers for syntax details.

```

The syntax of

```

table_factor
is extended in comparison with the SQL Standard. The latter accepts only
table_reference
, not a list of them inside a pair of parentheses.

```

This is a conservative extension if we consider each comma in a list of table\_reference items as equivalent to an inner join. For example:

```

SELECT * FROM t1 LEFT JOIN (t2, t3, t4)
  ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c)

```

is equivalent to:

```

SELECT * FROM t1 LEFT JOIN (t2 CROSS JOIN t3 CROSS JOIN t4)
  ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c)

```

In MariaDB,

CROSS JOIN  
is a syntactic equivalent to  
INNER JOIN  
(theys can replace each other). In standard SQL, they are not equivalent.  
INNER JOIN  
is used with an  
ON  
clause,  
CROSS JOIN  
is used otherwise.

In general, parentheses can be ignored in join expressions containing only inner join operations. MariaDB also supports nested joins (see <http://dev.mysql.com/doc/refman/5.1/en/nested-join-optimization.html> ).

See [System-versioned tables](#) for more information about

FOR SYSTEM\_TIME  
syntax.

Index hints can be specified to affect how the MariaDB optimizer makes use of indexes. For more information, see [How to force query plans](#).

## Examples

```
SELECT left_tbl.*  

  FROM left_tbl LEFT JOIN right_tbl ON left_tbl.id = right_tbl.id  

 WHERE right_tbl.id IS NULL;
```

### 1.1.4.1.2.1.4 Comma vs JOIN

A query to grab the list of phone numbers for clients who ordered in the last two weeks might be written in a couple of ways. Here are two:

```
SELECT *  

  FROM  

  clients,  

  orders,  

  phoneNumbers  

 WHERE  

  clients.id = orders.clientId  

 AND clients.id = phoneNumbers.clientId  

 AND orderPlaced >= NOW() - INTERVAL 2 WEEK;
```

```
SELECT *  

  FROM  

  clients  

 INNER JOIN orders ON clients.id = orders.clientId  

 INNER JOIN phoneNumbers ON clients.id = phoneNumbers.clientId  

 WHERE  

 orderPlaced >= NOW() - INTERVAL 2 WEEK;
```

Does it make a difference? Not much as written. But you should use the second form. Why?

- **Readability.** Once the WHERE clause contains more than two conditions, it becomes tedious to pick out the difference between business logic (only dates in the last two weeks) and relational logic (which fields relate clients to orders). Using the JOIN syntax with an ON clause makes the WHERE list shorter, and makes it very easy to see how tables relate to each other.

- **Flexibility.** Let's say we need to see all clients even if they don't have a phone number in the system. With the second version, it's easy; just change

```
    INNER JOIN phoneNumbers  

      to
```

```
    LEFT JOIN phoneNumbers
```

. Try that with the first version, and MySQL version 5.0.12+ will issue a syntax error because of the change in precedence between the comma operator and the JOIN keyword. The solution is to rearrange the FROM clause or add parentheses to override the precedence, and that quickly becomes frustrating.

- **Portability.** The changes in 5.0.12 were made to align with SQL:2003. If your queries use standard syntax, you will have an easier time switching to a different database should the need ever arise.

## See Also

- ["MySQL joins: ON vs. USING vs. Theta-style"](#) — An interesting blog entry regarding this topic.

## 1.1.4.1.2.2 Subqueries

### 1.1.4.1.2.2.1 Subqueries

A subquery is a query nested in another query.



#### Scalar Subqueries

*Subquery returning a single value.*



#### Row Subqueries

*Subquery returning a row.*



#### Subqueries and ALL

*Return true if the comparison returns true for each row, or the subquery returns no rows.*



#### Subqueries and ANY

*Return true if the comparison returns true for at least one row returned by the subquery.*



#### Subqueries and EXISTS

*Returns true if the subquery returns any rows.*



#### Subqueries in a FROM Clause

*Subqueries are more commonly placed in a WHERE clause, but can also form part of the FROM clause.*



#### Subquery Optimizations

*Articles about subquery optimizations in MariaDB.*



#### Subqueries and JOINS

*Rewriting subqueries as JOINs, and using subqueries instead of JOINs.*



#### Subquery Limitations

*There are a number of limitations regarding subqueries.*

There are 1 related questions .

## 1.1.4.1.2.2.2 Scalar Subqueries

A scalar subquery is a [subquery](#) that returns a single value. This is the simplest form of a subquery, and can be used in most places a literal or single column value is valid.

The data type, length and [character set and collation](#) are all taken from the result returned by the subquery. The result of a subquery can always be NULL, that is, no result returned. Even if the original value is defined as NOT NULL, this is disregarded.

A subquery cannot be used where only a literal is expected, for example [LOAD DATA INFILE](#) expects a literal string containing the file name, and LIMIT requires a literal integer.

## Examples

```
CREATE TABLE sq1 (num TINYINT);

CREATE TABLE sq2 (num TINYINT);

INSERT INTO sq1 VALUES (1);

INSERT INTO sq2 VALUES (10* (SELECT num FROM sq1));

SELECT * FROM sq2;
+----+
| num |
+----+
| 10 |
+----+
```

Inserting a second row means the subquery is no longer a scalar, and this particular query is not valid:

```

INSERT INTO sq1 VALUES (2);

INSERT INTO sq2 VALUES (10* (SELECT num FROM sq1));
ERROR 1242 (21000): Subquery returns more than 1 row

```

No rows in the subquery, so the scalar is NULL:

```

INSERT INTO sq2 VALUES (10* (SELECT num FROM sq3 WHERE num='3'));

SELECT * FROM sq2;
+-----+
| num |
+-----+
|   10 |
|  NULL |
+-----+

```

A more traditional scalar subquery, as part of a WHERE clause:

```

SELECT * FROM sq1 WHERE num = (SELECT MAX(num)/10 FROM sq2);
+-----+
| num |
+-----+
|   1 |
+-----+

```

## 1.1.4.1.2.2.3 Row Subqueries

A row subquery is a [subquery](#) returning a single row, as opposed to a [scalar subquery](#), which returns a single column from a row, or a literal.

### Examples

```

CREATE TABLE staff (name VARCHAR(10), age TINYINT);

CREATE TABLE customer (name VARCHAR(10), age TINYINT);

INSERT INTO staff VALUES ('Bilhah',37), ('Valerius',61), ('Maia',25);

INSERT INTO customer VALUES ('Thanasis',48), ('Valerius',61), ('Brion',51);

SELECT * FROM staff WHERE (name,age) = (SELECT name,age FROM customer WHERE name='Valerius');
+-----+-----+
| name | age |
+-----+-----+
| Valerius | 61 |
+-----+-----+

```

Finding all rows in one table also in another:

```

SELECT name,age FROM staff WHERE (name,age) IN (SELECT name,age FROM customer);
+-----+-----+
| name | age |
+-----+-----+
| Valerius | 61 |
+-----+-----+

```

## 1.1.4.1.2.2.4 Subqueries and ALL

### Contents

1. [Syntax](#)
2. [Examples](#)

[Subqueries](#) using the ALL keyword will return

true  
if the comparison returns  
true  
for each row returned by the subquery, or the subquery returns no rows.

# Syntax

```
scalar_expression comparison_operator ALL <Table subquery>
```

- scalar\_expression  
may be any expression that evaluates to a single value
- comparison\_operator  
may be any one of:  
=,  
,  
>  
,  
<  
,  
>=  
,  
<=  
,  
<>  
or  
!=

ALL

returns:

- NULL  
if the comparison operator returns  
NULL  
for at least one row returned by the Table subquery or scalar\_expression returns  
NULL
- FALSE  
if the comparison operator returns  
FALSE  
for at least one row returned by the Table subquery.
- TRUE  
if the comparison operator returns  
TRUE  
for all rows returned by the Table subquery, or if Table subquery returns no rows.

NOT IN

is an alias for

<> ALL

## Examples

```
CREATE TABLE sq1 (num TINYINT);
CREATE TABLE sq2 (num2 TINYINT);
INSERT INTO sq1 VALUES(100);
INSERT INTO sq2 VALUES(40),(50),(60);
SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2);
+-----+
| num |
+-----+
| 100 |
+-----+
```

Since

```
100
> all of
40
,
50
and
60
, the evaluation is true and the row is returned
```

Adding a second row to sq1, where the evaluation for that record is false:

```
INSERT INTO sq1 VALUES(30);

SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2);
+-----+
| num |
+-----+
| 100 |
+-----+
```

Adding a new row to sq2, causing all evaluations to be false:

```
INSERT INTO sq2 VALUES(120);

SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2);
Empty set (0.00 sec)
```

When the subquery returns no results, the evaluation is still true:

```
SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2 WHERE num2 > 300);
+-----+
| num |
+-----+
| 100 |
| 30 |
+-----+
```

Evaluating against a NULL will cause the result to be unknown, or not true, and therefore return no rows:

```
INSERT INTO sq2 VALUES (NULL);

SELECT * FROM sq1 WHERE num > ALL (SELECT * FROM sq2);
```

## 1.1.4.1.2.2.5 Subqueries and ANY

### Contents

1. [Syntax](#)
2. [Examples](#)

[Subqueries](#) using the ANY keyword will return

true  
if the comparison returns  
true  
for at least one row returned by the subquery.

## Syntax

The required syntax for an

ANY  
or  
SOME  
quantified comparison is:

```
scalar_expression comparison_operator ANY <Table subquery>
```

Or:

```
scalar_expression comparison_operator SOME <Table subquery>
```

- scalar\_expression  
may be any expression that evaluates to a single value.

- comparison\_operator  
may be any one of  
=,  
,  
>  
,  
<  
,  
>=  
,  
<=  
,  
<>  
or  
!=  
.

ANY  
returns:

- TRUE  
if the comparison operator returns  
TRUE  
for at least one row returned by the Table subquery.
- FALSE  
if the comparison operator returns  
FALSE  
for all rows returned by the Table subquery, or Table subquery has zero rows.
- NULL  
if the comparison operator returns  
NULL  
for at least one row returned by the Table subquery and doesn't return  
TRUE  
for any of them, or if scalar\_expression returns  
NULL  
.

SOME  
is a synonym for  
ANY  
, and  
IN  
is a synonym for  
= ANY

## Examples

```
CREATE TABLE sq1 (num TINYINT);

CREATE TABLE sq2 (num2 TINYINT);

INSERT INTO sq1 VALUES(100);

INSERT INTO sq2 VALUES(40),(50),(120);

SELECT * FROM sq1 WHERE num > ANY (SELECT * FROM sq2);
+-----+
| num |
+-----+
| 100 |
+-----+
```

```
100  
is greater than two of the three values, and so the expression evaluates as true.
```

SOME is a synonym for ANY:

```
SELECT * FROM sq1 WHERE num < SOME (SELECT * FROM sq2);  
+-----+  
| num |  
+-----+  
| 100 |  
+-----+
```

IN  
is a synonym for  
= ANY  
, and here there are no matches, so no results are returned:

```
SELECT * FROM sq1 WHERE num IN (SELECT * FROM sq2);  
Empty set (0.00 sec)
```

```
INSERT INTO sq2 VALUES(100);  
Query OK, 1 row affected (0.05 sec)
```

```
SELECT * FROM sq1 WHERE num <> ANY (SELECT * FROM sq2);  
+-----+  
| num |  
+-----+  
| 100 |  
+-----+
```

Reading this query, the results may be counter-intuitive. It may seem to read as "SELECT \* FROM sq1 WHERE num does not match any results in sq2. Since it does match 100, it could seem that the results are incorrect. However, the query returns a result if the match does not match any of sq2. Since

100  
already does not match  
40  
, the expression evaluates to true immediately, regardless of the 100's matching. It may be more easily readable to use SOME in a case such as this:

```
SELECT * FROM sq1 WHERE num <> SOME (SELECT * FROM sq2);  
+-----+  
| num |  
+-----+  
| 100 |  
+-----+
```

## 1.1.4.1.2.2.6 Subqueries and EXISTS

### Syntax

```
SELECT ... WHERE EXISTS <Table subquery>
```

### Description

Subqueries using the

EXISTS  
keyword will return  
true  
if the subquery returns any rows. Conversely, subqueries using  
NOT EXISTS  
will return  
true  
only if the subquery returns no rows from the table.

EXISTS subqueries ignore the columns specified by the **SELECT** of the subquery, since they're not relevant. For example,

```
SELECT col1 FROM t1 WHERE EXISTS (SELECT * FROM t2);
```

and

```
SELECT col1 FROM t1 WHERE EXISTS (SELECT col2 FROM t2);
```

produce identical results.

## Examples

```
CREATE TABLE sq1 (num TINYINT);
CREATE TABLE sq2 (num2 TINYINT);
INSERT INTO sq1 VALUES(100);
INSERT INTO sq2 VALUES(40),(50),(60);
SELECT * FROM sq1 WHERE EXISTS (SELECT * FROM sq2 WHERE num2>50);
+-----+
| num |
+-----+
| 100 |
+-----+
SELECT * FROM sq1 WHERE NOT EXISTS (SELECT * FROM sq2 GROUP BY num2 HAVING MIN(num2)=40);
Empty set (0.00 sec)
```

### 1.1.4.1.2.2.7 Subqueries in a FROM Clause

Although [subqueries](#) are more commonly placed in a WHERE clause, they can also form part of the FROM clause. Such subqueries are commonly called derived tables.

If a subquery is used in this way, you must also use an AS clause to name the result of the subquery.

#### ORACLE mode

MariaDB starting with [10.6.0](#)

From [MariaDB 10.6.0 , anonymous subqueries in a FROM clause](#) (no AS clause) are permitted in [ORACLE mode](#) .

## Examples

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);
```

Assume that, given the data above, you want to return the average total for all students. In other words, the average of Chun's 148 (75+73), Esben's 74 (43+31), etc.

You cannot do the following:

```
SELECT AVG(SUM(score)) FROM student GROUP BY name;
ERROR 1111 (HY000): Invalid use of group function
```

A subquery in the FROM clause is however permitted:

```
SELECT AVG(sq_sum) FROM (SELECT SUM(score) AS sq_sum FROM student GROUP BY name) AS t;
+-----+
| AVG(sq_sum) |
+-----+
| 134.0000 |
+-----+
```

From MariaDB 10.6 in ORACLE mode , the following is permitted:

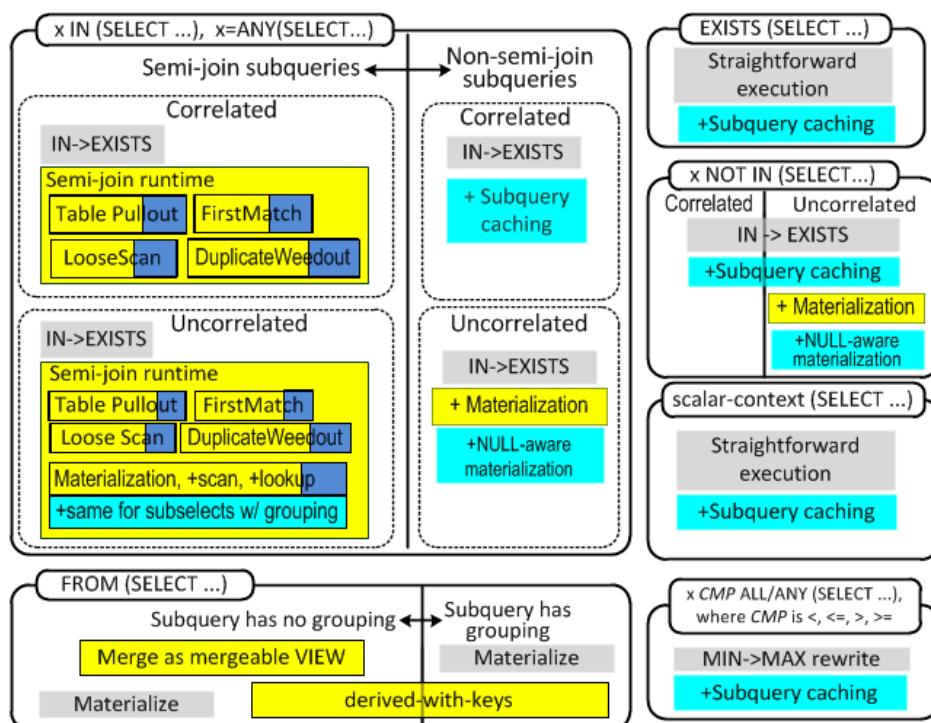
```
SELECT * FROM (SELECT 1 FROM DUAL), (SELECT 2 FROM DUAL);
```

## 1.1.4.1.2.2.8 Subquery Optimizations

### 1.1.4.1.2.2.8.1 Subquery Optimizations Map

Below is a map showing all types of subqueries allowed in the SQL language, and the optimizer strategies available to handle them.

- Uncolored areas represent different kinds of subqueries, for example:
  - Subqueries that have form $x \text{ IN } (\text{SELECT } ...)$
  - Subqueries that are in the  $\text{FROM }$  clause
  - ... and so forth
- The size of each uncolored area roughly corresponds to how important (i.e. frequently used) that kind of subquery is. For example,
  - $x \text{ IN } (\text{SELECT } ...)$  queries are the most important, and
  - $\text{EXISTS } (\text{SELECT } ...)$  are relatively unimportant.
- Colored areas represent optimizations/execution strategies that are applied to handle various kinds of subqueries.
- The color of optimization indicates which version of MySQL/MariaDB it was available in (see legend below)



MySQL 5.1

MariaDB 5.5, MySQL 5.6

MariaDB 5.5 only

MySQL 5.6 only

Some things are not on the map:

- MariaDB doesn't evaluate expensive subqueries when doing optimization (this means, EXPLAIN is always fast). MySQL 5.6 has made a progress in this regard but its optimizer will still evaluate certain kinds of subqueries (for example, scalar-context subqueries used in range predicates)

### Links to pages about individual optimizations:

- [IN->EXISTS](#)
- [Subquery Caching](#)
- [Semi-join optimizations](#)
  - Table pullout
  - FirstMatch
  - Materialization, +scan, +lookup
  - LooseScan
  - DuplicateWeedout execution strategy
- [Non-semi-join Materialization](#) (including NULL-aware and partial matching)

- Derived table optimizations
  - Derived table merge
  - Derived table with keys

## See also

- Subquery optimizations in MariaDB 5.3

### 1.1.4.1.2.2.8.2 Semi-join Subquery Optimizations

#### Contents

1. What is a semi-join subquery
  1. Difference from inner joins
2. Semi-join optimizations in MariaDB
3. See Also

MariaDB has a set of optimizations specifically targeted at *semi-join subqueries*.

## What is a semi-join subquery

A semi-join subquery has a form of

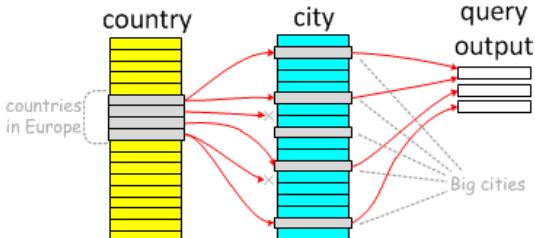
```
SELECT ... FROM outer_tables WHERE expr IN (SELECT ... FROM inner_tables ...)
```

that is, the subquery is an IN-subquery and it is located in the WHERE clause. The most important part here is *with semi-join subquery, we're only interested in records of outer\_tables that have matches in the subquery*

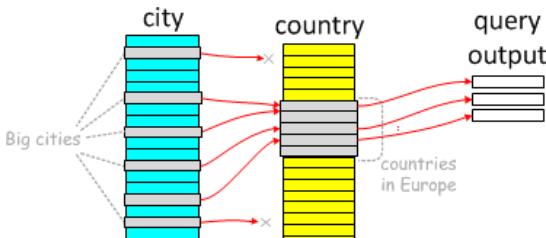
Let's see why this is important. Consider a semi-join subquery:

```
select * from Country
where
  Continent='Europe' and
  Country.Code in (select City.country
                    from City
                    where City.Population>1*1000*1000);
```

One can execute it "naturally", by starting from countries in Europe and checking if they have populous Cities:



The semi-join property also allows "backwards" execution: we can start from big cities, and check which countries they are in:



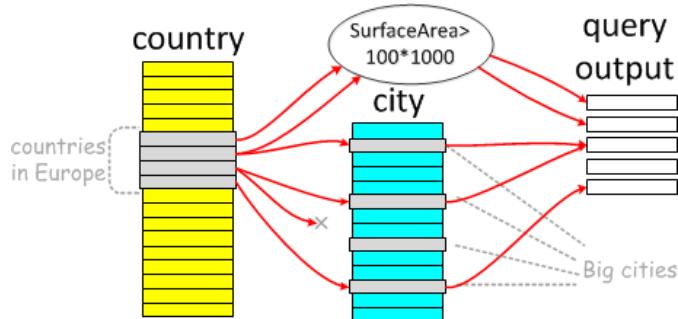
To contrast, let's change the subquery to be non-semi-join:

```
select * from Country
where
  Country.Continent='Europe' and
  (Country.Code in (select City.country
                    from City where City.Population>1*1000*1000)
   or Country.SurfaceArea > 100*1000 -- Added this part
);
```

It is still possible to start from countries, and then check

- if a country has any big cities

- if it has a large surface area:



The opposite, city-to-country way is not possible. This is not a semi-join.

## Difference from inner joins

Semi-join operations are similar to regular relational joins. There is a difference though: with semi-joins, you don't care how many matches an inner table has for an outer row. In the above countries-with-big-cities example, Germany will be returned once, even if it has three cities with populations of more than one million each.

## Semi-join optimizations in MariaDB

MariaDB uses semi-join optimizations to run IN subqueries starting from [MariaDB 5.3](#). Starting in [MariaDB 5.3.3](#), Semi-join subquery optimizations are enabled by default. You can disable them by turning off their `optimizer_switch` like so:

```
SET optimizer_switch='semijoin=off'
```

MariaDB has five different semi-join execution strategies:

- [Table pullout optimization](#)
- [FirstMatch execution strategy](#)
- [Semi-join Materialization execution strategy](#)
- [LooseScan execution strategy](#)
- [DuplicateWeedout execution strategy](#)

## See Also

- [What is MariaDB 5.3](#)
- [Subquery Optimizations Map](#)
- ["Observations about subquery use cases" blog post](#)
- <http://en.wikipedia.org/wiki/Semijoin>

## 1.1.4.1.2.2.8.3 Table Pullout Optimization

### Contents

1. [The idea of Table Pullout](#)
2. [Table pullout in action](#)
3. [Table pullout fact sheet](#)
4. [Controlling table pullout](#)

Table pullout is an optimization for [Semi-join subqueries](#).

## The idea of Table Pullout

Sometimes, a subquery can be re-written as a join. For example:

```
select *
from City
where City.Country in (select Country.Code
                      from Country
                      where Country.Population < 100*1000);
```

If we know that there can be, at most, one country with with a given value of

`Country.Code`

(we can tell that if we see that table `Country` has a primary key or unique index over that column), we can re-write this query as:

```

select City.*
from
  City, Country
where
  City.Country=Country.Code AND Country.Population < 100*1000;

```

## Table pullout in action

If one runs

`EXPLAIN`

for the above query in MySQL 5.1-5.6 or MariaDB 5.1 -5.2, they'll get this plan:

```

MySQL [world]> explain select * from City where City.Country in (select Country.Code from Country where Country.Population < 100
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | City | ALL | NULL | NULL | NULL | NULL | 4079 | Using where |
| 2 | DEPENDENT SUBQUERY | Country | unique_subquery | PRIMARY,Population | PRIMARY | 3 | func | 1 | Using where |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

It shows that the optimizer is going to do a full scan on table

`City`

, and for each city it will do a lookup in table

`Country`

If one runs the same query in MariaDB 5.3 , they will get this plan:

```

MariaDB [world]> explain select * from City where City.Country in (select Country.Code from Country where Country.Population < 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | Country | range | PRIMARY,Population | Population | 4 | NULL | 37 | Using index condit
| 1 | PRIMARY | City | ref | Country | Country | 3 | world.Country.Code | 18 |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

The interesting parts are:

- Both tables have
  - `select_type=PRIMARY`
  - , and
  - `id=1`
  - as if they were in one join.
- The `Country` table is first, followed by the `City` table.

Indeed, if one runs EXPLAIN EXTENDED; SHOW WARNINGS, they will see that the subquery is gone and it was replaced with a join:

```

MariaDB [world]> show warnings
***** 1. row *****
Level: Note
Code: 1003
Message: select `world`.`City`.`ID` AS `ID`,`world`.`City`.`Name` AS
`Name`, `world`.`City`.`Country` AS `Country`, `world`.`City`.`Population` AS
`Population`

from `world`.`City` join `world`.`Country` where

(`world`.`City`.`Country` = `world`.`Country`.`Code`) and (`world`.`Country` .
`Population` < (100 * 1000))
1 row in set (0.00 sec)

```

Changing the subquery into a join allows feeding the join to the join optimizer, which can make a choice between two possible join orders:

1. City -> Country
2. Country -> City

as opposed to the single choice of

## 1. City->Country

which we had before the optimization.

In the above example, the choice produces a better query plan. Without pullout, the query plan with a subquery would read

$$(4079 + 1 * 4079) = 8158$$

table records. With table pullout, the join plan would read

$$(37 + 37 * 18) = 703$$

rows. Not all row reads are equal, but generally, reading

10

times fewer table records is faster.

## Table pullout fact sheet

- Table pullout is possible only in semi-join subqueries.

- Table pullout is based on

UNIQUE

/

PRIMARY

key definitions.

- Doing table pullout does not cut off any possible query plans, so MariaDB will always try to pull out as much as possible.

- Table pullout is able to pull individual tables out of subqueries to their parent selects. If all tables in a subquery have been pulled out, the subquery (i.e. its semi-join) is removed completely.

- One common bit of advice for optimizing MySQL has been "If possible, rewrite your subqueries as joins". Table pullout does exactly that, so manual rewrites are no longer necessary.

## Controlling table pullout

There is no separate @@optimizer\_switch flag for table pullout. Table pullout can be disabled by switching off all semi-join optimizations with

```
SET @@optimizer_switch='semijoin=off'  
command.
```

## 1.1.4.1.2.2.8.4 Non-semi-join Subquery Optimizations

### Contents

- 1. Applicability
  - 1. Subquery in a disjunction (OR)
  - 2. Negated subquery predicate (NOT IN)
  - 3. Subquery in the SELECT or HAVING clause
  - 4. Subquery with a UNION
- 2. Materialization for non-correlated IN-subqueries
  - 1. Materialization basics
  - 2. NULL-aware efficient execution
  - 3. Limitations
- 3. The IN-TO-EXISTS transformation
- 4. Performance discussion
  - 1. Example speedup over MySQL 5.x and MariaDB 5.1/5.2
  - 2. Performance guidelines
- 5. Optimizer control

Certain kinds of IN-subqueries cannot be flattened into [semi-joins](#). These subqueries can be both correlated or non-correlated. In order to provide consistent performance in all cases, MariaDB provides several alternative strategies for these types of subqueries. Whenever several strategies are possible, the optimizer chooses the optimal one based on cost estimates.

The two primary non-semi-join strategies are materialization (also called outside-in materialization), and in-to-exists transformation. Materialization is applicable only for non-correlated subqueries, while in-to-exist can be used both for correlated and non-correlated subqueries.

## Applicability

An IN subquery cannot be flattened into a semi-join in the following cases. The examples below use the *World* database from the MariaDB regression test suite.

### Subquery in a disjunction (OR)

The subquery is located directly or indirectly under an OR operation in the WHERE clause of the outer query.

Query pattern:

```
SELECT ... FROM ... WHERE (expr1, ..., exprN) [NOT] IN (SELECT ... ) OR expr;
```

Example:

```
SELECT Name FROM Country
WHERE (Code IN (select Country from City where City.Population > 100000) OR
      Name LIKE 'L%') AND
      surfacearea > 1000000;
```

## Negated subquery predicate (NOT IN)

The subquery predicate itself is negated.

Query pattern:

```
SELECT ... FROM ... WHERE ... (expr1, ..., exprN) NOT IN (SELECT ... ) ...;
```

Example:

```
SELECT Country.Name
FROM Country, CountryLanguage
WHERE Code NOT IN (SELECT Country FROM CountryLanguage WHERE Language = 'English')
      AND CountryLanguage.Language = 'French'
      AND Code = Country;
```

## Subquery in the SELECT or HAVING clause

The subquery is located in the SELECT or HAVING clauses of the outer query.

Query pattern:

```
SELECT field1, ..., (SELECT ...) WHERE ...;
SELECT ... WHERE ... HAVING (SELECT ...);
```

Example:

```
select Name, City.id in (select capital from Country where capital is not null) as is_capital
from City
where City.population > 1000000;
```

## Subquery with a UNION

The subquery itself is a UNION, while the IN predicate may be anywhere in the query where IN is allowed.

Query pattern:

```
... [NOT] IN (SELECT ... UNION SELECT ...)
```

Example:

```
SELECT * from City where (Name, 91) IN
(SELECT Name, round(Population/1000) FROM City WHERE Country = "IND" AND Population > 2500000
UNION
SELECT Name, round(Population/1000) FROM City WHERE Country = "IND" AND Population < 100000);
```

## Materialization for non-correlated IN-subqueries

### Materialization basics

The basic idea of subquery materialization is to execute the subquery and store its result in an internal temporary table indexed on all its columns. Naturally, this is possible only when the subquery is non-correlated. The IN predicate tests whether its left operand is present in the subquery result. Therefore it is not necessary to store duplicate subquery result rows in the temporary table. Storing only unique subquery rows provides two benefits - the size of the temporary table is smaller, and the index on all its columns can be unique.

If the size of the temporary table is less than the tmp\_table\_size system variable, the table is a hash-indexed in-memory HEAP table. In the rare cases when the subquery result exceeds this limit, the temporary table is stored on disk in an ARIA or MyISAM B-tree indexed table (ARIA is the default).

Subquery materialization happens on demand during the first execution of the IN predicate. Once the subquery is materialized, the IN predicate is evaluated very efficiently by index lookups of the outer expression into the unique index of the materialized temporary table. If there is a match, IN is

TRUE, otherwise IN is FALSE.

## NULL-aware efficient execution

An IN predicate may produce a NULL result if there is a NULL value in either of its arguments. Depending on its location in a query, a NULL predicate value is equivalent to FALSE. These are the cases when substituting NULL with FALSE would reject exactly the same result rows. A NULL result of IN is indistinguishable from a FALSE if the IN predicate is:

- not negated,
- not a function argument,
- inside a WHERE or ON clause.

In all these cases the evaluation of IN is performed as described in the previous paragraph via index lookups into the materialized subquery. In all remaining cases when NULL cannot be substituted with FALSE, it is not possible to use index lookups. This is not a limitation in the server, but a consequence of the NULL semantics in the ANSI SQL standard.

Suppose an IN predicate is evaluated as

```
NULL IN (select  
not_null_col from t1)
```

, that is, the left operand of IN is a NULL value, and there are no NULLs in the subquery. In this case the value of IN is neither FALSE, nor TRUE. Instead it is NULL. If we were to perform an index lookup with the NULL as a key, such a value would not be found in not\_null\_col, and the IN predicate would incorrectly produce a FALSE.

In general, an NULL value on either side of an IN acts as a "wildcard" that matches any value, and if a match exists, the result of IN is NULL. Consider the following example:

If the left argument of IN is the row:

```
(
```

```
7
```

```
,
```

```
NULL
```

```
,
```

```
9
```

```
)
```

, and the result of the right subquery operand of IN is the table:

```
(7, 8, 10)  
(6, NULL, NULL)  
(7, 11, 9)
```

The the IN predicate matches the row

```
(
```

```
7
```

```
,
```

```
11
```

```
,
```

)

, and the result of IN is NULL. Matches where the differing values on either side of the IN arguments are matched by a NULL in the other IN argument, are called *partial matches*.

In order to efficiently compute the result of an IN predicate in the presence of NULLs, MariaDB implements two special algorithms for [partial matching](#), described here in detail .

- Rowid-merge partial matching

This technique is used when the number of rows in the subquery result is above a certain limit. The technique creates special indexes on some of the columns of the temporary table, and merges them by alternative scanning of each index thus performing an operation similar to set-intersection.

- Table scan partial matching

This algorithm is used for very small tables when the overhead of the rowid-merge algorithm is not justifiable. Then the server simply scans the materialized subquery, and checks for partial matches. Since this strategy doesn't need any in-memory buffers, it is also used when there is not enough memory to hold the indexes of the rowid-merge strategy.

## Limitations

In principle the subquery materialization strategy is universal, however, due to some technical limitations in the MariaDB server, there are few cases when the server cannot apply this optimization.

- BLOB fields

Either the left operand of an IN predicate refers to a BLOB field, or the subquery selects one or more BLOBS.

- Incomparable fields

TODO

In the above cases, the server reverts to the [IN-TO-EXISTS](#) transformation.

## The IN-TO-EXISTS transformation

This optimization is the only subquery execution strategy that existed in older versions of MariaDB and MySQL prior to [MariaDB 5.3](#) . We have made various changes and fixed a number of bugs in this code as well, but in essence it remains the same.

## Performance discussion

### Example speedup over MySQL 5.x and [MariaDB 5.1 /5.2](#)

Depending on the query and data, either of the two strategies described here may result in orders of magnitude better/worse plan than the other strategy.

Older versions of MariaDB and any current MySQL version (including MySQL 5.5, and MySQL 5.6 DMR as of July 2011) implement only the IN-TO-EXISTS transformation. As illustrated below, this strategy is inferior in many common cases to subquery materialization.

Consider the following query over the data of the DBT3 benchmark scale 10. Find customers with top balance in their nations:

```
SELECT * FROM part
WHERE p_partkey IN
    (SELECT l_partkey FROM lineitem
     WHERE l_shipdate between '1997-01-01' and '1997-02-01')
ORDER BY p_retailprice DESC LIMIT 10;
```

The times to run this query is as follows:

- Execution time in [MariaDB 5.2 /MySQL 5.x \(any MySQL\)](#): > 1 h

The query takes more than one hour (we didn't wait longer), which makes it impractical to use subqueries in such cases. The EXPLAIN below shows that the subquery was transformed into a correlated one, which indicates an IN-TO-EXISTS transformation.

id   select_type   table   type   key   ref   rows   Extra
1   PRIMARY   part   ALL   NULL   NULL   199755   Using where; Using filesort
2   DEPENDENT SUBQUERY   lineitem   index_subquery   i_l_suppkey_partkey   func   14   Using where

- Execution time in [MariaDB 5.3](#) : 43 sec

In [MariaDB 5.3](#) it takes less than a minute to run the same query. The EXPLAIN shows that the subquery remains uncorrelated, which is an indication that it is being executed via subquery materialization.

id	select_type	table	type	key	ref	rows	Extra
1	PRIMARY	part	ALL	NULL	NULL	199755	Using temporary; Using filesort
1	PRIMARY	<subquery2>	eq_ref	distinct_key	func	1	
2	MATERIALIZED	lineitem	range	l_shipdate_partkey	NULL	160060	Using where; Using index

The speedup here is practically infinite, because both MySQL and older MariaDB versions cannot complete the query in any reasonable time.

In order to show the benefits of partial matching we extended the *customer* table from the DBT3 benchmark with two extra columns:

- c\_pref\_nationkey - preferred nation to buy from,
- c\_pref\_brand - preferred brand.

Both columns are prefixed with the percent NULL values in the column, that is, c\_pref\_nationkey\_05 contains 5% NULL values.

Consider the query "Find all customers that didn't buy from a preferred country, and from a preferred brand withing some date ranges":

```
SELECT count(*)
FROM customer
WHERE (c_custkey, c_pref_nationkey_05, c_pref_brand_05) NOT IN
  (SELECT o_custkey, s_nationkey, p_brand
   FROM orders, supplier, part, lineitem
   WHERE l_orderkey = o_orderkey AND
     l_suppkey = s_suppkey AND
     l_partkey = p_partkey AND
     p_retailprice < 1200 AND
     l_shipdate >= '1996-04-01' AND l_shipdate < '1996-04-05' AND
     o_orderdate >= '1996-04-01' AND o_orderdate < '1996-04-05');
```

- Execution time in [MariaDB 5.2](#) /MySQL 5.x (any MySQL): **40 sec**
- Execution time in [MariaDB 5.3](#) : **2 sec**

The speedup for this query is 20 times.

## Performance guidelines

TODO

## Optimizer control

In certain cases it may be necessary to override the choice of the optimizer. Typically this is needed for benchmarking or testing purposes, or to mimic the behavior of an older version of the server, or if the optimizer made a poor choice.

All the above strategies can be controlled via the following switches in `optimizer_switch` system variable.

- materialization=on/off

In some very special cases, even if materialization was forced, the optimizer may still revert to the IN-TO-EXISTS strategy if materialization is not applicable. In the cases when materialization requires partial matching (because of the presence of NULL values), there are two subordinate switches that control the two partial matching strategies:

- partial\_match\_rowid\_merge=on/off

This switch controls the Rowid-merge strategy. In addition to this switch, the system variable `rowid_merge_buff_size` controls the maximum memory available to the Rowid-merge strategy.

- partial\_match\_table\_scan=on/off

Controls the alternative partial match strategy that performs matching via a table scan.

- in\_to\_exists=on/off

This switch controls the IN-TO-EXISTS transformation.

- tmp\_table\_size and max\_heap\_table\_size system variables

The `tmp_table_size` system variable sets the upper limit for internal MEMORY temporary tables. If an internal temporary table exceeds this size, it is converted automatically into a Aria or MyISAM table on disk with a B-tree index. Notice however, that a MEMORY table cannot be larger than `max_heap_table_size`.

The two main optimizer switches - `materialization` and `in_to_exists` cannot be simultaneously off. If both are set to off, the server will issue an error.

## 1.1.4.1.2.2.8.5 Subquery Cache

## Contents

1. Administration
2. Visibility
3. Implementation
4. Performance Impact
  1. Example 1
  2. Example 2
  3. Example 3
  4. Example 4
5. See Also

The goal of the subquery cache is to optimize the evaluation of correlated subqueries by storing results together with correlation parameters in a cache and avoiding re-execution of the subquery in cases where the result is already in the cache.

## Administration

The cache is on by default. One can switch it off using the [optimizer\\_switch](#)

```
subquery_cache  
setting, like so:
```

```
SET optimizer_switch='subquery_cache=off';
```

The efficiency of the subquery cache is visible in 2 statistical variables:

- [Subquery\\_cache\\_hit](#) - Global counter for all subquery cache hits.
- [Subquery\\_cache\\_miss](#) - Global counter for all subquery cache misses.

The session variables [tmp\\_table\\_size](#) and [max\\_heap\\_table\\_size](#) influence the size of in-memory temporary tables in the table used for caching. It cannot grow more than the minimum of the above variables values (see the [Implementation](#) section for details).

## Visibility

Your usage of the cache is visible in

```
EXTENDED EXPLAIN  
output (warnings) as  
<expr_cache></list of parameters//>(</cached expression//>  
. For example:
```

```
EXPLAIN EXTENDED SELECT * FROM t1 WHERE a IN (SELECT b FROM t2);  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | filtered | Extra |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1 | PRIMARY | t1 | ALL | NULL | NULL | NULL | NULL | 2 | 100.00 | Using where |  
| 2 | DEPENDENT SUBQUERY | t2 | ALL | NULL | NULL | NULL | NULL | 2 | 100.00 | Using where |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
2 rows in set, 1 warning (0.00 sec)  
  
SHOW WARNINGS;  
+-----+-----+  
| Level | Code | Message |  
+-----+-----+  
| Note | 1003 | SELECT `test`.`t1`.`a` AS `a` FROM `test`.`t1` WHERE <expr_cache><`test`.`t1`.`a`>(<in_optimizer>(`test`.`t1`.`  
+-----+-----+  
1 row in set (0.00 sec)
```

In the example above the presence of

```
"<expr_cache><`test`.`t1`.`a`>(...)"
```

is how you know you are using the subquery cache.

## Implementation

Every subquery cache creates a temporary table where the results and all parameters are stored. It has a unique index over all parameters. First the cache is created in a [MEMORY](#) table (if doing this is impossible the cache becomes disabled for that expression). When the table grows up to the minimum of

```
tmp_table_size  
and  
max_heap_table_size  
, the hit rate will be checked:
```

- if the hit rate is really small (<0.2) the cache will be disabled.

- if the hit rate is moderate (<0.7) the table will be cleaned (all records deleted) to keep the table in memory
- if the hit rate is high the table will be converted to a disk table (for 5.3.0 it can only be converted to a disk table).

```
hit rate = hit / (hit + miss)
```

## Performance Impact

Here are some examples that show the performance impact of the subquery cache (these tests were made on a 2.53 GHz Intel Core 2 Duo MacBook Pro with dbt-3 scale 1 data set).

example	cache on	cache off	gain	hit	miss	hit rate
1	1.01sec	1 hour 31 min 43.33sec	5445x	149975	25	99.98%
2	0.21sec	1.41sec	6.71x	6285	220	96.6%
3	2.54sec	2.55sec	1.00044x	151	461	24.67%
4	1.87sec	1.95sec	0.96x	0	23026	0%

## Example 1

Dataset from DBT-3 benchmark, a query to find customers with balance near top in their nation:

```
select count(*) from customer
where
    c_acctbal > 0.8 * (select max(c_acctbal)
                          from customer C
                          where C.c_nationkey=customer.c_nationkey
                          group by c_nationkey);
```

## Example 2

DBT-3 benchmark, Query #17

```
select sum(l_extendedprice) / 7.0 as avg_yearly
from lineitem, part
where
    p_partkey = l_partkey and
    p_brand = 'Brand#42' and p_container = 'JUMBO BAG' and
    l_quantity < (select 0.2 * avg(l_quantity) from lineitem
                  where l_partkey = p_partkey);
```

## Example 3

DBT-3 benchmark, Query #2

```
select
    s_acctbal, s_name, n_name, p_partkey, p_mfgr, s_address, s_phone, s_comment
from
    part, supplier, partsupp, nation, region
where
    p_partkey = ps_partkey and s_suppkey = ps_suppkey and p_size = 33
    and p_type like '%STEEL' and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey and r_name = 'MIDDLE EAST'
    and ps_supplycost =
        select
            min(ps_supplycost)
        from
            partsupp, supplier, nation, region
        where
            p_partkey = ps_partkey and s_suppkey = ps_suppkey
            and s_nationkey = n_nationkey and n_regionkey = r_regionkey
            and r_name = 'MIDDLE EAST'
)
order by
    s_acctbal desc, n_name, s_name, p_partkey;
```

## Example 4

DBT-3 benchmark, Query #20

```

select
    s_name, s_address
from
    supplier, nation
where
    s_suppkey in (
        select
            distinct (ps_suppkey)
        from
            partsupp, part
        where
            ps_partkey=p_partkey
            and p_name like 'indian%'
            and ps_availqty > (
                select
                    0.5 * sum(l_quantity)
                from
                    lineitem
                where
                    l_partkey = ps_partkey
                    and l_suppkey = ps_suppkey
                    and l_shipdate >= '1995-01-01'
                    and l_shipdate < date_ADD('1995-01-01',interval 1 year)
            )
        )
        and s_nationkey = n_nationkey and n_name = 'JAPAN'
order by
    s_name;

```

## See Also

- [Query cache](#)
- <http://mysqlmaniac.com/2012/what-about-the-subqueries/> blog post describing impact of subquery cache optimization on queries used by DynamicPageList MediaWiki extension
- <http://varokism.blogspot.ru/2013/06/mariadb-subquery-cache-in-real-use-case.html> Another use case from the real world
- [What is MariaDB 5.3](#)

## 1.1.4.1.2.2.8.6 Condition Pushdown Into IN subqueries

This article describes Condition Pushdown into IN subqueries as implemented in [MDEV-12387](#).

`optimizer_switch` flag name:

```
condition_pushdown_for_subquery
```

## 1.1.4.1.2.2.8.7 Conversion of Big IN Predicates Into Subqueries

Starting from [MariaDB 10.3](#), the optimizer converts certain big IN predicates into IN subqueries.

That is, an IN predicate in the form

```
column [NOT] IN (const1, const2, .... )
```

is converted into an equivalent IN-subquery:

```
column [NOT] IN (select ... from temporary_table)
```

which opens new opportunities for the query optimizer.

The conversion happens if the following conditions are met:

- the IN list has more than 1000 elements (One can control it through the `in_predicate_conversion_threshold` parameter).
- the [NOT] IN condition is at the top level of the WHERE/ON clause.

## Controlling the Optimization

- The optimization is on by default. [MariaDB 10.3.18](#) (and debug builds prior to that) introduced the `in_predicate_conversion_threshold` variable. Set to

```
0
```

to disable the optimization.

# Benefits of the Optimization

If

column  
is a key-prefix, MariaDB optimizer will process the condition

```
column [NOT] IN (const1, const2, .... )
```

by trying to construct a range access. If the list is large, the analysis may take a lot of memory and CPU time. The problem gets worse when column is a part of a multi-column index and the query has conditions on other parts of the index.

Conversion of IN predicates into a subqueries bypass the range analysis, which means the query optimization phase will use less CPU and memory.

Possible disadvantages of the conversion are:

- The optimization may convert 'IN LIST elements' key accesses to a table scan (if there is no other usable index for the table)
- The estimates for the number of rows matching the  
IN (...)  
are less precise.

## See Also

- [IN operator](#)

## Links

<https://jira.mariadb.org/browse/MDEV-12176>

## 1.1.4.1.2.2.8.8 EXISTS-to-IN Optimization

### Contents

1. [Trivially-correlated EXISTS subqueries](#)
2. [Semi-join EXISTS subqueries](#)
3. [Handling of NULL values](#)
4. [Control](#)
5. [Limitations](#)

MySQL (including MySQL 5.6) has only one execution strategy for EXISTS subqueries. The strategy is essentially the straightforward, "naive" execution, without any rewrites.

MariaDB 5.3 introduced a rich set of optimizations for IN subqueries. Since then, it makes sense to convert an EXISTS subquery into an IN so that the new optimizations can be used.

EXISTS  
will be converted into  
IN  
in two cases:

1. Trivially correlated EXISTS subqueries
2. Semi-join EXISTS

We will now describe these two cases in detail

### Trivially-correlated EXISTS subqueries

Often, EXISTS subquery is correlated, but the correlation is trivial. The subquery has form

```
EXISTS (SELECT ... FROM ... WHERE outer_col= inner_col AND inner_where)
```

and "outer\_col" is the only place where the subquery refers to outside fields. In this case, the subquery can be re-written into uncorrelated IN:

```
outer_col IN (SELECT inner_col FROM ... WHERE inner_where)
```

(

NULL  
values require some special handling, see below). For uncorrelated IN subqueries, MariaDB is able a cost-based choice between two execution strategies:

- [IN-to-EXISTS](#) (basically, convert back into EXISTS)

- Materialization

That is, converting trivially-correlated

```
EXISTS
into uncorrelated
IN
gives query optimizer an option to use Materialization strategy for the subquery.
```

Currently, EXISTS->IN conversion works only for subqueries that are at top level of the WHERE clause, or are under NOT operation which is directly at top level of the WHERE clause.

## Semi-join EXISTS subqueries

If

```
EXISTS
subquery is an AND-part of the
WHERE
clause:
```

```
SELECT ... FROM outer_tables WHERE EXISTS (SELECT ...) AND ...
```

then it satisfies the main property of [semi-join subqueries](#) :

*with semi-join subquery, we're only interested in records of outer\_tables that have matches in the subquery*

Semi-join optimizer offers a rich set of execution strategies for both correlated and uncorrelated subqueries. The set includes FirstMatch strategy which is an equivalent of how EXISTS subqueries are executed, so we do not lose any opportunities when converting an EXISTS subquery into a semi-join.

In theory, it makes sense to convert all kinds of EXISTS subqueries: convert both correlated and uncorrelated ones, convert irrespectively of whether the subquery has inner=outer equality.

In practice, the subquery will be converted only if it has inner=outer equality. Both correlated and uncorrelated subqueries are converted.

## Handling of NULL values

TODO: rephrase this:

- IN has complicated NULL-semantics. NOT EXISTS doesn't.
- EXISTS-to-IN adds IS NOT NULL before the subquery predicate, when required

## Control

The optimization is controlled by the

```
exists_to_in
flag in optimizer_switch . Before MariaDB 10.0.12 , the optimization was OFF by default. Since MariaDB 10.0.12 , it has been ON by default.
```

## Limitations

EXISTS-to-IN doesn't handle

- subqueries that have GROUP BY, aggregate functions, or HAVING clause
- subqueries are UNIONs
- a number of degenerate edge cases

## 1.1.4.1.2.2.8.9 Optimizing GROUP BY and DISTINCT Clauses in Subqueries

A DISTINCT clause and a GROUP BY without a corresponding HAVING clause have no meaning in IN/ALL/ANY/SOME/EXISTS subqueries. The reason is that IN/ALL/ANY/SOME/EXISTS only check if an outer row satisfies some condition with respect to all or any row in the subquery result. Therefore it doesn't matter if the subquery has duplicate result rows or not - if some condition is true for some row of the subquery, this condition will be true for all duplicates of this row. Notice that GROUP BY without a corresponding HAVING clause is equivalent to a DISTINCT.

MariaDB 5.3 and later versions automatically remove DISTINCT and GROUP BY without HAVING if these clauses appear in an IN/ALL/ANY/SOME/EXISTS subquery. For instance:

```
select * from t1
where t1.a > ALL(select distinct b from t2 where t2.c > 100)
```

is transformed to:

```
select * from t1
where t1.a > ALL(select b from t2 where t2.c > 100)
```

Removing these unnecessary clauses allows the optimizer to find more efficient query plans because it doesn't need to take care of post-processing the subquery result to satisfy DISTINCT / GROUP BY.

## 1.1.4.1.2.2.9 Subqueries and JOINS

A [subquery](#) can quite often, but not in all cases, be rewritten as a [JOIN](#).

### Contents

1. [Rewriting Subqueries as JOINS](#)
2. [Using Subqueries instead of JOINS](#)

## Rewriting Subqueries as JOINS

A subquery using

IN  
can be rewritten with the  
DISTINCT  
keyword, for example:

```
SELECT * FROM table1 WHERE col1 IN (SELECT col1 FROM table2);
```

can be rewritten as:

```
SELECT DISTINCT table1.* FROM table1, table2 WHERE table1.col1=table2.col1;
```

NOT IN  
or  
NOT EXISTS

queries can also be rewritten. For example, these two queries returns the same result:

```
SELECT * FROM table1 WHERE col1 NOT IN (SELECT col1 FROM table2);
SELECT * FROM table1 WHERE NOT EXISTS (SELECT col1 FROM table2 WHERE table1.col1=table2.col1);
```

and both can be rewritten as:

```
SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.id=table2.id WHERE table2.id IS NULL;
```

Subqueries that can be rewritten as a LEFT JOIN are sometimes more efficient.

## Using Subqueries instead of JOINS

There are some scenarios, though, which call for subqueries rather than joins:

- When you want duplicates, but not false duplicates. Suppose

```
Table_1
has three rows —{
1
,
1
,
2
} — and
Table_2
has two rows —{
1
,
2
,
2
}. If you need to list the rows in
Table_1
which are also in
Table_2
, only this subquery-based
```

```
SELECT
statement will give the right answer (
1
,
1
,
2
):
```

```
SELECT Table_1.column_1
FROM Table_1
WHERE Table_1.column_1 IN
(SELECT Table_2.column_1
FROM Table_2);
```

This SQL statement won't work:

```
SELECT Table_1.column_1
FROM Table_1,Table_2
WHERE Table_1.column_1 = Table_2.column_1;
```

because the result will be {

```
1
,
1
,
2
,
2
```

} — and the duplication of 2 is an error. This SQL statement won't work either:

```
SELECT DISTINCT Table_1.column_1
FROM Table_1,Table_2
WHERE Table_1.column_1 = Table_2.column_1;
```

because the result will be {

```
1
,
2
```

} — and the removal of the duplicated 1 is an error too.

- When the outermost statement is not a query. The SQL statement:

```
UPDATE Table_1 SET column_1 = (SELECT column_1 FROM Table_2);
```

can't be expressed using a join unless some rare SQL3 features are used.

- When the join is over an expression. The SQL statement:

```
SELECT * FROM Table_1
WHERE column_1 + 5 =
(SELECT MAX(column_1) FROM Table_2);
```

is hard to express with a join. In fact, the only way we can think of is this SQL statement:

```
SELECT Table_1.*
FROM Table_1,
(SELECT MAX(column_1) AS max_column_1 FROM Table_2) AS Table_2
WHERE Table_1.column_1 + 5 = Table_2.max_column_1;
```

which still involves a parenthesized query, so nothing is gained from the transformation.

- When you want to see the exception. For example, suppose the question is: what books are longer than Das Kapital? These two queries are effectively almost the same:

```

SELECT DISTINCT Bookcolumn_1.*
FROM Books AS Bookcolumn_1 JOIN Books AS Bookcolumn_2 USING(page_count)
WHERE title = 'Das Kapital';

SELECT DISTINCT Bookcolumn_1.*
FROM Books AS Bookcolumn_1
WHERE Bookcolumn_1.page_count >
(SELECT DISTINCT page_count
FROM Books AS Bookcolumn_2
WHERE title = 'Das Kapital');

```

The difference is between these two SQL statements is, if there are two editions of *Das Kapital* (with different page counts), then the self-join example will return the books which are longer than the shortest edition of *Das Kapital*. That might be the wrong answer, since the original question didn't ask for "... longer than

ANY  
book named *Das Kapital*" (it seems to contain a false assumption that there's only one edition).

## 1.1.4.1.2.2.10 Subquery Limitations

### Contents

1. ORDER BY and LIMIT
2. Modifying and Selecting from the Same Table
3. Row Comparison Operations
4. Correlated Subqueries
5. Stored Functions

There are a number of limitations regarding **subqueries**, which are discussed below.

The following tables and data will be used in the examples that follow:

```

CREATE TABLE staff(name VARCHAR(10),age TINYINT);

CREATE TABLE customer(name VARCHAR(10),age TINYINT);

```

```

INSERT INTO staff VALUES
('Bilhah',37), ('Valerius',61), ('Maia',25);

INSERT INTO customer VALUES
('Thanasis',48), ('Valerius',61), ('Brion',51);

```

### ORDER BY and LIMIT

To use **ORDER BY** or limit **LIMIT** in **subqueries** both must be used.. For example:

```

SELECT * FROM staff WHERE name IN (SELECT name FROM customer ORDER BY name);
+-----+-----+
| name | age |
+-----+-----+
| Valerius | 61 |
+-----+-----+

```

is valid, but

```

SELECT * FROM staff WHERE name IN (SELECT NAME FROM customer ORDER BY name LIMIT 1);
ERROR 1235 (42000): This version of MariaDB doesn't
yet support 'LIMIT & IN/ALL/ANY/SOME subquery'

```

is not.

### Modifying and Selecting from the Same Table

It's not possible to both modify and select from the same table in a subquery. For example:

```
DELETE FROM staff WHERE name = (SELECT name FROM staff WHERE age=61);
ERROR 1093 (HY000): Table 'staff' is specified twice, both
as a target for 'DELETE' and as a separate source for data
```

## Row Comparison Operations

There is only partial support for row comparison operations. The expression in

```
expr op {ALL|ANY|SOME} subquery,
```

must be scalar and the subquery can only return a single column.

However, because of the way

```
IN  
is implemented (it is rewritten as a sequence of  
=  
comparisons and  
AND  
, the expression in
```

```
expression [NOT] IN subquery
```

is permitted to be an n-tuple and the subquery can return rows of n-tuples.

For example:

```
SELECT * FROM staff WHERE (name,age) NOT IN (
    SELECT name,age FROM customer WHERE age >=51]
);
+-----+-----+
| name | age |
+-----+-----+
| Bilhah | 37 |
| Maia | 25 |
+-----+-----+
```

is permitted, but

```
SELECT * FROM staff WHERE (name,age) = ALL (
    SELECT name,age FROM customer WHERE age >=51
);
ERROR 1241 (21000): Operand should contain 1 column(s)
```

is not.

## Correlated Subqueries

Subqueries in the FROM clause cannot be correlated subqueries. They cannot be evaluated for each row of the outer query since they are evaluated to produce a result set during when the query is executed.

## Stored Functions

A subquery can refer to a [stored function](#) which modifies data. This is an extension to the SQL standard, but can result in indeterminate outcomes. For example, take:

```
SELECT ... WHERE x IN (SELECT f() ...);
```

where *f()* inserts rows. The function *f()* could be executed a different number of times depending on how the optimizer chooses to handle the query.

This sort of construct is therefore not safe to use in replication that is not [row-based](#), as there could be different results on the master and the slave.

## 1.1.4.1.2.3 UNION

UNION

is used to combine the results from multiple [SELECT](#) statements into a single result set.

## Syntax

```
SELECT ...
UNION [ALL | DISTINCT] SELECT ...
[UNION [ALL | DISTINCT] SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [ALL/DISTINCT](#)
  2. [ORDER BY and LIMIT](#)
  3. [HIGH\\_PRIORITY](#)
  4. [SELECT ... INTO ...](#)
  5. [Parentheses](#)
3. [Examples](#)
4. [See Also](#)

## Description

### UNION

is used to combine the results from multiple [SELECT](#) statements into a single result set.

The column names from the first

`SELECT`

statement are used as the column names for the results returned. Selected columns listed in corresponding positions of each `SELECT` statement should have the same data type. (For example, the first column selected by the first statement should have the same type as the first column selected by the other statements.)

If they don't, the type and length of the columns in the result take into account the values returned by all of the `SELECT`s, so there is no need for explicit casting. Note that currently this is not the case for [recursive CTEs](#) - see [MDEV-12325](#).

Table names can be specified as

`db_name`

`tbl_name`

. This permits writing

`UNION`

s which involve multiple databases. See [Identifier Qualifiers](#) for syntax details.

`UNION` queries cannot be used with [aggregate functions](#).

`EXCEPT`

and

`UNION`

have the same operation precedence and

`INTERSECT`

has a higher precedence, unless [running in Oracle mode](#), in which case all three have the same precedence.

## ALL/DISTINCT

The

`ALL`

keyword causes duplicate rows to be preserved. The

`DISTINCT`

keyword (the default if the keyword is omitted) causes duplicate rows to be removed by the results.

`UNION ALL` and `UNION DISTINCT` can both be present in a query. In this case, `UNION DISTINCT` will override any `UNION ALLs` to its left.

MariaDB starting with [10.1.1](#)

Until [MariaDB 10.1.1](#), all

`UNION ALL`

statements required the server to create a temporary table. Since [MariaDB 10.1.1](#), the server can in most cases execute

`UNION ALL`

without creating a temporary table, improving performance (see [MDEV-334](#)).

## ORDER BY and LIMIT

Individual `SELECT`s can contain their own `ORDER BY` and `LIMIT` clauses. In this case, the individual queries need to be wrapped between parentheses.

However, this does not affect the order of the UNION, so they only are useful to limit the record read by one SELECT.

The UNION can have global ORDER BY and LIMIT clauses, which affect the whole resultset. If the columns retrieved by individual SELECT statements have an alias (AS), the ORDER BY must use that alias, not the real column names.

## HIGH\_PRIORITY

Specifying a query as HIGH\_PRIORITY will not work inside a UNION. If applied to the first SELECT, it will be ignored. Applying to a later SELECT results in a syntax error:

```
ERROR 1234 (42000): Incorrect usage/placement of 'HIGH_PRIORITY'
```

## SELECT ... INTO ...

Individual SELECTs cannot be written INTO DUMPFILE or INTO OUTFILE . If the last SELECT statement specifies INTO DUMPFILE or INTO OUTFILE, the entire result of the UNION will be written. Placing the clause after any other SELECT will result in a syntax error.

If the result is a single row, SELECT ... INTO @var\_name can also be used.

MariaDB starting with 10.4.0

## Parentheses

From MariaDB 10.4.0 , parentheses can be used to specify precedence. Before this, a syntax error would be returned.

## Examples

UNION  
between tables having different column names:

```
(SELECT e_name AS name, email FROM employees)
UNION
(SELECT c_name AS name, email FROM customers);
```

Specifying the

UNION

's global order and limiting total rows:

```
(SELECT name, email FROM employees)
UNION
(SELECT name, email FROM customers)
ORDER BY name LIMIT 10;
```

Adding a constant row:

```
(SELECT 'John Doe' AS name, 'john.doe@example.net' AS email)
UNION
(SELECT name, email FROM customers);
```

Differing types:

```
SELECT CAST('x' AS CHAR(1)) UNION SELECT REPEAT('y',4);
+-----+
| CAST('x' AS CHAR(1)) |
+-----+
| x
| yyyy
+-----+
```

Returning the results in order of each individual SELECT by use of a sort column:

```
(SELECT 1 AS sort_column, e_name AS name, email FROM employees)
UNION
(SELECT 2, c_name AS name, email FROM customers) ORDER BY sort_column;
```

Difference between UNION, EXCEPT and INTERSECT .

INTERSECT ALL

and  
EXCEPT ALL  
are available from [MariaDB 10.5.0](#).

```
CREATE TABLE seqs (i INT);
INSERT INTO seqs VALUES (1),(2),(3),(3),(4),(5),(6);

SELECT i FROM seqs WHERE i <= 3 UNION SELECT i FROM seqs WHERE i>=3;
+---+
| i |
+---+
| 1 |
| 2 |
| 3 |
| 3 |
| 4 |
| 5 |
| 6 |
+---+

SELECT i FROM seqs WHERE i <= 3 UNION ALL SELECT i FROM seqs WHERE i>=3;
+---+
| i |
+---+
| 1 |
| 2 |
| 2 |
| 3 |
| 3 |
| 3 |
| 3 |
| 4 |
| 5 |
| 6 |
+---+

SELECT i FROM seqs WHERE i <= 3 EXCEPT SELECT i FROM seqs WHERE i>=3;
+---+
| i |
+---+
| 1 |
| 2 |
+---+

SELECT i FROM seqs WHERE i <= 3 EXCEPT ALL SELECT i FROM seqs WHERE i>=3;
+---+
| i |
+---+
| 1 |
| 2 |
| 2 |
+---+

SELECT i FROM seqs WHERE i <= 3 INTERSECT SELECT i FROM seqs WHERE i>=3;
+---+
| i |
+---+
| 3 |
+---+

SELECT i FROM seqs WHERE i <= 3 INTERSECT ALL SELECT i FROM seqs WHERE i>=3;
+---+
| i |
+---+
| 3 |
| 3 |
+---+
```

Parentheses for specifying precedence, from [MariaDB 10.4.0](#)

```

CREATE OR REPLACE TABLE t1 (a INT);
CREATE OR REPLACE TABLE t2 (b INT);
CREATE OR REPLACE TABLE t3 (c INT);

INSERT INTO t1 VALUES (1),(2),(3),(4);
INSERT INTO t2 VALUES (5),(6);
INSERT INTO t3 VALUES (1),(6);

((SELECT a FROM t1) UNION (SELECT b FROM t2)) INTERSECT (SELECT c FROM t3);
+-----+
| a   |
+-----+
| 1   |
| 6   |
+-----+

(SELECT a FROM t1) UNION ((SELECT b FROM t2) INTERSECT (SELECT c FROM t3));
+-----+
| a   |
+-----+
| 1   |
| 2   |
| 3   |
| 4   |
| 6   |
+-----+

```

## See Also

- [SELECT](#)
- [EXCEPT](#)
- [INTERSECT](#)
- [Recursive Common Table Expressions Overview](#)
- [Get Set for Set Theory: UNION, INTERSECT and EXCEPT in SQL](#) (video tutorial)

## 1.1.4.1.2.4 EXCEPT

MariaDB starting with [10.3.0](#)

EXCEPT was introduced in [MariaDB 10.3.0](#).

The result of

```

EXCEPT
is all records of the left
SELECT
result set except records which are in right
SELECT
result set, i.e. it is subtraction of two result sets. From MariaDB 10.6.1,
MINUS
is a synonym.

```

## Syntax

```

SELECT ...
  (INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL | DISTINCT]) SELECT ...
  [(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL | DISTINCT]) SELECT ...]
  [ORDER BY [column [, column ...]]]
  [LIMIT {[offset,] row_count | row_count OFFSET offset}]

```

## Contents

1. [Syntax](#)
  1. [Description](#)
  2. [Parentheses](#)
  3. [ALL/DISTINCT](#)
2. [Examples](#)
3. [See Also](#)

Please note:

- Brackets for explicit operation precedence are not supported; use a subquery in the `FROM` clause as a workaround).

## Description

MariaDB has supported

`EXCEPT`  
and `INTERSECT` in addition to `UNION` since [MariaDB 10.3](#).

All behavior for naming columns,

`ORDER BY`  
and  
`LIMIT`  
is the same as for  
`UNION`

`EXCEPT`  
implicitly supposes a  
`DISTINCT`  
operation.

The result of

`EXCEPT`  
is all records of the left  
`SELECT`  
result except records which are in right  
`SELECT`  
result set, i.e. it is subtraction of two result sets.

`EXCEPT`  
and  
`UNION`  
have the same operation precedence and  
`INTERSECT`  
has a higher precedence, unless [running in Oracle mode](#), in which case all three have the same precedence.

MariaDB starting with [10.4.0](#)

### Parentheses

From [MariaDB 10.4.0](#), parentheses can be used to specify precedence. Before this, a syntax error would be returned.

MariaDB starting with [10.5.0](#)

### ALL/DISTINCT

`EXCEPT ALL`  
and  
`EXCEPT DISTINCT`  
were introduced in [MariaDB 10.5.0](#). The  
`ALL`  
operator leaves duplicates intact, while the  
`DISTINCT`  
operator removes duplicates.  
`DISTINCT`  
is the default behavior if neither operator is supplied, and the only behavior prior to [MariaDB 10.5](#).

## Examples

Show customers which are not employees:

```
(SELECT e_name AS name, email FROM customers)
EXCEPT
(SELECT c_name AS name, email FROM employees);
```

Difference between **UNION**, **EXCEPT** and **INTERSECT**.

INTERSECT ALL  
and  
EXCEPT ALL  
are available from [MariaDB 10.5.0](#).

```
CREATE TABLE seqs (i INT);
INSERT INTO seqs VALUES (1),(2),(3),(4),(5),(6);

SELECT i FROM seqs WHERE i <= 3 UNION SELECT i FROM seqs WHERE i>=3;
```

```
+----+
| i |
+----+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
+----+
```

```
SELECT i FROM seqs WHERE i <= 3 UNION ALL SELECT i FROM seqs WHERE i>=3;
```

```
+----+
| i |
+----+
| 1 |
| 2 |
| 2 |
| 3 |
| 3 |
| 3 |
| 3 |
| 4 |
| 5 |
| 6 |
+----+
```

```
SELECT i FROM seqs WHERE i <= 3 EXCEPT SELECT i FROM seqs WHERE i>=3;
```

```
+----+
| i |
+----+
| 1 |
| 2 |
+----+
```

```
SELECT i FROM seqs WHERE i <= 3 EXCEPT ALL SELECT i FROM seqs WHERE i>=3;
```

```
+----+
| i |
+----+
| 1 |
| 2 |
| 2 |
+----+
```

```
SELECT i FROM seqs WHERE i <= 3 INTERSECT SELECT i FROM seqs WHERE i>=3;
```

```
+----+
| i |
+----+
| 3 |
+----+
```

```
SELECT i FROM seqs WHERE i <= 3 INTERSECT ALL SELECT i FROM seqs WHERE i>=3;
```

```
+----+
| i |
+----+
| 3 |
| 3 |
+----+
```

```

CREATE OR REPLACE TABLE t1 (a INT);
CREATE OR REPLACE TABLE t2 (b INT);
CREATE OR REPLACE TABLE t3 (c INT);

INSERT INTO t1 VALUES (1),(2),(3),(4);
INSERT INTO t2 VALUES (5),(6);
INSERT INTO t3 VALUES (1),(6);

((SELECT a FROM t1) UNION (SELECT b FROM t2)) EXCEPT (SELECT c FROM t3);
+---+
| a |
+---+
| 2 |
| 3 |
| 4 |
| 5 |
+---+

(SELECT a FROM t1) UNION ((SELECT b FROM t2) EXCEPT (SELECT c FROM t3));
+---+
| a |
+---+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+---+

```

## See Also

- [UNION](#)
- [INTERSECT](#)
- [Get Set for Set Theory: UNION, INTERSECT and EXCEPT in SQL](#) (video tutorial)

## 1.1.4.1.2.5 INTERSECT

MariaDB starting with [10.3.0](#)

INTERSECT was introduced in [MariaDB 10.3.0](#).

The result of an intersect is the intersection of right and left

```

SELECT ...
results, i.e. only records that are present in both result sets will be included in the result of the operation.
```

## Syntax

```

SELECT ...
(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL | DISTINCT]) SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] | UNION [ALL | DISTINCT]) SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

## Contents

1. [Syntax](#)
  1. [Description](#)
    1. [Parentheses](#)
    2. [ALL/DISTINCT](#)
  2. [Examples](#)
  3. [See Also](#)

## Description

MariaDB has supported

`INTERSECT`  
(as well as `EXCEPT`) in addition to `UNION` since [MariaDB 10.3](#).

All behavior for naming columns,

`ORDER BY`

and  
LIMIT  
is the same as for [UNION](#) .

INTERSECT  
implicitly supposes a  
DISTINCT  
operation.

The result of an intersect is the intersection of right and left

SELECT  
results, i.e. only records that are present in both result sets will be included in the result of the operation.

INTERSECT  
has higher precedence than  
UNION  
and  
EXCEPT  
(unless running [running in Oracle mode](#), in which case all three have the same precedence). If possible it will be executed linearly but if not it will be translated to a subquery in the  
FROM  
clause:

```
(select a,b from t1)
union
(select c,d from t2)
intersect
(select e,f from t3)
union
(select 4,4);
```

will be translated to:

```
(select a,b from t1)
union
select c,d from
((select c,d from t2)
 intersect
 (select e,f from t3)) dummy_subselect
union
(select 4,4)
```

MariaDB starting with [10.4.0](#)

## Parentheses

From [MariaDB 10.4.0](#), parentheses can be used to specify precedence. Before this, a syntax error would be returned.

MariaDB starting with [10.5.0](#)

## ALL/DISTINCT

INTERSECT ALL  
and  
INTERSECT DISTINCT  
were introduced in [MariaDB 10.5.0](#). The  
ALL  
operator leaves duplicates intact, while the  
DISTINCT  
operator removes duplicates.  
DISTINCT  
is the default behavior if neither operator is supplied, and the only behavior prior to [MariaDB 10.5](#).

## Examples

Show customers which are employees:

```
(SELECT e_name AS name, email FROM employees)
INTERSECT
(SELECT c_name AS name, email FROM customers);
```

Difference between UNION , EXCEPT and INTERSECT.

INTERSECT ALL

and

EXCEPT ALL

are available from MariaDB 10.5.0 .

```
CREATE TABLE seqs (i INT);
INSERT INTO seqs VALUES (1),(2),(2),(3),(3),(4),(5),(6);

SELECT i FROM seqs WHERE i <= 3 UNION SELECT i FROM seqs WHERE i>=3;
+----+
| i |
+----+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
+----+

SELECT i FROM seqs WHERE i <= 3 UNION ALL SELECT i FROM seqs WHERE i>=3;
+----+
| i |
+----+
| 1 |
| 2 |
| 2 |
| 3 |
| 3 |
| 3 |
| 3 |
| 4 |
| 5 |
| 6 |
+----+

SELECT i FROM seqs WHERE i <= 3 EXCEPT SELECT i FROM seqs WHERE i>=3;
+----+
| i |
+----+
| 1 |
| 2 |
+----+

SELECT i FROM seqs WHERE i <= 3 EXCEPT ALL SELECT i FROM seqs WHERE i>=3;
+----+
| i |
+----+
| 1 |
| 2 |
| 2 |
+----+

SELECT i FROM seqs WHERE i <= 3 INTERSECT SELECT i FROM seqs WHERE i>=3;
+----+
| i |
+----+
| 3 |
+----+

SELECT i FROM seqs WHERE i <= 3 INTERSECT ALL SELECT i FROM seqs WHERE i>=3;
+----+
| i |
+----+
| 3 |
| 3 |
+----+
```

```
CREATE OR REPLACE TABLE t1 (a INT);
CREATE OR REPLACE TABLE t2 (b INT);
CREATE OR REPLACE TABLE t3 (c INT);

INSERT INTO t1 VALUES (1),(2),(3),(4);
INSERT INTO t2 VALUES (5),(6);
INSERT INTO t3 VALUES (1),(6);

((SELECT a FROM t1) UNION (SELECT b FROM t2)) INTERSECT (SELECT c FROM t3);
+---+
| a |
+---+
| 1 |
| 6 |
+---+

(SELECT a FROM t1) UNION ((SELECT b FROM t2) INTERSECT (SELECT c FROM t3));
+---+
| a |
+---+
| 1 |
| 2 |
| 3 |
| 4 |
| 6 |
+---+
```

## See Also

- [UNION](#)
- [EXCEPT](#)
- [Get Set for Set Theory: UNION, INTERSECT and EXCEPT in SQL](#) (video tutorial)

## 1.1.4.1.2.6 Precedence Control in Table Operations

MariaDB starting with [10.4.0](#)

Beginning in [MariaDB 10.4](#), you can control the ordering of execution on table operations using parentheses.

## Syntax

```
( expression )
[ORDER BY [column[, column...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)

## Description

Using parentheses in your SQL allows you to control the order of execution for

`SELECT`

statements and [Table Value Constructor](#), including

`UNION`

,

`EXCEPT`

, and

## INTERSECT

operations. MariaDB executes the parenthetical expression before the rest of the statement. You can then use

## ORDER BY

and

## LIMIT

clauses to further organize the result-set.

**Note :** In practice, the Optimizer may rearrange the exact order in which MariaDB executes different parts of the statement. When it calculates the result-set, however, it returns values as though the parenthetical expression were executed first.

## Example

```
CREATE TABLE test.t1 (num INT);

INSERT INTO test.t1 VALUES (1),(2),(3);

(SELECT * FROM test.t1
UNION
VALUES (10))
INTERSECT
VALUES (1),(3),(10),(11);
+-----+
| num |
+-----+
|   1 |
|   3 |
|  10 |
+-----+

((SELECT * FROM test.t1
UNION
VALUES (10))
INTERSECT
VALUES (1),(3),(10),(11))
ORDER BY 1 DESC;
+-----+
| num |
+-----+
|  10 |
|   3 |
|    1 |
+-----+
```

## 1.1.4.1.2.7 MINUS

MariaDB starting with 10.6.1

### MINUS

was introduced as a synonym for EXCEPT from MariaDB 10.6.1 .

## 1.1.4.1.3 LIMIT

### Contents

1. [Description](#)
  1. [Multi-Table Updates](#)
  2. [GROUP\\_CONCAT](#)
2. [Examples](#)
3. [See Also](#)

## Description

Use the

```
LIMIT
clause to restrict the number of returned rows. When you use a single integer n with
LIMIT
, the first n rows will be returned. Use the ORDER BY clause to control which rows come first. You can also select a number of rows after an offset using either of the following:
```

```
LIMIT offset, row_count
LIMIT row_count OFFSET offset
```

When you provide an offset *m* with a limit *n* , the first *m* rows will be ignored, and the following *n* rows will be returned.

Executing an UPDATE with the

```
LIMIT
clause is not safe for replication.
LIMIT 0
is an exception to this rule (see MDEV-6170 ).
```

There is a LIMIT ROWS EXAMINED optimization which provides the means to terminate the execution of SELECT statements which examine too many rows, and thus use too many resources. See LIMIT ROWS EXAMINED .

## Multi-Table Updates

MariaDB starting with 10.3.2

Until MariaDB 10.3.1 , it was not possible to use

```
LIMIT
(or ORDER BY ) in a multi-table UPDATE statement. This restriction was lifted in MariaDB 10.3.2 .
```

## GROUP\_CONCAT

MariaDB starting with 10.3.2

Starting from MariaDB 10.3.3 , it is possible to use

```
LIMIT
with GROUP_CONCAT() .
```

## Examples

```
CREATE TABLE members (name VARCHAR(20));
INSERT INTO members VALUES('Jagdish'),('Kenny'),('Rokurou'),('Immaculada');

SELECT * FROM members;
+-----+
| name   |
+-----+
| Jagdish |
| Kenny  |
| Rokurou|
| Immaculada |
+-----+
```

Select the first two names (no ordering specified):

```
SELECT * FROM members LIMIT 2;
+-----+
| name   |
+-----+
| Jagdish |
| Kenny  |
+-----+
```

All the names in alphabetical order:

```
SELECT * FROM members ORDER BY name;
```

```
+-----+
| name      |
+-----+
| Immaculada |
| Jagdish    |
| Kenny      |
| Rokurou    |
+-----+
```

The first two names, ordered alphabetically:

```
SELECT * FROM members ORDER BY name LIMIT 2;
```

```
+-----+
| name      |
+-----+
| Immaculada |
| Jagdish    |
+-----+
```

The third name, ordered alphabetically (the first name would be offset zero, so the third is offset two):

```
SELECT * FROM members ORDER BY name LIMIT 2,1;
```

```
+-----+
| name      |
+-----+
| Kenny     |
+-----+
```

From MariaDB 10.3.2 ,

LIMIT

can be used in a multi-table update:

```
CREATE TABLE warehouse (product_id INT, qty INT);
INSERT INTO warehouse VALUES (1,100),(2,100),(3,100),(4,100);

CREATE TABLE store (product_id INT, qty INT);
INSERT INTO store VALUES (1,5),(2,5),(3,5),(4,5);

UPDATE warehouse,store SET warehouse.qty = warehouse.qty-2, store.qty = store.qty+2
WHERE (warehouse.product_id = store.product_id AND store.product_id  >= 1)
ORDER BY store.product_id DESC LIMIT 2;
```

```
SELECT * FROM warehouse;
```

```
+-----+
| product_id | qty   |
+-----+
|          1 |  100 |
|          2 |  100 |
|          3 |   98 |
|          4 |   98 |
+-----+
```

```
SELECT * FROM store;
```

```
+-----+
| product_id | qty   |
+-----+
|          1 |    5 |
|          2 |    5 |
|          3 |    7 |
|          4 |    7 |
+-----+
```

From MariaDB 10.3.3 ,

LIMIT

can be used with GROUP\_CONCAT , so, for example, given the following table:

```
CREATE TABLE d (dd DATE, cc INT);
```

```
INSERT INTO d VALUES ('2017-01-01',1);
INSERT INTO d VALUES ('2017-01-02',2);
INSERT INTO d VALUES ('2017-01-04',3);
```

the following query:

```
SELECT SUBSTRING_INDEX(GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC),",",1) FROM d;
+-----+
| SUBSTRING_INDEX(GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC),",",1) |
+-----+
| 2017-01-04:3 |
+-----+
```

can be more simply rewritten as:

```
SELECT GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC LIMIT 1) FROM d;
+-----+
| GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC LIMIT 1) |
+-----+
| 2017-01-04:3 |
+-----+
```

## See Also

- [ROWNUM\(\) function](#)
- [SELECT](#)
- [UPDATE](#)
- [DELETE](#)
- [Joins and Subqueries](#)
- [ORDER BY](#)
- [GROUP BY](#)
- [Common Table Expressions](#)
- [SELECT WITH ROLLUP](#)
- [SELECT INTO OUTFILE](#)
- [SELECT INTO DUMPFILE](#)
- [FOR UPDATE](#)
- [LOCK IN SHARE MODE](#)
- [Optimizer Hints](#)
- [SELECT ... OFFSET ... FETCH](#)

## 1.1.4.1.4 ORDER BY

### Contents

1. [Description](#)
2. [Examples](#)
3. [See Also](#)

## Description

Use the

ORDER BY

clause to order a resultset, such as that are returned from a [SELECT](#) statement. You can specify just a column or use any expression with functions. If you are using the

GROUP BY

clause, you can use grouping functions in

ORDER BY

. Ordering is done after grouping.

You can use multiple ordering expressions, separated by commas. Rows will be sorted by the first expression, then by the second expression if they have the same value for the first, and so on.

You can use the keywords

ASC

and

DESC

after each ordering expression to force that ordering to be ascending or descending, respectively. Ordering is ascending by default.

You can also use a single integer as the ordering expression. If you use an integer  $n$ , the results will be ordered by the  $n$ th column in the select expression.

When string values are compared, they are compared as if by the [STRCMP](#) function.

STRCMP

ignores trailing whitespace and may normalize characters and ignore case, depending on the [collation](#) in use.

Duplicated entries in the

ORDER BY

clause are removed.

ORDER BY

can also be used to order the activities of a [DELETE](#) or [UPDATE](#) statement (usually with the [LIMIT](#) clause).

MariaDB starting with [10.3.2](#)

Until [MariaDB 10.3.1](#), it was not possible to use

ORDER BY

(or [LIMIT](#)) in a multi-table [UPDATE](#) statement. This restriction was lifted in [MariaDB 10.3.2](#).

MariaDB starting with [10.5](#)

From [MariaDB 10.5](#), MariaDB allows packed sort keys and values of non-sorted fields in the sort buffer. This can make filesort temporary files much smaller when VARCHAR, CHAR or BLOBS are used, notably speeding up some ORDER BY sorts.

## Examples

```
CREATE TABLE seq (i INT, x VARCHAR(1));
INSERT INTO seq VALUES (1,'a'), (2,'b'), (3,'b'), (4,'f'), (5,'e');

SELECT * FROM seq ORDER BY i;
+---+---+
| i | x |
+---+---+
| 1 | a |
| 2 | b |
| 3 | b |
| 4 | f |
| 5 | e |
+---+---+

SELECT * FROM seq ORDER BY i DESC;
+---+---+
| i | x |
+---+---+
| 5 | e |
| 4 | f |
| 3 | b |
| 2 | b |
| 1 | a |
+---+---+

SELECT * FROM seq ORDER BY x,i;
+---+---+
| i | x |
+---+---+
| 1 | a |
| 2 | b |
| 3 | b |
| 5 | e |
| 4 | f |
+---+---+
```

ORDER BY in an [UPDATE](#) statement, in conjunction with [LIMIT](#):

```
UPDATE seq SET x='z' WHERE x='b' ORDER BY i DESC LIMIT 1;

SELECT * FROM seq;
+---+---+
| i | x |
+---+---+
| 1 | a |
| 2 | b |
| 3 | z |
| 4 | f |
| 5 | e |
+---+---+
```

From [MariaDB 10.3.2](#),

ORDER BY

can be used in a multi-table update:

```
CREATE TABLE warehouse (product_id INT, qty INT);
INSERT INTO warehouse VALUES (1,100),(2,100),(3,100),(4,100);

CREATE TABLE store (product_id INT, qty INT);
INSERT INTO store VALUES (1,5),(2,5),(3,5),(4,5);

UPDATE warehouse,store SET warehouse.qty = warehouse.qty-2, store.qty = store.qty+2
WHERE (warehouse.product_id = store.product_id AND store.product_id >= 1)
ORDER BY store.product_id DESC LIMIT 2;

SELECT * FROM warehouse;
+-----+-----+
| product_id | qty |
+-----+-----+
| 1 | 100 |
| 2 | 100 |
| 3 | 98 |
| 4 | 98 |
+-----+-----+

SELECT * FROM store;
+-----+-----+
| product_id | qty |
+-----+-----+
| 1 | 5 |
| 2 | 5 |
| 3 | 7 |
| 4 | 7 |
+-----+-----+
```

## See Also

- Why is ORDER BY in a FROM subquery ignored?
- SELECT
- UPDATE
- DELETE
- Improvements to ORDER BY Optimization
- Joins and Subqueries
- LIMIT
- GROUP BY
- Common Table Expressions
- SELECT WITH ROLLUP
- SELECT INTO OUTFILE
- SELECT INTO DUMPFILE
- FOR UPDATE
- LOCK IN SHARE MODE
- Optimizer Hints

## 1.1.4.1.5 GROUP BY

### Contents

1. WITH ROLLUP
2. GROUP BY Examples
3. See Also

Use the

GROUP BY

clause in a **SELECT** statement to group rows together that have the same value in one or more column, or the same computed value using expressions with any **functions and operators** except **grouping functions**. When you use a

GROUP BY

clause, you will get a single result row for each group of rows that have the same value for the expression given in

GROUP BY

When grouping rows, grouping values are compared as if by the

=

operator. For string values, the

=

operator ignores trailing whitespace and may normalize characters and ignore case, depending on the [collation](#) in use.

You can use any of the grouping functions in your select expression. Their values will be calculated based on all the rows that have been grouped together for each result row. If you select a non-grouped column or a value computed from a non-grouped column, it is undefined which row the returned value is taken from. This is not permitted if the

`ONLY_FULL_GROUP_BY`  
`SQL_MODE` is used.

You can use multiple expressions in the

`GROUP BY`  
clause, separated by commas. Rows are grouped together if they match on each of the expressions.

You can also use a single integer as the grouping expression. If you use an integer  $n$ , the results will be grouped by the  $n$ th column in the select expression.

The

`WHERE`  
clause is applied before the  
`GROUP BY`  
clause. It filters non-aggregated rows before the rows are grouped together. To filter grouped rows based on aggregate values, use the  
`HAVING`  
clause. The  
`HAVING`  
clause takes any expression and evaluates it as a boolean, just like the  
`WHERE`  
clause. You can use grouping functions in the  
`HAVING`  
clause. As with the select expression, if you reference non-grouped columns in the  
`HAVING`  
clause, the behavior is undefined.

By default, if a

`GROUP BY`  
clause is present, the rows in the output will be sorted by the expressions used in the  
`GROUP BY`  
. You can also specify  
`ASC`  
or  
`DESC`  
(ascending, descending) after those expressions, like in [ORDER BY](#). The default is  
`ASC`  
.

If you want the rows to be sorted by another field, you can add an explicit [ORDER BY](#). If you don't want the result to be ordered, you can add [ORDER BY NULL](#).

## WITH ROLLUP

The

`WITH ROLLUP`  
modifier adds extra rows to the resultset that represent super-aggregate summaries. For a full description with examples, see [SELECT WITH ROLLUP](#).

## GROUP BY Examples

Consider the following table that records how many times each user has played and won a game:

```
CREATE TABLE plays (name VARCHAR(16), plays INT, wins INT);
INSERT INTO plays VALUES
("John", 20, 5),
("Robert", 22, 8),
("Wanda", 32, 8),
("Susan", 17, 3);
```

Get a list of win counts along with a count:

```

SELECT wins, COUNT(*) FROM plays GROUP BY wins;
+-----+
| wins | COUNT(*) |
+-----+
|   3 |      1 |
|   5 |      1 |
|   8 |      2 |
+-----+
3 rows in set (0.00 sec)

```

The

GROUP BY  
expression can be a computed value, and can refer back to an identifier specified with  
AS  
. Get a list of win averages along with a count:

```

SELECT (wins / plays) AS winavg, COUNT(*) FROM plays GROUP BY winavg;
+-----+
| winavg | COUNT(*) |
+-----+
| 0.1765 |      1 |
| 0.2500 |      2 |
| 0.3636 |      1 |
+-----+
3 rows in set (0.00 sec)

```

You can use any [grouping function](#) in the select expression. For each win average as above, get a list of the average play count taken to get that average:

```

SELECT (wins / plays) AS winavg, AVG(plays) FROM plays
  GROUP BY winavg;
+-----+
| winavg | AVG(plays) |
+-----+
| 0.1765 |    17.0000 |
| 0.2500 |    26.0000 |
| 0.3636 |    22.0000 |
+-----+
3 rows in set (0.00 sec)

```

You can filter on aggregate information using the

HAVING  
clause. The  
HAVING  
clause is applied after  
GROUP BY  
and allows you to filter on aggregate data that is not available to the  
WHERE  
clause. Restrict the above example to results that involve an average number of plays over 20:

```

SELECT (wins / plays) AS winavg, AVG(plays) FROM plays
  GROUP BY winavg HAVING AVG(plays) > 20;
+-----+
| winavg | AVG(plays) |
+-----+
| 0.2500 |    26.0000 |
| 0.3636 |    22.0000 |
+-----+
2 rows in set (0.00 sec)

```

## See Also

- [SELECT](#)
- [Joins and Subqueries](#)
- [LIMIT](#)
- [ORDER BY](#)
- [Common Table Expressions](#)
- [SELECT WITH ROLLUP](#)
- [SELECT INTO OUTFILE](#)
- [SELECT INTO DUMPFILE](#)
- [FOR UPDATE](#)
- [LOCK IN SHARE MODE](#)

- Optimizer Hints

## 1.1.4.1.6 Common Table Expressions

### 1.1.4.1.6.1 WITH

MariaDB starting with [10.2.1](#)

Common Table Expressions were introduced in [MariaDB 10.2.1](#).

#### Syntax

```
WITH [RECURSIVE] table_reference [(columns_list)] AS (
    SELECT ...
)
[CYCLE cycle_column_list RESTRICT]
SELECT ...
```

#### Contents

1. [Syntax](#)
2. [Description](#)
  1. [CYCLE ... RESTRICT](#)
3. [Examples](#)
4. [See Also](#)

#### Description

The

`WITH`

keyword signifies a [Common Table Expression](#) (CTE). It allows you to refer to a subquery expression many times in a query, as if having a temporary table that only exists for the duration of a query.

There are two kinds of CTEs:

- [Non-Recursive](#)
- [Recursive](#) (signified by the  
`RECURSIVE`  
keyword, supported since [MariaDB 10.2.2](#))

You can use

```
table_reference
as any normal table in the external
SELECT
part. You can also use
WITH
in subqueries, as well as with EXPLAIN and SELECT.
```

Poorly-formed recursive CTEs can in theory cause infinite loops. The `max_recursive_iterations` system variable limits the number of recursions.

#### CYCLE ... RESTRICT

MariaDB starting with [10.5.2](#)

The CYCLE clause enables CTE cycle detection, avoiding excessive or infinite loops, MariaDB supports a relaxed, non-standard grammar.

The SQL Standard permits a CYCLE clause, as follows:

```
WITH RECURSIVE ... (
...
)
CYCLE <cycle column list>
SET <cycle mark column> TO <cycle mark value> DEFAULT <non-cycle mark value>
USING <path column>
```

where all clauses are mandatory.

MariaDB does not support this, but from 10.5.2 permits a non-standard relaxed grammar, as follows:

```
WITH RECURSIVE ... (
...
)
CYCLE <cycle column list> RESTRICT
```

With the use of  
 CYCLE ... RESTRICT  
 it makes no difference whether the CTE uses  
 UNION ALL  
 or  
 UNION DISTINCT  
 anymore.  
 UNION ALL  
 means "all rows, but without cycles", which is exactly what the  
 CYCLE  
 clause enables. And  
 UNION DISTINCT  
 means all rows should be different, which, again, is what will happen — as uniqueness is enforced over a subset of columns, complete rows  
 will automatically all be different.

## Examples

Below is an example with the

WITH

at the top level:

```
WITH t AS (SELECT a FROM t1 WHERE b >= 'c')
SELECT * FROM t2, t WHERE t2.c = t.a;
```

The example below uses

WITH

in a subquery:

```
SELECT t1.a, t1.b FROM t1, t2
WHERE t1.a > t2.c
AND t2.c IN(WITH t AS (SELECT * FROM t1 WHERE t1.a < 5)
            SELECT t2.c FROM t2, t WHERE t2.c = t.a);
```

Below is an example of a Recursive CTE:

```
WITH RECURSIVE ancestors AS
( SELECT * FROM folks
  WHERE name='Alex'
UNION
  SELECT f./*
    FROM folks AS f, ancestors AS a
   WHERE f.id = a.father OR f.id = a.mother )
SELECT * FROM ancestors;
```

Take the following structure, and data,

```
CREATE TABLE t1 (from_ int, to_ int);
INSERT INTO t1 VALUES (1,2), (1,100), (2,3), (3,4), (4,1);
SELECT * FROM t1;
+-----+-----+
| from_ | to_  |
+-----+-----+
|     1 |     2 |
|     1 |    100|
|     2 |     3 |
|     3 |     4 |
|     4 |     1 |
+-----+-----+
```

Given the above, the following query would theoretically result in an infinite loop due to the last record in t1 (note that `max_recursive_iterations` is set to 10 for the purposes of this example, to avoid the excessive number of cycles):

```

SET max_recursive_iterations=10;

WITH RECURSIVE cte (depth, from_, to_) AS (
  SELECT 0,1,1 UNION DISTINCT SELECT depth+1, t1.from_, t1.to_
    FROM t1, cte WHERE t1.from_ = cte.to_
)
SELECT * FROM cte;
+-----+-----+-----+
| depth | from_ | to_ |
+-----+-----+-----+
|    0 |     1 |    1 |
|    1 |     1 |    2 |
|    1 |     1 |   100 |
|    2 |     2 |    3 |
|    3 |     3 |    4 |
|    4 |     4 |    1 |
|    5 |     1 |    2 |
|    5 |     1 |   100 |
|    6 |     2 |    3 |
|    7 |     3 |    4 |
|    8 |     4 |    1 |
|    9 |     1 |    2 |
|    9 |     1 |   100 |
|   10 |     2 |    3 |
+-----+-----+-----+

```

However, the CYCLE ... RESTRICT clause (from [MariaDB 10.5.2](#) ) can overcome this:

```

WITH RECURSIVE cte (depth, from_, to_) AS (
  SELECT 0,1,1 UNION SELECT depth+1, t1.from_, t1.to_
    FROM t1, cte WHERE t1.from_ = cte.to_
)
CYCLE from_, to_ RESTRICT
SELECT * FROM cte;
+-----+-----+-----+
| depth | from_ | to_ |
+-----+-----+-----+
|    0 |     1 |    1 |
|    1 |     1 |    2 |
|    1 |     1 |   100 |
|    2 |     2 |    3 |
|    3 |     3 |    4 |
|    4 |     4 |    1 |
+-----+-----+-----+

```

## See Also

- [Non-Recursive Common Table Expressions Overview](#)
- [Recursive Common Table Expressions Overview](#)

## 1.1.4.1.6.2 Non-Recursive Common Table Expressions Overview

### Contents

1. [Non-Recursive CTEs](#)
  1. [A CTE referencing Another CTE](#)
  2. [Multiple Uses of a CTE](#)

Common Table Expressions (CTEs) are a standard SQL feature, and are essentially temporary named result sets. There are two kinds of CTEs: Non-Recursive, which this article covers; and [Recursive](#) .

MariaDB starting with [10.2.1](#)

Common table expressions were introduced in [MariaDB 10.2.1](#) .

## Non-Recursive CTEs

The `WITH` keyword signifies a CTE. It is given a name, followed by a body (the main query) as follows:

```

WITH          CTE name
with engineers as (
    select *
    from employees
    where dept='Engineering'
)
select *
from engineers      ← CTE Usage
where ...

```

CTEs are similar to derived tables. For example

```

WITH engineers AS
( SELECT * FROM employees
  WHERE dept = 'Engineering' )

SELECT * FROM engineers
WHERE ...

```

```

SELECT * FROM
( SELECT * FROM employees
  WHERE dept = 'Engineering' ) AS engineers
WHERE
...

```

A non-recursive CTE is basically a query-local [VIEW](#). There are several advantages and caveats to them. The syntax is more readable than nested  
`FROM (SELECT ...)`  
. A CTE can refer to another and it can be referenced from multiple places.

## A CTE referencing Another CTE

Using this format makes for a more readable SQL than a nested

`FROM(SELECT ...)`

clause. Below is an example of this:

```

WITH engineers AS (
  SELECT * FROM employees
  WHERE dept IN('Development','Support') ),
eu_engineers AS ( SELECT * FROM engineers WHERE country IN('NL',...) )
SELECT
...
FROM eu_engineers;

```

## Multiple Uses of a CTE

This can be an 'anti-self join', for example:

```

WITH engineers AS (
  SELECT * FROM employees
  WHERE dept IN('Development','Support') )

SELECT * FROM engineers E1
WHERE NOT EXISTS
  (SELECT 1 FROM engineers E2
   WHERE E2.country=E1.country
   AND E2.name <> E1.name );

```

Or, for year-over-year comparisons, for example:

```

WITH sales_product_year AS (
SELECT product, YEAR(ship_date) AS year,
SUM(price) AS total_amt
FROM item_sales
GROUP BY product, year )

SELECT *
FROM sales_product_year CUR,
sales_product_year PREV,
WHERE CUR.product=PREV.product
AND CUR.year=PREV.year + 1
AND CUR.total_amt > PREV.total_amt

```

Another use is to compare individuals against their group. Below is an example of how this might be executed:

```

WITH sales_product_year AS (
SELECT product,
YEAR(ship_date) AS year,
SUM(price) AS total_amt
FROM item_sales
GROUP BY product, year
)

SELECT *
FROM sales_product_year S1
WHERE
total_amt >
(SELECT 0.1 * SUM(total_amt)
FROM sales_product_year S2
WHERE S2.year = S1.year)

```

## 1.1.4.1.6.3 Recursive Common Table Expressions Overview

MariaDB starting with 10.2.2

Recursive Common Table Expressions have been supported since MariaDB 10.2.2 .

### Contents

1. Syntax example
2. Computation
3. Summary so far
4. CAST to avoid truncating data
5. Examples
  1. Transitive closure - determining bus destinations
  2. Computing paths - determining bus routes
  3. CAST to avoid data truncation

Common Table Expressions (CTEs) are a standard SQL feature, and are essentially temporary named result sets. CTEs first appeared in the SQL standard in 1999, and the first implementations began appearing in 2007.

There are two kinds of CTEs:

- Non-recursive
- Recursive, which this article covers.

SQL is generally poor at recursive structures.



CTEs permit a query to reference itself. A recursive CTE will repeatedly execute subsets of the data until it obtains the complete result set. This makes it particularly useful for handling hierarchical or tree-structured data. `max_recursive_iterations` avoids infinite loops.

## Syntax example

WITH RECURSIVE signifies a recursive CTE. It is given a name, followed by a body (the main query) as follows:

```
with recursive ancestors as (
    select * from folks
    where name = 'Alex' ← Anchor part
    union [all]
    select f.*
    from folks as f, ancestors AS a
    where
        f.id = a.father or f.id = a.mother ← Recursive use of CTE
)
select * from ancestors; ← Recursive part
```

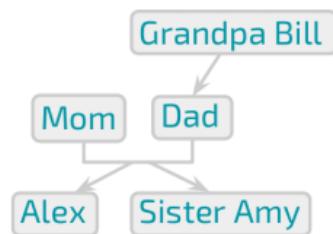
```
WITH ← WITH
with engineers as ( ← CTE name
    select *
    from employees
    where dept='Engineering' ← CTE Body
)
select *
from engineers ← CTE Usage
where ...
```

## Computation

### Recursive CTE computation

Given the following structure:

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL
10	Grandpa Bill	NULL	NULL
98	Sister Amy	20	30



### Computation

First execute the anchor part of the query:

```
with recursive ancestors as (
    select * from folks
    where name = 'Alex'
    union
    select f.*
    from folks as f, ancestors AS a
    where
        f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

### Result table

id	name	father	mother
100	Alex	20	30

Step #1:  
execution of the anchor part

## Computation

## Result table

```
with recursive ancestors as (
  select * from folks
  where name = 'Alex'
  union
  select f.*
  from folks as f, ancestors AS a
  where
    f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

id	name	father	mother
100	Alex	20	30

Next, execute the recursive part of the query:

### Step #2: execution of the recursive part

## Computation

## Result table

```
with recursive ancestors as (
  select * from folks
  where name = 'Alex'
  union
  select f.*
  from folks as f, ancestors AS a
  where
    f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL

## Computation

## Result table

```
with recursive ancestors as (
  select * from folks
  where name = 'Alex'
  union
  select f.*
  from folks as f, ancestors AS a
  where
    f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL
10	Grandpa Bill	NULL	NULL
98	Sister Amy	20	30

## Computation

## Result table

```
with recursive ancestors as (
  select * from folks
  where name = 'Alex'
  union
  select f.*
  from folks as f, ancestors AS a
  where
    f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL
10	Grandpa Bill	NULL	NULL
98	Sister Amy	20	30

## Computation

## Result table

```
with recursive ancestors as (
  select * from folks
  where name = 'Alex'
  union
  select f.*
  from folks as f, ancestors AS a
  where
    f.id = a.father or f.id = a.mother
)
select * from ancestors;
```

id	name	father	mother
100	Alex	20	30
20	Dad	10	NULL
30	Mom	NULL	NULL
10	Grandpa Bill	NULL	NULL
98	Sister Amy	20	30

No results!

## Summary so far

```
with recursive R as (
  select anchor_data
  union [all]
  select recursive_part
  from R, ...
)
select ...
```

1. Compute anchor\_data
2. Compute recursive\_part to get the new data
3. if (new data is non-empty) goto 2;

## CAST to avoid truncating data

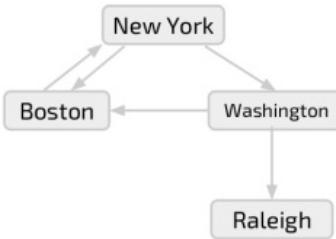
As currently implemented by MariaDB and by the SQL Standard, data may be truncated if not correctly cast. It is necessary to **CAST** the column to the correct width if the CTE's recursive part produces wider values for a column than the CTE's nonrecursive part. Some other DBMS give an error in this situation, and MariaDB's behavior may change in future - see [MDEV-12325](#). See the [examples below](#).

## Examples

Transitive closure - determining bus destinations

Sample data:

bus_routes	
origin	dst
New York	Boston
Boston	New York
New York	Washington
Washington	Boston
Washington	Raleigh



```
CREATE TABLE bus_routes (origin varchar(50), dst varchar(50));
INSERT INTO bus_routes VALUES
  ('New York', 'Boston'),
  ('Boston', 'New York'),
  ('New York', 'Washington'),
  ('Washington', 'Boston'),
  ('Washington', 'Raleigh');
```

Now, we want to return the bus destinations with New York as the origin:

```
WITH RECURSIVE bus_dst as (
  SELECT origin as dst FROM bus_routes WHERE origin='New York'
  UNION
  SELECT bus_routes.dst FROM bus_routes JOIN bus_dst ON bus_dst.dst= bus_routes.origin
)
SELECT * FROM bus_dst;
+-----+
| dst   |
+-----+
| New York |
| Boston   |
| Washington |
| Raleigh  |
+-----+
```

The above example is computed as follows:

First, the anchor data is calculated:

- Starting from New York
- Boston and Washington are added

Next, the recursive part:

- Starting from Boston and then Washington
- Raleigh is added
- UNION excludes nodes that are already present.

## Computing paths - determining bus routes

This time, we are trying to get bus routes such as “New York -> Washington -> Raleigh”.

Using the same sample data as the previous example:

```
WITH RECURSIVE paths (cur_path, cur_dest) AS (
    SELECT origin, origin FROM bus_routes WHERE origin='New York'
    UNION
    SELECT CONCAT(paths.cur_path, ',', bus_routes.dst), bus_routes.dst
    FROM paths
    JOIN bus_routes
    ON paths.cur_dest = bus_routes.origin AND
       NOT FIND_IN_SET(bus_routes.dst, paths.cur_path)
)
SELECT * FROM paths;
```

cur_path	cur_dest
New York	New York
New York,Boston	Boston
New York,Washington	Washington
New York,Washington,Boston	Boston
New York,Washington,Raleigh	Raleigh

## CAST to avoid data truncation

In the following example, data is truncated because the results are not specifically cast to a wide enough type:

```
WITH RECURSIVE tbl AS (
    SELECT NULL AS col
    UNION
    SELECT "THIS NEVER SHOWS UP" AS col FROM tbl
)
SELECT col FROM tbl
```

col
NULL

Explicitly use `CAST` to overcome this:

```
WITH RECURSIVE tbl AS (
    SELECT CAST(NULL AS CHAR(50)) AS col
    UNION SELECT "THIS NEVER SHOWS UP" AS col FROM tbl
)
SELECT * FROM tbl;
```

col
NULL
THIS NEVER SHOWS UP

## 1.1.4.1.7 SELECT WITH ROLLUP

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Syntax

See [SELECT](#) for the full syntax.

### Description

The

WITH ROLLUP

modifier adds extra rows to the resultset that represent super-aggregate summaries. The super-aggregated column is represented by a NULL value. Multiple aggregates over different columns will be added if there are multiple GROUP BY columns.

The LIMIT clause can be used at the same time, and is applied after the

WITH ROLLUP

rows have been added.

WITH ROLLUP

cannot be used with ORDER BY . Some sorting is still possible by using

ASC

or

DESC

clauses with the

GROUP BY

column, although the super-aggregate rows will always be added last.

## Examples

These examples use the following sample table

```
CREATE TABLE booksales (
    country VARCHAR(35), genre ENUM('fiction','non-fiction'), year YEAR, sales INT);

INSERT INTO booksales VALUES
    ('Senegal','fiction',2014,12234), ('Senegal','fiction',2015,15647),
    ('Senegal','non-fiction',2014,64980), ('Senegal','non-fiction',2015,78901),
    ('Paraguay','fiction',2014,87970), ('Paraguay','fiction',2015,76940),
    ('Paraguay','non-fiction',2014,8760), ('Paraguay','non-fiction',2015,9030);
```

The addition of the

WITH ROLLUP

modifier in this example adds an extra row that aggregates both years:

```
SELECT year, SUM(sales) FROM booksales GROUP BY year;
+-----+
| year | SUM(sales) |
+-----+
| 2014 | 173944 |
| 2015 | 180518 |
+-----+
2 rows in set (0.08 sec)

SELECT year, SUM(sales) FROM booksales GROUP BY year WITH ROLLUP;
+-----+
| year | SUM(sales) |
+-----+
| 2014 | 173944 |
| 2015 | 180518 |
| NULL | 354462 |
+-----+
```

In the following example, each time the genre, the year or the country change, another super-aggregate row is added:

```

SELECT country, year, genre, SUM(sales)
  FROM booksales GROUP BY country, year, genre;
+-----+-----+-----+
| country | year | genre      | SUM(sales) |
+-----+-----+-----+
| Paraguay | 2014 | fiction    | 87970 |
| Paraguay | 2014 | non-fiction | 8760 |
| Paraguay | 2015 | fiction    | 76940 |
| Paraguay | 2015 | non-fiction | 9030 |
| Senegal  | 2014 | fiction    | 12234 |
| Senegal  | 2014 | non-fiction | 64980 |
| Senegal  | 2015 | fiction    | 15647 |
| Senegal  | 2015 | non-fiction | 78901 |
+-----+-----+-----+


SELECT country, year, genre, SUM(sales)
  FROM booksales GROUP BY country, year, genre WITH ROLLUP;
+-----+-----+-----+
| country | year | genre      | SUM(sales) |
+-----+-----+-----+
| Paraguay | 2014 | fiction    | 87970 |
| Paraguay | 2014 | non-fiction | 8760 |
| Paraguay | 2014 | NULL       | 96730 |
| Paraguay | 2015 | fiction    | 76940 |
| Paraguay | 2015 | non-fiction | 9030 |
| Paraguay | 2015 | NULL       | 85970 |
| Paraguay | NULL  | NULL       | 182700 |
| Senegal  | 2014 | fiction    | 12234 |
| Senegal  | 2014 | non-fiction | 64980 |
| Senegal  | 2014 | NULL       | 77214 |
| Senegal  | 2015 | fiction    | 15647 |
| Senegal  | 2015 | non-fiction | 78901 |
| Senegal  | 2015 | NULL       | 94548 |
| Senegal  | NULL  | NULL       | 171762 |
| NULL    | NULL  | NULL       | 354462 |
+-----+-----+-----+

```

The LIMIT clause, applied after WITH ROLLUP:

```

SELECT country, year, genre, SUM(sales)
  FROM booksales GROUP BY country, year, genre WITH ROLLUP LIMIT 4;
+-----+-----+-----+
| country | year | genre      | SUM(sales) |
+-----+-----+-----+
| Paraguay | 2014 | fiction    | 87970 |
| Paraguay | 2014 | non-fiction | 8760 |
| Paraguay | 2014 | NULL       | 96730 |
| Paraguay | 2015 | fiction    | 76940 |
+-----+-----+-----+

```

Sorting by year descending:

```

SELECT country, year, genre, SUM(sales)
  FROM booksales GROUP BY country, year DESC, genre WITH ROLLUP;
+-----+-----+-----+
| country | year | genre      | SUM(sales) |
+-----+-----+-----+
| Paraguay | 2015 | fiction    | 76940 |
| Paraguay | 2015 | non-fiction | 9030 |
| Paraguay | 2015 | NULL       | 85970 |
| Paraguay | 2014 | fiction    | 87970 |
| Paraguay | 2014 | non-fiction | 8760 |
| Paraguay | 2014 | NULL       | 96730 |
| Paraguay | NULL  | NULL       | 182700 |
| Senegal  | 2015 | fiction    | 15647 |
| Senegal  | 2015 | non-fiction | 78901 |
| Senegal  | 2015 | NULL       | 94548 |
| Senegal  | 2014 | fiction    | 12234 |
| Senegal  | 2014 | non-fiction | 64980 |
| Senegal  | 2014 | NULL       | 77214 |
| Senegal  | NULL  | NULL       | 171762 |
| NULL    | NULL  | NULL       | 354462 |
+-----+-----+-----+

```

## See Also

- [SELECT](#)
- [Joins and Subqueries](#)
- [LIMIT](#)
- [ORDER BY](#)
- [GROUP BY](#)
- [Common Table Expressions](#)
- [SELECT INTO OUTFILE](#)
- [SELECT INTO DUMPFILE](#)
- [FOR UPDATE](#)
- [LOCK IN SHARE MODE](#)
- [Optimizer Hints](#)

## 1.1.4.1.8 SELECT INTO OUTFILE

### Syntax

```
SELECT ... INTO OUTFILE 'file_name'  
    [CHARACTER SET charset_name]  
    [export_options]  
  
export_options:  
    [{FIELDS | COLUMNS}  
        [TERMINATED BY 'string']  
        [[OPTIONALLY] ENCLOSED BY 'char']  
        [ESCAPED BY 'char']  
    ]  
    [LINES  
        [STARTING BY 'string']  
        [TERMINATED BY 'string']  
    ]
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [Character-sets](#)
3. [Example](#)
4. [See Also](#)

### Description

```
SELECT INTO OUTFILE
```

writes the resulting rows to a file, and allows the use of column and row terminators to specify a particular output format. The default is to terminate fields with tabs (`\t`) and lines with newlines (`\n`).

The file must not exist. It cannot be overwritten. A user needs the `FILE` privilege to run this statement. Also, MariaDB needs permission to write files in the specified location. If the `secure_file_priv` system variable is set to a non-empty directory name, the file can only be written to that directory.

The

```
LOAD DATA INFILE
```

statement complements

```
SELECT INTO OUTFILE
```

### Character-sets

The

```
CHARACTER SET
```

clause specifies the `character set` in which the results are to be written. Without the clause, no conversion takes place (the binary character set). In this case, if there are multiple character sets, the output will contain these too, and may not easily be able to be reloaded.

In cases where you have two servers using different character-sets, using

```
SELECT INTO OUTFILE
to transfer data from one to the other can have unexpected results. To ensure that MariaDB correctly interprets the escape sequences, use the
CHARACTER SET
clause on both the
SELECT INTO OUTFILE
statement and the subsequent
```

```
LOAD DATA INFILE
```

statement.

## Example

The following example produces a file in the CSV format:

```
SELECT customer_id, firstname, surname INTO OUTFILE '/exportdata/customers.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ""
LINES TERMINATED BY '\n'
FROM customers;
```

## See Also

- [SELECT](#)
- [LOAD\\_DATA\(\)](#) function
- [LOAD DATA INFILE](#)
- [SELECT INTO Variable](#)
- [SELECT INTO DUMPFILE](#)

### 1.1.4.1.9 SELECT INTO DUMPFILE

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [See Also](#)

## Syntax

```
SELECT ... INTO DUMPFILE 'file_path'
```

## Description

```
SELECT ... INTO DUMPFILE
```

is a [SELECT](#) clause which writes the resultset into a single unformatted row, without any separators, in a file. The results will not be returned to the client.

*file\_path* can be an absolute path, or a relative path starting from the data directory. It can only be specified as a [string literal](#), not as a variable. However, the statement can be dynamically composed and executed as a prepared statement to work around this limitation.

This statement is binary-safe and so is particularly useful for writing [BLOB](#) values to file. It can be used, for example, to copy an image or an audio document from the database to a file. [SELECT ... INTO FILE](#) can be used to save a text file.

The file must not exist. It cannot be overwritten. A user needs the [FILE](#) privilege to run this statement. Also, MariaDB needs permission to write files in the specified location. If the [secure\\_file\\_priv](#) system variable is set to a non-empty directory name, the file can only be written to that directory.

Since [MariaDB 5.1](#), the [character\\_set\\_filesystem](#) system variable has controlled interpretation of file names that are given as literal strings.

## Example

```
SELECT _utf8'Hello world!' INTO DUMPFILE '/tmp/world';

SELECT LOAD_FILE('/tmp/world') AS world;
+-----+
| world      |
+-----+
| Hello world! |
+-----+
```

## See Also

- [SELECT](#)
- [LOAD\\_FILE\(\)](#)
- [SELECT INTO Variable](#)
- [SELECT INTO OUTFILE](#)

## 1.1.4.1.10 FOR UPDATE

InnoDB supports row-level locking. Selected rows can be locked using [LOCK IN SHARE MODE](#) or [FOR UPDATE](#). In both cases, a lock is acquired on the rows read by the query, and it will be released when the current transaction is committed.

The

FOR UPDATE  
clause of [SELECT](#) applies only when [autocommit](#) is set to 0 or the

SELECT  
is enclosed in a transaction. A lock is acquired on the rows, and other transactions are prevented from writing the rows, acquire locks, and from reading them (unless their isolation level is

[READ UNCOMMITTED](#)

).

If

autocommit  
is set to 1, the [LOCK IN SHARE MODE](#) and  
FOR UPDATE  
clauses have no effect.

If the isolation level is set to [SERIALIZABLE](#), all plain

SELECT  
statements are converted to  
SELECT ... LOCK IN SHARE MODE

## Example

```
SELECT * FROM trans WHERE period=2001 FOR UPDATE;
```

## See Also

- [SELECT](#)
- [LOCK IN SHARE MODE](#)
- [InnoDB Lock Modes](#)

## 1.1.4.1.11 LOCK IN SHARE MODE

InnoDB supports row-level locking. Selected rows can be locked using

[LOCK IN SHARE MODE](#)  
or [FOR UPDATE](#). In both cases, a lock is acquired on the rows read by the query, and it will be released when the current transaction is committed.

When

[LOCK IN SHARE MODE](#)  
is specified in a [SELECT](#) statement, MariaDB will wait until all transactions that have modified the rows are committed. Then, a write lock is acquired. All transactions can read the rows, but if they want to modify them, they have to wait until your transaction is committed.

If

autocommit  
is set to 1, the [LOCK IN SHARE MODE](#) and [FOR UPDATE](#) clauses have no effect.

## See Also

- [SELECT](#)
- [FOR UPDATE](#)
- [InnoDB Lock Modes](#)

## 1.1.4.1.12 Optimizer Hints

# Optimizer hints

There are some options available in `SELECT` to affect the execution plan. These are known as optimizer hints.

## HIGH\_PRIORITY

```
HIGH_PRIORITY
gives the statement a higher priority. If the table is locked, high priority
SELECT
s will be executed as soon as the lock is released, even if other statements are queued.
HIGH_PRIORITY
applies only if the storage engine only supports table-level locking (
MyISAM
,
MEMORY
,
MERGE
). See HIGH\_PRIORITY and LOW\_PRIORITY clauses for details.
```

## SQL\_CACHE / SQL\_NO\_CACHE

If the `query_cache_type` system variable is set to 2 or

```
DEMAND
, and the current statement is cacheable,
SQL_CACHE
causes the query to be cached and
SQL_NO_CACHE
causes the query not to be cached. For
UNION
S,
SQL_CACHE
or
SQL_NO_CACHE
should be specified for the first query. See also The Query Cache for more detail and a list of the types of statements that aren't cacheable.
```

## SQL\_BUFFER\_RESULT

```
SQL_BUFFER_RESULT
forces the optimizer to use a temporary table to process the result. This is useful to free locks as soon as possible.
```

## SQL\_SMALL\_RESULT / SQL\_BIG\_RESULT

```
SQL_SMALL_RESULT
and
SQL_BIG_RESULT
tell the optimizer whether the result is very big or not. Usually,
GROUP BY
and
DISTINCT
operations are performed using a temporary table. Only if the result is very big, using a temporary table is not convenient. The optimizer
automatically knows if the result is too big, but you can force the optimizer to use a temporary table with
SQL_SMALL_RESULT
, or avoid the temporary table using
SQL_BIG_RESULT
```

## STRAIGHT\_JOIN

```
STRAIGHT_JOIN
applies to the JOIN queries, and tells the optimizer that the tables must be read in the order they appear in the
SELECT
. For
const
and
system
```

table this option is sometimes ignored.

## SQL\_CALC\_FOUND\_ROWS

SQL\_CALC\_FOUND\_ROWS  
is only applied when using the  
LIMIT  
clause. If this option is used, MariaDB will count how many rows would match the query, without the  
LIMIT  
clause. That number can be retrieved in the next query, using [FOUND\\_ROWS\(\)](#).

## USE/FORCE/IGNORE INDEX

USE INDEX  
,  
FORCE INDEX  
and  
IGNORE INDEX  
constrain the query planning to a specific index.

For further information about some of these options, see [How to force query plans](#).

## 1.1.4.1.13 PROCEDURE

The

PROCEDURE  
clause of [SELECT](#) passes the whole result set to a Procedure which will process it. These Procedures are not [Stored Procedures](#), and can only be written in the C language, so it is necessary to recompile the server.

Currently, the only available procedure is [ANALYSE](#), which examines the resultset and suggests the optimal datatypes for each column. It is defined in the

sql/sql\_analyse.cc  
file, and can be used as an example to create more Procedures.

This clause cannot be used in a [view](#)'s definition.

## See Also

- [SELECT](#)
- [Stored Procedures](#)

## 1.1.4.1.14 HANDLER

### 1.1.4.1.14.1 HANDLER Commands

#### Syntax

```
HANDLER tbl_name OPEN [ [AS] alias]
HANDLER tbl_name READ index_name { = | >= | <= | < } (value1,value2,...)
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ index_name { FIRST | NEXT | PREV | LAST }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ { FIRST | NEXT }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name CLOSE
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Key Lookup](#)
4. [Key Scans](#)
5. [Table Scans](#)
6. [Limitations](#)
  1. [Finding 'Old Rows'](#)
  2. [Invisible Columns](#)
  3. [System-Versioned Tables](#)
  4. [Other Limitations](#)
7. [Error Codes](#)
8. [See Also](#)

## Description

The

HANDLER

statement provides direct access to table storage engine interfaces for key lookups and key or table scans. It is available for at least [Aria](#), [Memory](#), [MyISAM](#) and [InnoDB](#) tables (and should work with most 'normal' storage engines, but not with system tables, [MERGE](#) or [views](#)).

HANDLER ... OPEN

opens a table, allowing it to be accessible to subsequent

HANDLER ... READ

statements. The table can either be opened using an alias (which must then be used by

HANDLER ... READ

, or a table name.

The table object is only closed when

HANDLER ... CLOSE

is called by the session, and is not shared by other sessions.

[Prepared statements](#) work with

HANDLER READ

, which gives a much higher performance (50% speedup) as there is no parsing and all data is transformed in binary (without conversions to text, as with the normal protocol).

The HANDLER command does not work with [partitioned tables](#).

## Key Lookup

A key lookup is started with:

```
HANDLER tbl_name READ index_name { = | >= | <= | < } (value,value) [LIMIT...]
```

The values stands for the value of each of the key columns. For most key types (except for HASH keys in MEMORY storage engine) you can use a prefix subset of it's columns.

If you are using LIMIT, then in case of >= or > then there is an implicit NEXT implied, while if you are using <= or < then there is an implicit PREV implied.

After the initial read, you can use

```
HANDLER tbl_name READ index_name NEXT [ LIMIT ... ]  
or  
HANDLER tbl_name READ index_name PREV [ LIMIT ... ]
```

to scan the rows in key order.

Note that the row order is not defined for keys with duplicated values and will vary from engine to engine.

## Key Scans

You can scan a table in key order by doing:

```
HANDLER tbl_name READ index_name FIRST [ LIMIT ... ]  
HANDLER tbl_name READ index_name NEXT [ LIMIT ... ]
```

or, if the handler supports backwards key scans (most do):

```
HANDLER tbl_name READ index_name LAST [ LIMIT ... ]
HANDLER tbl_name READ index_name PREV [ LIMIT ... ]
```

## Table Scans

You can scan a table in row order by doing:

```
HANDLER tbl_name READ FIRST [ LIMIT ... ]
HANDLER tbl_name READ NEXT [ LIMIT ... ]
```

## Limitations

As this is a direct interface to the storage engine, some limitations may apply for what you can do and what happens if the table changes. Here follows some of the common limitations:

### Finding 'Old Rows'

HANDLER READ is not transaction safe, consistent or atomic. It's ok for the storage engine to return rows that existed when you started the scan but that were later deleted. This can happen as the storage engine may cache rows as part of the scan from a previous read.

You may also find rows committed since the scan originally started.

### Invisible Columns

```
HANDLER ... READ
also reads the data of invisible-columns .
```

## System-Versioned Tables

```
HANDLER ... READ
reads everything from system-versioned tables , and so includes
row_start
and
row_end
fields, as well as all rows that have since been deleted or changed, including when history partitions are used.
```

## Other Limitations

- If you do an [ALTER TABLE](#) , all your HANDLER's for that table are automatically closed.
- If you do an ALTER TABLE for a table that is used by some other connection with HANDLER, the ALTER TABLE will wait for the HANDLER to be closed.
- For HASH keys, you must use all key parts when searching for a row.
- For HASH keys, you can't do a key scan of all values. You can only find all rows with the same key value.
- While each HANDLER READ command is atomic, if you do a scan in many steps, then some engines may give you error 1020 if the table changed between the commands. Please refer to the [specific engine handler page](#) if this happens.

## Error Codes

- Error 1031 (ER\_ILLEGAL\_HA) Table storage engine for 't1' doesn't have this option
  - If you get this for HANDLER OPEN it means the storage engine doesn't support HANDLER calls.
  - If you get this for HANDLER READ it means you are trying to use an incomplete HASH key.
- Error 1020 (ER\_CHECKREAD) Record has changed since last read in table '...'
  - This means that the table changed between two reads and the handler can't handle this case for the given scan.

## See Also

- [What is MariaDB 5.3](#)

## 1.1.4.1.14.2 HANDLER for MEMORY Tables

This article explains how to use [HANDLER commands](#) efficiently with [MEMORY/HEAP](#) tables.

If you want to scan a table for over different key values, not just search for exact key values, you should create your keys with 'USING BTREE':

```
CREATE TABLE t1 (a INT, b INT, KEY(a), KEY b USING BTREE (b)) engine=memory;
```

In the above table,

- a is a **HASH** key that only supports exact matches (=) while
- b is a **BTREE** key that you can use to scan the table in key order, starting from start or from a given key value.

The limitations for HANDLER READ with Memory|HEAP tables are:

## Limitations for HASH keys

- You must use all key parts when searching for a row.
- You can't do a key scan of all values. You can only find all rows with the same key value.
- READ NEXT gives error 1031 if the tables changed since last read.

## Limitations for BTREE keys

- READ NEXT gives error 1031 if the tables changed since last read. This limitation can be lifted in the future.

## Limitations for table scans

- READ NEXT gives error 1031 if the table was truncated since last READ call.

## See also

See also the the limitations listed in [HANDLER commands](#).

## 1.1.4.1.15 DUAL

### Description

You are allowed to specify

DUAL

as a dummy table name in situations where no tables are referenced, such as the following **SELECT** statement:

```
SELECT 1 + 1 FROM DUAL;
+-----+
| 1 + 1 |
+-----+
|      2 |
+-----+
```

DUAL

is purely for the convenience of people who require that all

SELECT

statements should have

FROM

and possibly other clauses. MariaDB ignores the clauses. MariaDB does not require

FROM DUAL

if no tables are referenced.

FROM DUAL could be used when you only SELECT computed values, but require a WHERE clause, perhaps to test that a script correctly handles empty resultsets:

```
SELECT 1 FROM DUAL WHERE FALSE;
Empty set (0.00 sec)
```

## See Also

- [SELECT](#)

## 1.1.4.1.16 SELECT ... OFFSET ... FETCH

MariaDB starting with [10.6.0](#)

```
SELECT ... OFFSET ... FETCH  
was introduced in MariaDB 10.6.
```

## Syntax

```
OFFSET start { ROW | ROWS }  
FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } { ONLY | WITH TIES }
```

## Description

The

OFFSET clause allows one to return only those elements of a resultset that come after a specified offset. The FETCH clause specifies the number of rows to return, while ONLY or WITH TIES specifies whether or not to also return any further results that tie for last place according to the ordered resultset.

Either the singular

ROW or the plural  
ROWS can be used after the  
OFFSET and  
FETCH clauses; the choice has no impact on the results.

In the case of

WITH TIES, an ORDER BY clause is required, otherwise an ERROR will be returned.

```
SELECT i FROM t1 FETCH FIRST 2 ROWS WITH TIES;  
ERROR 4180 (HY000): FETCH ... WITH TIES requires ORDER BY clause to be present
```

## Examples

Given a table with 6 rows:

```
CREATE OR REPLACE TABLE t1 (i INT);  
INSERT INTO t1 VALUES (1),(2),(3),(4), (4), (5);  
SELECT i FROM t1 ORDER BY i ASC;  
+---+  
| i |  
+---+  
| 1 |  
| 2 |  
| 3 |  
| 4 |  
| 4 |  
| 5 |  
+---+
```

OFFSET 2 allows one to skip the first two results.

```
SELECT i FROM t1 ORDER BY i ASC OFFSET 2 ROWS;
+---+
| i |
+---+
| 3 |
| 4 |
| 4 |
| 5 |
+---+
```

FETCH FIRST 3 ROWS ONLY  
limits the results to three rows only

```
SELECT i FROM t1 ORDER BY i ASC OFFSET 1 ROWS FETCH FIRST 3 ROWS ONLY;
+---+
| i |
+---+
| 2 |
| 3 |
| 4 |
+---+
```

The same outcome can also be achieved with the [LIMIT](#) clause:

```
SELECT i FROM t1 ORDER BY i ASC LIMIT 3 OFFSET 1;
+---+
| i |
+---+
| 2 |
| 3 |
| 4 |
+---+
```

WITH TIES  
ensures the tied result  
4  
is also returned.

```
SELECT i FROM t1 ORDER BY i ASC OFFSET 1 ROWS FETCH FIRST 3 ROWS WITH TIES;
+---+
| i |
+---+
| 2 |
| 3 |
| 4 |
| 4 |
+---+
```

## See Also

- [ORDER BY](#)
- [SELECT](#)

### 1.1.4.2 Inserting and Loading Data

#### 1.1.4.2.1 INSERT

## Syntax

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ] [RETURNING select_expr
  [, select_expr ...]]
```

Or:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)]
SET col={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ] [RETURNING select_expr
  [, select_expr ...]]
```

Or:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
SELECT ...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ] [RETURNING select_expr
  [, select_expr ...]]
```

## Contents

1. [Syntax](#)
2. [INSERT DELAYED](#)
3. [HIGH PRIORITY and LOW PRIORITY](#)
4. [Defaults and Duplicate Values](#)
5. [INSERT IGNORE](#)
6. [INSERT ON DUPLICATE KEY UPDATE](#)
7. [Examples](#)
8. [INSERT ... RETURNING](#)
  1. [Examples](#)
9. [See Also](#)

The

`INSERT`  
statement is used to insert new rows into an existing table. The  
`INSERT ... VALUES`  
and  
`INSERT ... SET`  
forms of the statement insert rows based on explicitly specified values. The  
`INSERT ... SELECT`  
form inserts rows selected from another table or tables.  
`INSERT ... SELECT`  
is discussed further in the

[INSERT ... SELECT](#)

article.

The table name can be specified in the form

`db_name`  
. . .  
`tbl_name`  
or, if a default database is selected, in the form  
`tbl_name`  
(see [Identifier Qualifiers](#)). This allows to use `INSERT ... SELECT` to copy rows between different databases.

The PARTITION clause can be used in both the INSERT and the SELECT part. See [Partition Pruning and Selection](#) for details.

MariaDB starting with [10.5](#)

The RETURNING clause was introduced in [MariaDB 10.5](#).

The columns list is optional. It specifies which values are explicitly inserted, and in which order. If this clause is not specified, all values must be explicitly specified, in the same order they are listed in the table definition.

The list of value follow the

`VALUES`  
or  
`VALUE`

keyword (which are interchangeable, regardless how much values you want to insert), and is wrapped by parenthesis. The values must be listed in the same order as the columns list. It is possible to specify more than one list to insert more than one rows with a single statement. If many rows are inserted, this is a speed optimization.

For one-row statements, the

```
SET  
clause may be more simple, because you don't need to remember the columns order. All values are specified in the form  
col  
=  
expr
```

Values can also be specified in the form of a SQL expression or subquery. However, the subquery cannot access the same table that is named in the  
INTO  
clause.

If you use the

```
LOW_PRIORITY  
keyword, execution of the  
INSERT  
is delayed until no other clients are reading from the table. If you use the  
HIGH_PRIORITY  
keyword, the statement has the same priority as  
SELECT  
s. This affects only storage engines that use only table-level locking (MyISAM, MEMORY, MERGE). However, if one of these keywords is  
specified, concurrent inserts cannot be used. See HIGH\_PRIORITY and LOW\_PRIORITY clauses for details.
```

## INSERT DELAYED

For more details on the

```
DELAYED  
option, see INSERT DELAYED.
```

## HIGH PRIORITY and LOW PRIORITY

See [HIGH\\_PRIORITY and LOW\\_PRIORITY](#).

## Defaults and Duplicate Values

See [INSERT - Default & Duplicate Values](#) for details..

## INSERT IGNORE

See [INSERT IGNORE](#).

## INSERT ON DUPLICATE KEY UPDATE

See [INSERT ON DUPLICATE KEY UPDATE](#).

## Examples

Specifying the column names:

```
INSERT INTO person (first_name, last_name) VALUES ('John', 'Doe');
```

Inserting more than 1 row at a time:

```
INSERT INTO tbl_name VALUES (1, "row 1"), (2, "row 2");
```

Using the

```
SET  
clause:
```

```
INSERT INTO person SET first_name = 'John', last_name = 'Doe';
```

SELECTing from another table:

```
INSERT INTO contractor SELECT * FROM person WHERE status = 'c';
```

See [INSERT ON DUPLICATE KEY UPDATE](#) and [INSERT IGNORE](#) for further examples.

# INSERT ... RETURNING

INSERT ... RETURNING

returns a resultset of the inserted rows.

This returns the listed columns for all the rows that are inserted, or alternatively, the specified SELECT expression. Any SQL expressions which can be calculated can be used in the select expression for the RETURNING clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

## Examples

Simple INSERT statement

```
INSERT INTO t2 VALUES (1,'Dog'),(2,'Lion'),(3,'Tiger'),(4,'Leopard')
RETURNING id2,id2+id2,id2&id2,id2||id2;
+-----+-----+-----+-----+
| id2 | id2+id2 | id2&id2 | id2||id2 |
+-----+-----+-----+
|  1  |      2  |      1  |      1  |
|  2  |      4  |      2  |      1  |
|  3  |      6  |      3  |      1  |
|  4  |      8  |      4  |      1  |
+-----+-----+-----+
```

Using stored functions in RETURNING

```
DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
BEGIN
    RETURN (SELECT arg+arg);
END|
DELIMITER ;

PREPARE stmt FROM "INSERT INTO t1 SET id1=1, animal1='Bear' RETURNING f(id1), UPPER(animal1)";

EXECUTE stmt;
+-----+-----+
| f(id1) | UPPER(animal1) |
+-----+-----+
|     2  | BEAR           |
+-----+-----+
```

Subqueries in the RETURNING clause that return more than one row or column cannot be used.

Aggregate functions cannot be used in the RETURNING clause. Since aggregate functions work on a set of values, and if the purpose is to get the row count, ROW\_COUNT() with SELECT can be used or it can be used in INSERT...SELECT...RETURNING if the table in the RETURNING clause is not the same as the INSERT table.

## See Also

- [INSERT DELAYED](#)
- [INSERT SELECT](#)
- [REPLACE](#) Equivalent to DELETE + INSERT of conflicting row.
- [HIGH\\_PRIORITY and LOW\\_PRIORITY](#)
- [Concurrent Inserts](#)
- [INSERT - Default & Duplicate Values](#)
- [INSERT IGNORE](#)
- [INSERT ON DUPLICATE KEY UPDATE](#)
- [How to quickly insert data into MariaDB](#)

## 1.1.4.2.2 INSERT DELAYED

### Syntax

```
INSERT DELAYED ...
```



- INSERT DELAYED  
should be used only for  
INSERT  
statements that specify value lists. The server ignores  
DELAYED  
for  
INSERT ... SELECT  
or  
INSERT ... ON DUPLICATE KEY UPDATE  
statements.
- Because the  
`INSERT DELAYED`  
statement returns immediately, before the rows are inserted, you cannot use  
`LAST_INSERT_ID()`  
to get the  
`AUTO_INCREMENT`  
value that the statement might generate.
  - `DELAYED`  
rows are not visible to  
`SELECT`  
statements until they actually have been inserted.
  - After  
`INSERT DELAYED`  
`, ROW_COUNT()` returns the number of the rows you tried to insert, not the number of the successful writes.
  - `DELAYED`  
is ignored on slave replication servers, so that  
`INSERT DELAYED`  
is treated as a normal  
`INSERT`  
on slaves. This is because  
`DELAYED`  
could cause the slave to have different data than the master.  
`INSERT DELAYED`  
statements are not [safe for replication](#).
  - Pending  
`INSERT DELAYED`  
statements are lost if a table is write locked and `ALTER TABLE` is used to modify the table structure.
  - `INSERT DELAYED`  
is not supported for views. If you try, you will get an error like this:  
`ERROR 1347 (HY000): 'view_name' is not BASE TABLE`
  - `INSERT DELAYED`  
is not supported for [partitioned tables](#).
  - `INSERT DELAYED`  
is not supported within [stored programs](#).
  - `INSERT DELAYED`  
does not work with [triggers](#).
  - `INSERT DELAYED`  
does not work if there is a check constraint in place.
  - `INSERT DELAYED`  
does not work if [skip-new](#) mode is active.

## See Also

- [INSERT](#)
- [INSERT SELECT](#)
- [HIGH\\_PRIORITY and LOW\\_PRIORITY](#)
- [Concurrent Inserts](#)
- [INSERT - Default & Duplicate Values](#)
- [INSERT IGNORE](#)
- [INSERT ON DUPLICATE KEY UPDATE](#)

## 1.1.4.2.3 INSERT SELECT

# Syntax

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
      [INTO] tbl_name [(col_name,...)]
      SELECT ...
      [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

## Description

With

```
INSERT ... SELECT
, you can quickly insert many rows into a table from one or more other tables. For example:
```

```
INSERT INTO tbl_temp2 (fld_id)
SELECT tbl_temp1.fld_order_id
FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;
```

tbl\_name  
can also be specified in the form  
db\_name  
.  
tbl\_name  
(see [Identifier Qualifiers](#)). This allows to copy rows between different databases.

If the new table has a primary key or UNIQUE indexes, you can use [IGNORE](#) to handle duplicate key errors during the query. The newer values will not be inserted if an identical value already exists.

### REPLACE

can be used instead of  
INSERT  
to prevent duplicates on  
UNIQUE  
indexes by deleting old values. In that case,  
ON DUPLICATE KEY UPDATE  
cannot be used.

```
INSERT ... SELECT
works for tables which already exist. To create a table for a given resultset, you can use CREATE TABLE ... SELECT.
```

## See Also

- [INSERT](#)
- [INSERT DELAYED](#)
- [HIGH\\_PRIORITY and LOW\\_PRIORITY](#)
- [Concurrent Inserts](#)
- [INSERT - Default & Duplicate Values](#)
- [INSERT IGNORE](#)
- [INSERT ON DUPLICATE KEY UPDATE](#)

## 1.1.4.2.4 LOAD Data into Tables or Index

### 1.1.4.2.4.1 LOAD DATA INFILE

## Syntax

```

LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
[REPLACE | IGNORE]
INTO TABLE tbl_name
[CHARACTER SET charset_name]
[{{FIELDS | COLUMNS}
  [TERMINATED BY 'string']
  [[OPTIONALLY] ENCLOSED BY 'char']
  [ESCAPED BY 'char']]
]
[LINES
  [STARTING BY 'string']
  [TERMINATED BY 'string']]
]
[IGNORE number LINES]
[(col_name_or_user_var,...)]
[SET col_name = expr,...]

```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [LOAD DATA LOCAL INFILE](#)
  2. [REPLACE and IGNORE](#)
  3. [Character-sets](#)
  4. [Preprocessing Inputs](#)
  5. [Priority and Concurrency](#)
  6. [Progress Reporting](#)
  7. [Using mariadb-import/mysqlimport](#)
  8. [Indexing](#)
3. [Examples](#)
4. [See Also](#)

## Description

`LOAD DATA INFILE`  
is unsafe for statement-based replication.

Reads rows from a text file into the designated table on the database at a very high speed. The file name must be given as a literal string.

Files are written to disk using the `SELECT INTO OUTFILE` statement. You can then read the files back into a table using the

`LOAD DATA INFILE`  
statement. The  
`FIELDS`  
and  
`LINES`  
clauses are the same in both statements. These clauses are optional, but if both are specified then the  
`FIELDS`  
clause must precede  
`LINES`

Executing this statement activates

`INSERT`  
triggers.

One must have the `FILE` privilege to be able to execute `LOAD DATA`. This is to ensure normal users will not attempt to read system files.

Note that MariaDB's `systemd` unit file restricts access to

`/home`  
,

`/root`  
, and  
`/run/user`  
by default. See [Configuring access to home directories](#).

## LOAD DATA LOCAL INFILE

When you execute the

```
LOAD DATA INFILE  
statement, MariaDB Server attempts to read the input file from its own file system. In contrast, when you execute the  
LOAD DATA LOCAL INFILE  
statement, the client attempts to read the input file from its file system, and it sends the contents of the input file to the MariaDB Server. This  
allows you to load files from the client's local file system into the database.
```

In the event that you don't want to permit this operation (such as for security reasons), you can disable the

```
LOAD DATA LOCAL INFILE  
statement on either the server or the client.
```

- The

```
LOAD DATA LOCAL INFILE  
statement can be disabled on the server by setting the local\_infile system variable to  
0
```

- The

```
LOAD DATA LOCAL INFILE  
statement can be disabled on the client. If you are using MariaDB Connector/C, this can be done by unsetting the  
CLIENT_LOCAL_FILES  
capability flag with the mysql\_real\_connect function or by unsetting the  
MYSQL_OPT_LOCAL_INFILE  
option with mysql\_optionsv function. If you are using a different client or client library, then see the documentation for your specific client  
or client library to determine how it handles the
```

```
LOAD DATA LOCAL INFILE  
statement.
```

If the

```
LOAD DATA LOCAL INFILE  
statement is disabled by either the server or the client and if the user attempts to execute it, then the server will cause the statement to fail with  
the following error message:
```

```
The used command is not allowed with this MariaDB version
```

Note that it is not entirely accurate to say that the MariaDB version does not support the command. It would be more accurate to say that the MariaDB configuration does not support the command. See [MDEV-20500](#) for more information.

From [MariaDB 10.5.2](#), the error message is more accurate:

```
The used command is not allowed because the MariaDB server or client  
has disabled the local infile capability
```

## REPLACE and IGNORE

In cases where you load data from a file into a table that already contains data and has a [primary key](#), you may encounter issues where the statement attempts to insert a row with a primary key that already exists. When this happens, the statement fails with Error 1064, protecting the data already on the table. In cases where you want MariaDB to overwrite duplicates, use the

```
REPLACE  
keyword.
```

The

```
REPLACE  
keyword works like the REPLACE statement. Here, the statement attempts to load the data from the file. If the row does not exist, it adds it to the table. If the row contains an existing Primary Key, it replaces the table data. That is, in the event of a conflict, it assumes the file contains the desired row.
```

This operation can cause a degradation in load speed by a factor of 20 or more if the part that has already been loaded is larger than the capacity of the [InnoDB Buffer Pool](#). This happens because it causes a lot of turnaround in the buffer pool.

Use the

```
IGNORE  
keyword when you want to skip any rows that contain a conflicting primary key. Here, the statement attempts to load the data from the file. If the row does not exist, it adds it to the table. If the row contains an existing primary key, it ignores the addition request and moves on to the next. That is, in the event of a conflict, it assumes the table contains the desired row.
```

## Character-sets

When the statement opens the file, it attempts to read the contents using the default character-set, as defined by the [character\\_set\\_database](#) system variable.

In the cases where the file was written using a character-set other than the default, you can specify the character-set to use with the

CHARACTER SET clause in the statement. It ignores character-sets specified by the `SET NAMES` statement and by the `character_set_client` system variable.

Setting the

CHARACTER SET clause to a value of binary indicates "no conversion."

The statement interprets all fields in the file as having the same character-set, regardless of the column data type. To properly interpret file contents, you must ensure that it was written with the correct character-set. If you write a data file with `mysqldump -T` or with the `SELECT INTO OUTFILE` statement with the `mysql` client, be sure to use the

`--default-character-set` option, so that the output is written with the desired character-set.

When using mixed character sets, use the

CHARACTER SET clause in both `SELECT INTO OUTFILE` and `LOAD DATA INFILE` to ensure that MariaDB correctly interprets the escape sequences.

The `character_set_filesystem` system variable controls the interpretation of the filename.

It is currently not possible to load data files that use the

`ucs2` character set.

## Preprocessing Inputs

`col_name_or_user_var` can be a column name, or a user variable. In the case of a variable, the `SET` statement can be used to preprocess the value before loading into the table.

## Priority and Concurrency

In storage engines that perform table-level locking (`MyISAM`, `MEMORY` and `MERGE`), using the `LOW_PRIORITY` keyword, MariaDB delays insertions until no other clients are reading from the table. Alternatively, when using the `MyISAM` storage engine, you can use the `CONCURRENT` keyword to perform concurrent insertion.

The `LOW_PRIORITY` and `CONCURRENT` keywords are mutually exclusive. They cannot be used in the same statement.

## Progress Reporting

The

`LOAD DATA INFILE` statement supports `progress reporting`. You may find this useful when dealing with long-running operations. Using another client you can issue a `SHOW PROCESSLIST` query to check the progress of the data load.

## Using mariadb-import/mysqlimport

MariaDB ships with a separate utility for loading data from files: `mariadb-import` (or

`mysqlimport` before `MariaDB 10.5`). It operates by sending `LOAD DATA INFILE` statements to the server.

Using `mariadb-import/mysqlimport` you can compress the file using the

`--compress` option, to get better performance over slow networks, providing both the client and server support the compressed protocol. Use the `--local` option to load from the local file system.

## Indexing

In cases where the storage engine supports `ALTER TABLE... DISABLE KEYS` statements (`MyISAM` and `Aria`), the

`LOAD DATA INFILE` statement automatically disables indexes during the execution.

## Examples

You have a file with this content (note the the separator is ',', not tab, which is the default):

```
2,2  
3,3  
4,4  
5,5  
6,8
```

```
CREATE TABLE t1 (a int, b int, c int, d int);  
LOAD DATA LOCAL INFILE  
  '/tmp/loaddata7.dat' INTO TABLE t1 fields terminated by ',' (a,b) SET c=a+b;  
SELECT * FROM t1;  
+-----+-----+-----+  
| a    | b    | c    |  
+-----+-----+-----+  
| 2    | 2    | 4    |  
| 3    | 3    | 6    |  
| 4    | 4    | 8    |  
| 5    | 5    | 10   |  
| 6    | 8    | 14   |  
+-----+-----+-----+
```

Another example, given the following data (the separator is a tab):

```
1      a  
2      b
```

The value of the first column is doubled before loading:

```
LOAD DATA INFILE 'ld.txt' INTO TABLE ld (@i,v) SET i=@i*2;  
  
SELECT * FROM ld;  
+-----+-----+  
| i    | v    |  
+-----+-----+  
| 2    | a    |  
| 4    | b    |  
+-----+-----+
```

## See Also

- [How to quickly insert data into MariaDB](#)
- [Character Sets and Collations](#)
- [SELECT ... INTO OUTFILE](#)
- [mariadb-import/mysqlimport](#)

## 1.1.4.2.4.2 LOAD INDEX

## 1.1.4.2.4.3 LOAD XML

## Syntax

```
LOAD XML [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'  
  [REPLACE | IGNORE]  
  INTO TABLE [db_name.]tbl_name  
  [CHARACTER SET charset_name]  
  [ROWS IDENTIFIED BY '<tagname>']  
  [IGNORE number {LINES | ROWS}]  
  [(column_or_user_var,...)]  
  [SET col_name = expr,...]
```

## Description

The LOAD XML statement reads data from an XML file into a table. The

```
file_name  
must be given as a literal string. The  
tagname  
in the optional ROWS IDENTIFIED BY clause must also be given as a literal string, and must be surrounded by angle brackets (< and >).
```

LOAD XML acts as the complement of running the [mysql client](#) in XML output mode (that is, starting the client with the --xml option). To write data from a table to an XML file, use a command such as the following one from the system shell:

```
shell> mysql --xml -e 'SELECT * FROM mytable' > file.xml
```

To read the file back into a table, use LOAD XML INFILE. By default, the <row> element is considered to be the equivalent of a database table row; this can be changed using the ROWS IDENTIFIED BY clause.

This statement supports three different XML formats:

- Column names as attributes and column values as attribute values:

```
<row column1="value1" column2="value2" .../>
```

- Column names as tags and column values as the content of these tags:

```
<row>  
  <column1>value1</column1>  
  <column2>value2</column2>  
</row>
```

- Column names are the name attributes of <field> tags, and values are the contents of these tags:

```
<row>  
  <field name='column1'>value1</field>  
  <field name='column2'>value2</field>  
</row>
```

This is the format used by other tools, such as [mysqldump](#).

All 3 formats can be used in the same XML file; the import routine automatically detects the format for each row and interprets it correctly. Tags are matched based on the tag or attribute name and the column name.

The following clauses work essentially the same way for LOAD XML as they do for LOAD DATA:

- LOW\_PRIORITY or CONCURRENT
- LOCAL
- REPLACE or IGNORE
- CHARACTER SET
- (column\_or\_user\_var,...)
- SET

See [LOAD DATA](#) for more information about these clauses.

The IGNORE number LINES or IGNORE number ROWS clause causes the first number rows in the XML file to be skipped. It is analogous to the LOAD DATA statement's IGNORE ... LINES clause.

If the

[LOW\\_PRIORITY](#)

keyword is used, insertions are delayed until no other clients are reading from the table. The CONCURRENT keyword allows the use of [concurrent inserts](#). These clauses cannot be specified together.

This statement activates INSERT [triggers](#).

## See also

- The [CONNECT](#) storage engine has an [XML table type](#).

## 1.1.4.2.4.4 LOAD\_FILE

### Syntax

```
LOAD_FILE(file_name)
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Reads the file and returns the file contents as a string. To use this function, the file must be located on the server host, you must specify the full path name to the file, and you must have the FILE privilege. The file must be readable by all and it must be less than the size, in bytes, of the `max_allowed_packet` system variable. If the `secure_file_priv` system variable is set to a non-empty directory name, the file to be loaded must be located in that directory.

If the file does not exist or cannot be read because one of the preceding conditions is not satisfied, the function returns NULL.

Since MariaDB 5.1, the `character_set_filesystem` system variable has controlled interpretation of file names that are given as literal strings.

Statements using the `LOAD_FILE()` function are not [safe for statement based replication](#). This is because the slave will execute the `LOAD_FILE()` command itself. If the file doesn't exist on the slave, the function will return NULL.

## Examples

```
UPDATE t SET blob_col=LOAD_FILE('/tmp/picture') WHERE id=1;
```

## See Also

- [SELECT INTO DUMPFILE](#)

## 1.1.4.2.5 Concurrent Inserts

### Contents

1. [Notes](#)
2. [See Also](#)

The MyISAM storage engine supports concurrent inserts. This feature allows `SELECT` statements to be executed during `INSERT` operations, reducing contention.

Whether concurrent inserts can be used or not depends on the value of the `concurrent_insert` server system variable:

- NEVER  
(0) disables concurrent inserts.
- AUTO  
(1) allows concurrent inserts only when the target table has no free blocks (no data in the middle of the table has been deleted after the last `OPTIMIZE TABLE`). This is the default.
- ALWAYS  
(2) always enables concurrent inserts, in which case new rows are added at the end of a table if the table is being used by another thread.

If the `binary log` is used, `CREATE TABLE ... SELECT` and `INSERT ... SELECT` statements cannot use concurrent inserts. These statements acquire a read lock on the table, so concurrent inserts will need to wait. This way the log can be safely used to restore data.

Concurrent inserts are not used by replicas with the row based `replication` (see [binary log formats](#)).

If an `INSERT` statement contain the `HIGH_PRIORITY` clause, concurrent inserts cannot be used. `INSERT ... DELAYED` is usually unneeded if concurrent inserts are enabled.

`LOAD DATA INFILE` uses concurrent inserts if the

CONCURRENT  
keyword is specified and `concurrent_insert` is not  
NEVER  
. This makes the statement slower (even if no other sessions access the table) but reduces contention.

`LOCK TABLES` allows non-conflicting concurrent inserts if a

READ LOCAL  
lock is used. Concurrent inserts are not allowed if the  
LOCAL  
keyword is omitted.

## Notes

The decision to enable concurrent insert for a table is done when the table is opened. If you change the value of `concurrent_insert` it will only affect new opened tables. If you want it to work for also for tables in use or cached, you should do `FLUSH TABLES` after setting the variable.

## See Also

- [INSERT](#)
- [INSERT DELAYED](#)
- [INSERT SELECT](#)
- [HIGH\\_PRIORITY and LOW\\_PRIORITY](#)
- [INSERT - Default & Duplicate Values](#)
- [INSERT IGNORE](#)
- [INSERT ON DUPLICATE KEY UPDATE](#)

## 1.1.4.2.6 HIGH\_PRIORITY and LOW\_PRIORITY

### Contents

1. [See Also](#)

The `InnoDB` storage engine uses row-level locking to ensure data integrity. However some storage engines (such as `MEMORY`, `MyISAM`, `Aria` and `MERGE`) lock the whole table to prevent conflicts. These storage engines use two separate queues to remember pending statements; one is for `SELECTs` and the other one is for write statements (`INSERT`, `DELETE`, `UPDATE`). By default, the latter has a higher priority.

To give write operations a lower priority, the `low_priority_updates` server system variable can be set to

`ON`

. The option is available on both the global and session levels, and it can be set at startup or via the `SET` statement.

When too many table locks have been set by write statements, some pending `SELECTs` are executed. The maximum number of write locks that can be acquired before this happens is determined by the `max_write_lock_count` server system variable, which is dynamic.

If write statements have a higher priority (default), the priority of individual write statements (`INSERT`, `REPLACE`, `UPDATE`, `DELETE`) can be changed via the

`LOW_PRIORITY`  
attribute, and the priority of a  
`SELECT`  
statement can be raised via the  
`HIGH_PRIORITY`  
attribute. Also, `LOCK TABLES` supports a  
`LOW_PRIORITY`  
attribute for  
`WRITE`  
locks.

If read statements have a higher priority, the priority of an

`INSERT`  
can be changed via the  
`HIGH_PRIORITY`  
attribute. However, the priority of other write statements cannot be raised individually.

The use of

`LOW_PRIORITY`  
or  
`HIGH_PRIORITY`  
for an  
`INSERT`  
prevents `Concurrent Inserts` from being used.

## See Also

- [INSERT](#)
- [INSERT DELAYED](#)
- [INSERT SELECT](#)
- [Concurrent Inserts](#)
- [INSERT - Default & Duplicate Values](#)
- [INSERT IGNORE](#)
- [INSERT ON DUPLICATE KEY UPDATE](#)

## 1.1.4.2.7 IGNORE

## 1.1.4.2.8 INSERT - Default & Duplicate Values

## Contents

1. Default Values
2. Duplicate Values
3. See Also

## Default Values

If the

```
SQL_MODE  
contains  
STRICT_TRANS_TABLES  
and you are inserting into a transactional table (like InnoDB), or if the SQL_MODE contains  
STRICT_ALL_TABLES  
, all  
NOT NULL  
columns which does not have a  
DEFAULT  
value (and is not AUTO_INCREMENT ) must be explicitly referenced in  
INSERT  
statements. If not, an error like this is produced:
```

```
ERROR 1364 (HY000): Field 'col' doesn't have a default value
```

In all other cases, if a

```
NOT NULL  
column without a  
DEFAULT  
value is not referenced, an empty value will be inserted (for example, 0 for  
INTEGER  
columns and " " for  
CHAR  
columns). See NULL Values in MariaDB:Inserting for examples.
```

If a

```
NOT NULL  
column having a  
DEFAULT  
value is not referenced,  
NULL  
will be inserted.
```

If a

```
NULL  
column having a  
DEFAULT  
value is not referenced, its default value will be inserted. It is also possible to explicitly assign the default value using the  
DEFAULT  
keyword or the
```

```
DEFAULT()
```

function.

If the

```
DEFAULT  
keyword is used but the column does not have a  
DEFAULT  
value, an error like this is produced:
```

```
ERROR 1364 (HY000): Field 'col' doesn't have a default value
```

## Duplicate Values

By default, if you try to insert a duplicate row and there is a

```
UNIQUE  
index,  
INSERT
```

stops and an error like this is produced:

```
ERROR 1062 (23000): Duplicate entry 'dup_value' for key 'col'
```

To handle duplicates you can use the [IGNORE](#) clause, [INSERT ON DUPLICATE KEY UPDATE](#) or the [REPLACE](#) statement. Note that the IGNORE and DELAYED options are ignored when you use [ON DUPLICATE KEY UPDATE](#).

## See Also

- [INSERT](#)
- [INSERT DELAYED](#)
- [INSERT SELECT](#)
- [HIGH\\_PRIORITY](#) and [LOW\\_PRIORITY](#)
- [Concurrent Inserts](#)
- [INSERT IGNORE](#)
- [INSERT ON DUPLICATE KEY UPDATE](#)

## 1.1.4.2.9 INSERT IGNORE

### Contents

1. [Ignoring Errors](#)
2. [Examples](#)
3. [See Also](#)

## Ignoring Errors

Normally [INSERT](#) stops and rolls back when it encounters an error.

By using the [IGNORE](#) keyword all errors are converted to warnings, which will not stop inserts of additional rows.

The IGNORE and DELAYED options are ignored when you use [ON DUPLICATE KEY UPDATE](#).

Prior to MySQL and [MariaDB 5.5.28](#), no warnings were issued for duplicate key errors when using

```
IGNORE
. You can get the old behavior if you set OLD_MODE to
NO_DUP_KEY_WARNINGS_WITH_IGNORE
```

## Examples

```

CREATE TABLE t1 (x INT UNIQUE);

INSERT INTO t1 VALUES(1),(2);

INSERT INTO t1 VALUES(2),(3);
ERROR 1062 (23000): Duplicate entry '2' for key 'x'
SELECT * FROM t1;
+---+
| x |
+---+
| 1 |
| 2 |
+---+
2 rows in set (0.00 sec)

INSERT IGNORE INTO t1 VALUES(2),(3);
Query OK, 1 row affected, 1 warning (0.04 sec)

SHOW WARNINGS;
+-----+
| Level | Code | Message |
+-----+
| Warning | 1062 | Duplicate entry '2' for key 'x' |
+-----+

SELECT * FROM t1;
+---+
| x |
+---+
| 1 |
| 2 |
| 3 |
+---+

```

See [INSERT ON DUPLICATE KEY UPDATE](#) for further examples using that syntax.

## See Also

- [INSERT](#)
- [INSERT DELAYED](#)
- [INSERT SELECT](#)
- [HIGH\\_PRIORITY and LOW\\_PRIORITY](#)
- [Concurrent Inserts](#)
- [INSERT - Default & Duplicate Values](#)
- [INSERT IGNORE](#)
- [INSERT ON DUPLICATE KEY UPDATE](#)

## 1.1.4.2.10 INSERT ON DUPLICATE KEY UPDATE

### Syntax

```

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ]

```

Or:

```

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)]
SET col={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ]

```

Or:

```

INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
SELECT ...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ]

```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

INSERT ... ON DUPLICATE KEY UPDATE is a MariaDB/MySQL extension to the [INSERT](#) statement that, if it finds a duplicate unique or primary key, will instead perform an [UPDATE](#).

The row/s affected value is reported as 1 if a row is inserted, and 2 if a row is updated, unless the API's

```
CLIENT_FOUND_ROWS
flag is set.
```

If more than one unique index is matched, only the first is updated. It is not recommended to use this statement on tables with more than one unique index.

If the table has an [AUTO\\_INCREMENT](#) primary key and the statement inserts or updates a row, the [LAST\\_INSERT\\_ID\(\)](#) function returns its AUTO\_INCREMENT value.

The [VALUES\(\)](#) function can only be used in a

```
ON DUPLICATE KEY UPDATE
clause and has no meaning in any other context. It returns the column values from the
INSERT
portion of the statement. This function is particularly useful for multi-rows inserts.
```

The [IGNORE](#) and [DELAYED](#) options are ignored when you use

```
ON DUPLICATE KEY UPDATE
```

See [Partition Pruning and Selection](#) for details on the PARTITION clause.

This statement activates INSERT and UPDATE triggers. See [Trigger Overview](#) for details.

See also a similar statement, [REPLACE](#).

## Examples

```

CREATE TABLE ins_duplicate (id INT PRIMARY KEY, animal VARCHAR(30));
INSERT INTO ins_duplicate VALUES (1,'Aardvark'), (2,'Cheetah'), (3,'Zebra');

```

If there is no existing key, the statement runs as a regular INSERT:

```

INSERT INTO ins_duplicate VALUES (4,'Gorilla')
  ON DUPLICATE KEY UPDATE animal='Gorilla';
Query OK, 1 row affected (0.07 sec)

```

```

SELECT * FROM ins_duplicate;
+----+-----+
| id | animal |
+----+-----+
| 1 | Aardvark |
| 2 | Cheetah |
| 3 | Zebra   |
| 4 | Gorilla |
+----+-----+

```

A regular INSERT with a primary key value of 1 will fail, due to the existing key:

```

INSERT INTO ins_duplicate VALUES (1,'Antelope');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

```

However, we can use an INSERT ON DUPLICATE KEY UPDATE instead:

```
INSERT INTO ins_duplicate VALUES (1,'Antelope')
  ON DUPLICATE KEY UPDATE animal='Antelope';
Query OK, 2 rows affected (0.09 sec)
```

Note that there are two rows reported as affected, but this refers only to the UPDATE.

```
SELECT * FROM ins_duplicate;
+-----+
| id | animal |
+-----+
| 1 | Antelope |
| 2 | Cheetah |
| 3 | Zebra |
| 4 | Gorilla |
+-----+
```

Adding a second unique column:

```
ALTER TABLE ins_duplicate ADD id2 INT;
UPDATE ins_duplicate SET id2=id+10;
ALTER TABLE ins_duplicate ADD UNIQUE KEY(id2);
```

Where two rows match the unique keys match, only the first is updated. This can be unsafe and is not recommended unless you are certain what you are doing.

```
INSERT INTO ins_duplicate VALUES (2,'Lion',13)
  ON DUPLICATE KEY UPDATE animal='Lion';
Query OK, 2 rows affected (0.004 sec)

SELECT * FROM ins_duplicate;
+-----+
| id | animal | id2 |
+-----+
| 1 | Antelope | 11 |
| 2 | Lion | 12 |
| 3 | Zebra | 13 |
| 4 | Gorilla | 14 |
+-----+
```

Although the third row with an id of 3 has an id2 of 13, which also matched, it was not updated.

Changing id to an auto\_increment field. If a new row is added, the auto\_increment is moved forward. If the row is updated, it remains the same.

```

ALTER TABLE `ins_duplicate` CHANGE `id` `id` INT( 11 ) NOT NULL AUTO_INCREMENT;
ALTER TABLE ins_duplicate DROP id2;
SELECT Auto_increment FROM INFORMATION_SCHEMA.TABLES
  WHERE TABLE_NAME='ins_duplicate';
+-----+
| Auto_increment |
+-----+
|      5 |
+-----+

INSERT INTO ins_duplicate VALUES (2,'Leopard')
  ON DUPLICATE KEY UPDATE animal='Leopard';
Query OK, 2 rows affected (0.00 sec)

SELECT Auto_increment FROM INFORMATION_SCHEMA.TABLES
  WHERE TABLE_NAME='ins_duplicate';
+-----+
| Auto_increment |
+-----+
|      5 |
+-----+

INSERT INTO ins_duplicate VALUES (5,'Wild Dog')
  ON DUPLICATE KEY UPDATE animal='Wild Dog';
Query OK, 1 row affected (0.09 sec)

SELECT * FROM ins_duplicate;
+---+---+
| id | animal   |
+---+---+
| 1 | Antelope |
| 2 | Leopard   |
| 3 | Zebra     |
| 4 | Gorilla   |
| 5 | Wild Dog  |
+---+---+

SELECT Auto_increment FROM INFORMATION_SCHEMA.TABLES
  WHERE TABLE_NAME='ins_duplicate';
+-----+
| Auto_increment |
+-----+
|      6 |
+-----+

```

Referring to column values from the INSERT portion of the statement:

```

INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)
  ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);

```

See the [VALUES\(\)](#) function for more.

## See Also

- [INSERT](#)
- [INSERT DELAYED](#)
- [INSERT SELECT](#)
- [HIGH\\_PRIORITY and LOW\\_PRIORITY](#)
- [Concurrent Inserts](#)
- [INSERT - Default & Duplicate Values](#)
- [INSERT IGNORE](#)
- [VALUES\(\)](#)

## 1.1.4.2.11 INSERT...RETURNING

MariaDB starting with 10.5.0

INSERT ... RETURNING was added in [MariaDB 10.5.0](#), and returns a resultset of the inserted rows.

## Syntax

```

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ] [RETURNING select_expr
  [, select_expr ...]]

```

Or:

```

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)]
SET col={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ] [RETURNING select_expr
  [, select_expr ...]]

```

Or:

```

INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
SELECT ...
[ ON DUPLICATE KEY UPDATE
  col=expr
  [, col=expr] ... ] [RETURNING select_expr
  [, select_expr ...]]

```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

```

INSERT ... RETURNING
returns a resultset of the inserted rows.

```

This returns the listed columns for all the rows that are inserted, or alternatively, the specified SELECT expression. Any SQL expressions which can be calculated can be used in the select expression for the RETURNING clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

## Examples

Simple INSERT statements:

```

CREATE OR REPLACE TABLE t2 (id INT, animal VARCHAR(20), t TIMESTAMP);

INSERT INTO t2 (id) VALUES (2),(3) RETURNING id,t;
+-----+-----+
| id   | t      |
+-----+-----+
| 2    | 2021-04-28 00:59:32 |
| 3    | 2021-04-28 00:59:32 |
+-----+-----+

```

```

INSERT INTO t2(id,animal) VALUES (1,'Dog'),(2,'Lion'),(3,'Tiger'),(4,'Leopard')
  RETURNING id,id+id,id&id,id||id;
+-----+-----+-----+-----+
| id   | id+id | id&id | id||id |
+-----+-----+-----+-----+
| 1    | 2     | 1     | 1    |
| 2    | 4     | 2     | 1    |
| 3    | 6     | 3     | 1    |
| 4    | 8     | 4     | 1    |
+-----+-----+-----+-----+

```

```

DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
BEGIN
    RETURN (SELECT arg+arg);
END |

DELIMITER ;

PREPARE stmt FROM "INSERT INTO t1 SET id1=1, animal1='Bear' RETURNING f(id1), UPPER(animal1)";

EXECUTE stmt;
+-----+
| f(id1) | UPPER(animal1) |
+-----+
|     2   | BEAR           |
+-----+

```

Subqueries in the RETURNING clause that return more than one row or column cannot be used.

Aggregate functions cannot be used in the RETURNING clause. Since aggregate functions work on a set of values, and if the purpose is to get the row count, ROW\_COUNT() with SELECT can be used or it can be used in INSERT...SELECT...RETURNING if the table in the RETURNING clause is not the same as the INSERT table.

## See Also

- [INSERT](#)
- [REPLACE ... RETURNING](#)
- [DELETE ... RETURNING](#)
- [Returning clause \(video\)](#)

### 1.1.4.3 Changing and Deleting Data

#### 1.1.4.3.1 DELETE

#### 1.1.4.3.2 HIGH\_PRIORITY and LOW\_PRIORITY

#### 1.1.4.3.3 IGNORE

#### 1.1.4.3.4 REPLACE

#### 1.1.4.3.5 REPLACE...RETURNING

MariaDB starting with [10.5.0](#)

REPLACE ... RETURNING was added in [MariaDB 10.5.0](#), and returns a resultset of the replaced rows.

## Syntax

```

REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
{VALUES | VALUE} ({expr | DEFAULT},...),(...),...
[RETURNING select_expr
[, select_expr ...]]

```

Or:

```

REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name [PARTITION (partition_list)]
SET col={expr | DEFAULT}, ...
[RETURNING select_expr
[, select_expr ...]]

```

Or:

```

REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name [PARTITION (partition_list)] [(col,...)]
SELECT ...
[RETURNING select_expr
[, select_expr ...]]

```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

```

REPLACE ... RETURNING
returns a resultset of the replaced rows.

```

This returns the listed columns for all the rows that are replaced, or alternatively, the specified SELECT expression. Any SQL expressions which can be calculated can be used in the select expression for the RETURNING clause, including virtual columns and aliases, expressions which use various operators such as bitwise, logical and arithmetic operators, string functions, date-time functions, numeric functions, control flow functions, secondary functions and stored functions. Along with this, statements which have subqueries and prepared statements can also be used.

## Examples

Simple REPLACE statement

```

REPLACE INTO t2 VALUES (1,'Leopard'),(2,'Dog') RETURNING id2, id2+id2
as Total ,id2|id2, id2&&id2;
+-----+-----+-----+
| id2 | Total | id2|id2 | id2&&id2 |
+-----+-----+-----+
|  1 |      2 |      1 |      1 |
|  2 |      4 |      2 |      1 |
+-----+-----+-----+

```

Using stored functions in RETURNING

```

DELIMITER |
CREATE FUNCTION f(arg INT) RETURNS INT
BEGIN
    RETURN (SELECT arg+arg);
END |

DELIMITER ;
PREPARE stmt FROM "REPLACE INTO t2 SET id2=3, animal2='Fox' RETURNING f2(id2),
UPPER(animal2)";

EXECUTE stmt;
+-----+
| f2(id2) | UPPER(animal2) |
+-----+
|      6 | FOX             |
+-----+

```

Subqueries in the statement

```

REPLACE INTO t1 SELECT * FROM t2 RETURNING (SELECT id2 FROM t2 WHERE
id2 IN (SELECT id2 FROM t2 WHERE id2=1)) AS new_id;
+-----+
| new_id |
+-----+
|      1 |
|      1 |
|      1 |
|      1 |
+-----+

```

Subqueries in the RETURNING clause that return more than one row or column cannot be used..

Aggregate functions cannot be used in the RETURNING clause. Since aggregate functions work on a set of values and if the purpose is to get the row

count, ROW\_COUNT() with SELECT can be used, or it can be used in REPLACE...SELECT...RETURNING if the table in the RETURNING clause is not the same as the REPLACE table.

## See Also

- [INSERT ... RETURNING](#)
- [DELETE ... RETURNING](#)
- [Returning clause \(video\)](#)

## 1.1.4.3.6 TRUNCATE TABLE

## 1.1.4.3.7 UPDATE

## 1.1.5 Prepared Statements

### 1.1.5.1 PREPARE Statement

#### Syntax

```
PREPARE stmt_name FROM preparable_stmt
```

#### Contents

1. [Syntax](#)
2. [Description](#)
  - 1. Oracle Mode
3. [Permitted Statements](#)
4. [Example](#)
5. [See Also](#)

## Description

The

```
PREPARE  
statement prepares a statement and assigns it a name,  
stmt_name  
, by which to refer to the statement later. Statement names are not case sensitive.  
preparable_stmt
```

is either a string literal or a [user variable](#) (not a [local variable](#), an SQL expression or a subquery) that contains the text of the statement. The text must represent a single SQL statement, not multiple statements. Within the statement, "?" characters can be used as parameter markers to indicate where data values are to be bound to the query later when you execute it. The "?" characters should not be enclosed within quotes, even if you intend to bind them to string values. Parameter markers can be used only where expressions should appear, not for SQL keywords, identifiers, and so forth.

The scope of a prepared statement is the session within which it is created. Other sessions cannot see it.

If a prepared statement with the given name already exists, it is deallocated implicitly before the new statement is prepared. This means that if the new statement contains an error and cannot be prepared, an error is returned and no statement with the given name exists.

Prepared statements can be PREPARED and EXECUTED in a stored procedure, but not in a stored function or trigger. Also, even if the statement is PREPARED in a procedure, it will not be deallocated when the procedure execution ends.

A prepared statement can access [user-defined variables](#), but not [local variables](#) or procedure's parameters.

If the prepared statement contains a syntax error, PREPARE will fail. As a side effect, stored procedures can use it to check if a statement is valid. For example:

```
CREATE PROCEDURE `test_stmt` (IN sql_text TEXT)  
BEGIN  
    DECLARE EXIT HANDLER FOR SQLEXCEPTION  
    BEGIN  
        SELECT CONCAT(sql_text, ' is not valid');  
    END;  
    SET @SQL := sql_text;  
    PREPARE stmt FROM @SQL;  
    DEALLOCATE PREPARE stmt;  
END;
```

The [FOUND\\_ROWS\(\)](#) and [ROW\\_COUNT\(\)](#) functions, if called immediately after EXECUTE, return the number of rows read or affected by the prepared statements; however, if they are called after DEALLOCATE PREPARE, they provide information about this statement. If the prepared statement produces

errors or warnings, [GET DIAGNOSTICS](#) return information about them. DEALLOCATE PREPARE shouldn't clear the [diagnostics area](#), unless it produces an error.

A prepared statement is executed with

[EXECUTE](#)

and released with

[DEALLOCATE PREPARE](#)

The [max\\_prepared\\_stmt\\_count](#) server system variable determines the number of allowed prepared statements that can be prepared on the server. If it is set to 0, prepared statements are not allowed. If the limit is reached, an error similar to the following will be produced:

```
ERROR 1461 (42000): Can't create more than max_prepared_stmt_count statements  
(current value: 0)
```

## Oracle Mode

MariaDB starting with 10.3

In [Oracle mode from MariaDB 10.3](#),

```
PREPARE stmt FROM 'SELECT :1, :2'  
is used, instead of  
?
```

## Permitted Statements

Not all statements can be prepared. Only the following SQL commands are permitted:

- [ALTER TABLE](#)
- [ANALYZE TABLE](#)
- [BINLOG](#)
- [CACHE INDEX](#)
- [CALL](#)
- [CHANGE MASTER](#)
- [CHECKSUM {TABLE | TABLES}](#)
- [COMMIT](#)
- { [CREATE | DROP](#)  } [DATABASE](#)
- { [CREATE | DROP](#)  } [INDEX](#)
- { [CREATE | RENAME | DROP](#)  } [TABLE](#)
- { [CREATE | RENAME | DROP](#)  } [USER](#)
- { [CREATE | DROP](#)  } [VIEW](#)
- [DELETE](#)
- [DESCRIBE](#)
- [DO](#)
- [EXPLAIN](#)
- [FLUSH {TABLE | TABLES | TABLES WITH READ LOCK | HOSTS | PRIVILEGES | LOGS | STATUS | MASTER | SLAVE | DES\\_KEY\\_FILE | USER\\_RESOURCES | QUERY CACHE | TABLE\\_STATISTICS | INDEX\\_STATISTICS | USER\\_STATISTICS | CLIENT\\_STATISTICS}](#)
- [GRANT](#)
- [INSERT](#)
- [INSTALL { PLUGIN | SONAME }](#)
- [HANDLER READ](#)
- [KILL](#)
- [LOAD INDEX INTO CACHE](#)
- [OPTIMIZE TABLE](#)
- [REPAIR TABLE](#)
- [REPLACE](#)
- [RESET { MASTER | SLAVE | QUERY CACHE }](#)
- [REVOKE](#)
- [ROLLBACK](#)
- [SELECT](#)
- [SET](#)
- [SET GLOBAL SQL\\_SLAVE\\_SKIP\\_COUNTER](#)
- [SET ROLE](#)
- [SET SQL\\_LOG\\_BIN](#)
- [SET TRANSACTION ISOLATION LEVEL](#)
- [SHOW EXPLAIN](#)

- SHOW { DATABASES | TABLES | OPEN TABLES | TABLE STATUS | COLUMNS | INDEX | TRIGGERS | EVENTS | GRANTS | CHARACTER SET | COLLATION | ENGINES | PLUGINS [SONAME] | PRIVILEGES | PROCESSLIST | PROFILE | PROFILES | VARIABLES | STATUS | WARNINGS | ERRORS | TABLE\_STATISTICS | INDEX\_STATISTICS | USER\_STATISTICS | CLIENT\_STATISTICS | AUTHORS | CONTRIBUTORS }
- SHOW CREATE { DATABASE | TABLE | VIEW | PROCEDURE | FUNCTION | TRIGGER | EVENT }
- SHOW { FUNCTION | PROCEDURE } CODE
- SHOW BINLOG EVENTS
- SHOW SLAVE HOSTS
- SHOW {MASTER | BINARY} LOGS
- SHOW {MASTER | SLAVE | TABLES | INNODB | FUNCTION | PROCEDURE} STATUS
- SLAVE { START | STOP }
- TRUNCATE TABLE
- SHUTDOWN
- UNINSTALL {PLUGIN | SONAME}
- UPDATE

Synonyms are not listed here, but can be used. For example, DESC can be used instead of DESCRIBE.

MariaDB starting with 10.1.1

Compound statements can be prepared too.

Note that if a statement can be run in a stored routine, it will work even if it is called by a prepared statement. For example, SIGNAL can't be directly prepared. However, it is allowed in [stored routines](#). If the x() procedure contains SIGNAL, you can still prepare and execute the 'CALL x();' prepared statement.

MariaDB starting with 10.2.3

PREPARE now supports most kinds of expressions as well, for example:

```
PREPARE stmt FROM CONCAT('SELECT * FROM ', table_name);
```

MariaDB starting with 10.6.2

All statements can be prepared, except PREPARE , EXECUTE , and DEALLOCATE / DROP PREPARE .

When PREPARE is used with a statement which is not supported, the following error is produced:

```
ERROR 1295 (HY000): This command is not supported in the prepared statement protocol yet
```

## Example

```
create table t1 (a int,b char(10));
insert into t1 values (1,"one"),(2, "two"),(3,"three");
prepare test from "select * from t1 where a=?";
set @param=2;
execute test using @param;
+-----+
| a    | b    |
+-----+
|  2  | two |
+-----+
set @param=3;
execute test using @param;
+-----+
| a    | b    |
+-----+
|  3  | three |
+-----+
deallocate prepare test;
```

Since identifiers are not permitted as prepared statements parameters, sometimes it is necessary to dynamically compose an SQL statement. This technique is called *dynamic SQL* ). The following example shows how to use dynamic SQL:

```

CREATE PROCEDURE teststmt_test(IN tab_name VARCHAR(64))
BEGIN
    SET @sql = CONCAT('SELECT COUNT(*) FROM ', tab_name);
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END;

CALL teststmt_test('mysql.user');
+-----+
| COUNT(*) |
+-----+
|      4   |
+-----+

```

Use of variables in prepared statements:

```

PREPARE stmt FROM 'SELECT @x;';

SET @x = 1;

EXECUTE stmt;
+-----+
| @x   |
+-----+
|    1  |
+-----+

SET @x = 0;

EXECUTE stmt;
+-----+
| @x   |
+-----+
|    0  |
+-----+

DEALLOCATE PREPARE stmt;

```

## See Also

- [Out parameters in PREPARE](#)
- [EXECUTE Statement](#)
- [DEALLOCATE / DROP Prepared Statement](#)
- [EXECUTE IMMEDIATE](#)
- [Oracle mode from MariaDB 10.3](#)

## 1.1.5.2 Out Parameters in PREPARE

MariaDB 10.1.1

Out parameters in PREPARE were only available in [MariaDB 10.1.1](#)

One can use question mark placeholders for out-parameters in the PREPARE statement. Only SELECT ... INTO can be used this way:

```

prepare test from "select id into ? from t1 where val=?";
execute test using @out, @in;

```

This is particularly convenient when used with [compound statements](#) :

```

PREPARE stmt FROM "BEGIN NOT ATOMIC
DECLARE v_res INT;
SELECT COUNT(*) INTO v_res FROM t1;
SELECT 'Hello World', v_res INTO ?,?;
END" |

```

## 1.1.5.3 EXECUTE STATEMENT

## 1.1.5.4 DEALLOCATE / DROP PREPARE

### Syntax

```
{DEALLOCATE | DROP} PREPARE stmt_name
```

### Description

To deallocate a prepared statement produced with

PREPARE

, use a  
DEALLOCATE PREPARE  
statement that refers to the prepared statement name.

A prepared statement is implicitly deallocated when a new

PREPARE  
command is issued. In that case, there is no need to use  
DEALLOCATE

Attempting to execute a prepared statement after deallocating it results in an error, as if it was not prepared at all:

```
ERROR 1243 (HY000): Unknown prepared statement handler (stmt_name) given to EXECUTE
```

If the specified statement has not been PREPARED, an error similar to the following will be produced:

```
ERROR 1243 (HY000): Unknown prepared statement handler (stmt_name) given to DEALLOCATE PREPARE
```

### Example

See [example in PREPARE](#).

### See Also

- [PREPARE Statement](#)
- [EXECUTE Statement](#)
- [EXECUTE IMMEDIATE](#)

## 1.1.5.5 EXECUTE IMMEDIATE

MariaDB starting with [10.2.3](#)

EXECUTE IMMEDIATE was introduced in [MariaDB 10.2.3](#).

### Syntax

```
EXECUTE IMMEDIATE statement
```

### Description

EXECUTE IMMEDIATE  
executes a dynamic SQL statement created on the fly, which can reduce performance overhead.

For example:

```
EXECUTE IMMEDIATE 'SELECT 1'
```

which is shorthand for:

```
prepare stmt from "select 1";
execute stmt;
deallocate prepare stmt;
```

EXECUTE IMMEDIATE supports complex expressions as prepare source and parameters:

```
EXECUTE IMMEDIATE CONCAT('SELECT COUNT(*) FROM ', 't1', ' WHERE a=?') USING 5+5;
```

Limitations: subselects and stored function calls are not supported as a prepare source.

The following examples return an error:

```
CREATE OR REPLACE FUNCTION f1() RETURNS VARCHAR(64) RETURN 'SELECT * FROM t1';
EXECUTE IMMEDIATE f1();
ERROR 1970 (42000): EXECUTE IMMEDIATE does not support subqueries or stored functions

EXECUTE IMMEDIATE (SELECT 'SELECT * FROM t1');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use near
'SELECT 'SELECT * FROM t1'' at line 1

CREATE OR REPLACE FUNCTION f1() RETURNS INT RETURN 10;
EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING f1();
ERROR 1970 (42000): EXECUTE..USING does not support subqueries or stored functions

EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING (SELECT 10);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use near
'SELECT 10)' at line 1
```

One can use a user or an SP variable as a workaround:

```
CREATE OR REPLACE FUNCTION f1() RETURNS VARCHAR(64) RETURN 'SELECT * FROM t1';
SET @stmt=f1();
EXECUTE IMMEDIATE @stmt;

SET @stmt=(SELECT 'SELECT 1');
EXECUTE IMMEDIATE @stmt;

CREATE OR REPLACE FUNCTION f1() RETURNS INT RETURN 10;
SET @param=f1();
EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING @param;

SET @param=(SELECT 10);
EXECUTE IMMEDIATE 'SELECT * FROM t1 WHERE a=?' USING @param;
```

EXECUTE IMMEDIATE supports user variables and SP variables as OUT parameters

```
DELIMITER $$ 
CREATE OR REPLACE PROCEDURE p1(OUT a INT)
BEGIN
    SET a:= 10;
END;
$$
DELIMITER ;
SET @a=2;
EXECUTE IMMEDIATE 'CALL p1(?)' USING @a;
SELECT @a;
+----+
| @a |
+----+
| 10 |
+----+
```

Similar to PREPARE, EXECUTE IMMEDIATE is allowed in stored procedures but is not allowed in stored functions.

This example uses EXECUTE IMMEDIATE inside a stored procedure:

```

DELIMITER $$  

CREATE OR REPLACE PROCEDURE p1()  

BEGIN  

    EXECUTE IMMEDIATE 'SELECT 1';  

END;  

$$  

DELIMITER ;
CALL p1;
+---+
| 1 |
+---+
| 1 |
+---+

```

This script returns an error:

```

DELIMITER $$  

CREATE FUNCTION f1() RETURNS INT  

BEGIN  

    EXECUTE IMMEDIATE 'DO 1';  

    RETURN 1;  

END;  

$$  

ERROR 1336 (0A000): Dynamic SQL is not allowed in stored function or trigger

```

EXECUTE IMMEDIATE can use DEFAULT and IGNORE indicators as bind parameters:

```

CREATE OR REPLACE TABLE t1 (a INT DEFAULT 10);
EXECUTE IMMEDIATE 'INSERT INTO t1 VALUES (?)' USING DEFAULT;
SELECT * FROM t1;
+---+
| a |
+---+
| 10 |
+---+

```

EXECUTE IMMEDIATE increments the [Com\\_execute\\_immediate](#) status variable, as well as the [Com\\_stmt\\_prepare](#), [Com\\_stmt\\_execute](#) and [Com\\_stmt\\_close](#) status variables.

Note, EXECUTE IMMEDIATE does not increment the [Com\\_execute\\_sql](#) status variable. [Com\\_execute\\_sql](#) is used only for [PREPARE .. EXECUTE](#).

This session screenshot demonstrates how EXECUTE IMMEDIATE affects status variables:

```

SELECT * FROM INFORMATION_SCHEMA.SESSION_STATUS WHERE VARIABLE_NAMERLIKE
('COM_(EXECUTE|STMT_PREPARE|STMT_EXECUTE|STMT_CLOSE)');

+-----+-----+
| VARIABLE_NAME | VARIABLE_VALUE |  

+-----+-----+
| COM_EXECUTE_IMMEDIATE | 0 |  

| COM_EXECUTE_SQL | 0 |  

| COM_STMT_CLOSE | 0 |  

| COM_STMT_EXECUTE | 0 |  

| COM_STMT_PREPARE | 0 |  

+-----+-----+  
  

EXECUTE IMMEDIATE 'SELECT 1';
+---+
| 1 |
+---+
| 1 |
+---+  
  

SELECT * FROM INFORMATION_SCHEMA.SESSION_STATUS WHERE VARIABLE_NAMERLIKE
('COM_(EXECUTE|STMT_PREPARE|STMT_EXECUTE|STMT_CLOSE)');
+-----+-----+
| VARIABLE_NAME | VARIABLE_VALUE |  

+-----+-----+
| COM_EXECUTE_IMMEDIATE | 1 |  

| COM_EXECUTE_SQL | 0 |  

| COM_STMT_CLOSE | 1 |  

| COM_STMT_EXECUTE | 1 |  

| COM_STMT_PREPARE | 1 |  

+-----+-----+

```

## 1.1.6 Programmatic and Compound Statements

### 1.1.6.1 Using Compound Statements Outside of Stored Programs

MariaDB starting with 10.1.1

Starting from MariaDB 10.1.1 compound statements can also be used outside of stored programs .

```
delimiter |
IF @have_innodb THEN
CREATE TABLE IF NOT EXISTS innodb_index_stats (
    database_name      VARCHAR(64) NOT NULL,
    table_name         VARCHAR(64) NOT NULL,
    index_name         VARCHAR(64) NOT NULL,
    last_update        TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    stat_name          VARCHAR(64) NOT NULL,
    stat_value         BIGINT UNSIGNED NOT NULL,
    sample_size        BIGINT UNSIGNED,
    stat_description   VARCHAR(1024) NOT NULL,
    PRIMARY KEY (database_name, table_name, index_name, stat_name)
) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8_bin STATS_PERSISTENT=0;
END IF|
Query OK, 0 rows affected, 2 warnings (0.00 sec)
```

Note, that using compound statements this way is subject to following limitations:

- Only `BEGIN` , `IF` , `CASE` , `LOOP` , `WHILE` , `REPEAT` statements may start a compound statement outside of stored programs.
- `BEGIN` must use the  
`BEGIN NOT ATOMIC`  
syntax (otherwise it'll be confused with `BEGIN` that starts a transaction).
- A compound statement might not start with a label.
- A compound statement is parsed completely — note "2 warnings" in the above example, even if the condition was false (InnoDB was, indeed, disabled), and the `CREATE TABLE` statement was not executed, it was still parsed and the parser produced "Unknown storage engine" warning.

Inside a compound block first three limitations do not apply, one can use anything that can be used inside a stored program — including labels, condition handlers, variables, and so on:

```
BEGIN NOT ATOMIC
DECLARE foo CONDITION FOR 1146;
DECLARE x INT DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SET x=1;
INSERT INTO test.t1 VALUES ("hdlr1", val, 2);
END|
```

Example how to use

```
IF
:
IF (1>0) THEN BEGIN NOT ATOMIC SELECT 1; END ; END IF;;
```

Example of how to use

```
WHILE
loop:
```

```
DELIMITER |
BEGIN NOT ATOMIC
DECLARE x INT DEFAULT 0;
WHILE x <= 10 DO
    SET x = x + 1;
    SELECT x;
END WHILE;
END|
DELIMITER ;
```

### 1.1.6.2 BEGIN END

# Syntax

```
[begin_label:] BEGIN [NOT ATOMIC]
[statement_list]
END [end_label]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

NOT ATOMIC

is required when used outside of a [stored procedure](#). Inside stored procedures or within an anonymous block,

BEGIN

alone starts a new anonymous block.

## Description

BEGIN ... END

syntax is used for writing compound statements. A compound statement can contain multiple statements, enclosed by the

BEGIN

and

END

keywords. statement\_list represents a list of one or more statements, each terminated by a semicolon (i.e.,

;

) statement delimiter. statement\_list is optional, which means that the empty compound statement (

BEGIN END

) is legal.

Note that

END

will perform a commit. If you are running in [autocommit](#) mode, every statement will be committed separately. If you are not running in autocommit

mode, you must execute a [COMMIT](#) or [ROLLBACK](#) after

END

to get the database up to date.

Use of multiple statements requires that a client is able to send statement strings containing the ; statement delimiter. This is handled in the [mysql](#) command-line client with the [DELIMITER](#) command. Changing the

```
;;
end-of-statement delimiter (for example, to
//)
) allows
;
to be used in a program body.
```

A compound statement within a [stored program](#) can be [labeled](#).

```
end_label
cannot be given unless
begin_label
also is present. If both are present, they must be the same.
```

BEGIN ... END

constructs can be nested. Each block can define its own variables, a

CONDITION

, a

HANDLER

and a [CURSOR](#), which don't exist in the outer blocks. The most local declarations override the outer objects which use the same name (see example below).

The declarations order is the following:

- DECLARE  
    local variables ;
- DECLARE CONDITION  
    s ;

- ```
DECLARE CURSOR
  S;
```
- ```
DECLARE HANDLER
  S;
```

Note that

```
DECLARE HANDLER
contains another
BEGIN ... END
construct.
```

Here is an example of a very simple, anonymous block:

```
BEGIN NOT ATOMIC
SET @a=1;
CREATE TABLE test.t1(a INT);
END |
```

Below is an example of nested blocks in a stored procedure:

```
CREATE PROCEDURE t()
BEGIN
  DECLARE x TINYINT UNSIGNED DEFAULT 1;
  BEGIN
    DECLARE x CHAR(2) DEFAULT '02';
    DECLARE y TINYINT UNSIGNED DEFAULT 10;
    SELECT x, y;
  END;
  SELECT x;
END;
```

In this example, a **TINYINT** variable,

x  
is declared in the outer block. But in the inner block  
x  
is re-declared as a **CHAR** and an  
y  
variable is declared. The inner **SELECT** shows the "new" value of  
x  
, and the value of  
y  
. But when x is selected in the outer block, the "old" value is returned. The final **SELECT** doesn't try to read  
y  
, because it doesn't exist in that context.

## See Also

- [Using compound statements outside of stored programs](#)
- [Changes in Oracle mode from MariaDB 10.3](#)

### 1.1.6.3 CASE Statement

#### Syntax

```
CASE case_value
  WHEN when_value THEN statement_list
  [WHEN when_value THEN statement_list] ...
  [ELSE statement_list]
END CASE
```

Or:

```
CASE
  WHEN search_condition THEN statement_list
  [WHEN search_condition THEN statement_list] ...
  [ELSE statement_list]
END CASE
```

# Description

The text on this page describes the

CASE  
statement for [stored programs](#). See the [CASE OPERATOR](#) for details on the CASE operator outside of stored programs.

The

CASE  
statement for [stored programs](#) implements a complex conditional construct. If a  
search\_condition  
evaluates to true, the corresponding SQL statement list is executed. If no search condition matches, the statement list in the  
ELSE  
clause is executed. Each  
statement\_list  
consists of one or more statements.

The

CASE  
statement cannot have an  
ELSE NULL  
clause, and it is terminated with  
END CASE  
instead of  
END  
. implements a complex conditional construct. If a  
search\_condition  
evaluates to true, the corresponding SQL statement list is executed. If no search condition matches, the statement list in the  
ELSE  
clause is executed. Each  
statement\_list  
consists of one or more statements.

If no when\_value or search\_condition matches the value tested and the

CASE  
statement contains no  
ELSE  
clause, a Case not found for  
CASE  
statement error results.

Each statement\_list consists of one or more statements; an empty

statement\_list  
is not allowed. To handle situations where no value is matched by any  
WHEN  
clause, use an  
ELSE  
containing an empty [BEGIN ... END](#) block, as shown in this example:

```
DELIMITER |
CREATE PROCEDURE p()
BEGIN
    DECLARE v INT DEFAULT 1;
    CASE v
        WHEN 2 THEN SELECT v;
        WHEN 3 THEN SELECT 0;
        ELSE BEGIN END;
    END CASE;
END;
|
```

The indentation used here in the

ELSE  
clause is for purposes of clarity only, and is not otherwise significant. See [Delimiters in the mysql client](#) for more on the use of the delimiter command.

**Note:** The syntax of the

CASE  
statement used inside stored programs differs slightly from that of the SQL CASE expression described in [CASE OPERATOR](#). The  
CASE  
statement cannot have an  
ELSE NULL  
clause, and it is terminated with  
END CASE

```
instead of  
END
```

## 1.1.6.4 DECLARE CONDITION

### Syntax

```
DECLARE condition_name CONDITION FOR condition_value  
  
condition_value:  
    SQLSTATE [VALUE] sqlstate_value  
    | mysql_error_code
```

### Description

The

```
DECLARE ... CONDITION
```

statement defines a named error condition. It specifies a condition that needs specific handling and associates a name with that condition. Later, the name can be used in a [DECLARE ... HANDLER](#), [SIGNAL](#) or [RESIGNAL](#) statement (as long as the statement is located in the same [BEGIN ... END](#) block).

Conditions must be declared after [local variables](#), but before [CURSORs](#) and [HANDLERS](#).

A condition\_value for

```
DECLARE ... CONDITION
```

can be an [SQLSTATE](#) value (a 5-character string literal) or a MySQL error code (a number). You should not use SQLSTATE value '00000' or MySQL error code 0, because those indicate success rather than an error condition. If you try, or if you specify an invalid SQLSTATE value, an error like this is produced:

```
ERROR 1407 (42000): Bad SQLSTATE: '00000'
```

For a list of SQLSTATE values and MariaDB error codes, see [MariaDB Error Codes](#).

## 1.1.6.5 DECLARE HANDLER

### Syntax

```
DECLARE handler_type HANDLER  
    FOR condition_value [, condition_value] ...  
    statement  
  
handler_type:  
    CONTINUE  
    | EXIT  
    | UNDO  
  
condition_value:  
    SQLSTATE [VALUE] sqlstate_value  
    | condition_name  
    | SQLWARNING  
    | NOT FOUND  
    | SQLEXCEPTION  
    | mariadb_error_code
```

### Description

The

```
DECLARE ... HANDLER
```

statement specifies handlers that each may deal with one or more conditions. If one of these conditions occurs, the specified statement is executed. statement can be a simple statement (for example,

```
SET var_name = value
```

), or it can be a compound statement written using [BEGIN](#) and [END](#).

Handlers must be declared after local variables, a

```
CONDITION
```

and a [CURSOR](#).

For a

CONTINUE

handler, execution of the current program continues after execution of the handler statement. For an EXIT handler, execution terminates for the BEGIN ... END compound statement in which the handler is declared. (This is true even if the condition occurs in an inner block.) The

UNDO

handler type statement is not supported.

If a condition occurs for which no handler has been declared, the default action is

EXIT

A condition\_value for

DECLARE ... HANDLER

can be any of the following values:

- An SQLSTATE value (a 5-character string literal) or a MariaDB error code (a number). You should not use SQLSTATE value '00000' or MariaDB error code 0, because those indicate success rather than an error condition. For a list of SQLSTATE values and MariaDB error codes, see [MariaDB Error Codes](#).
- A condition name previously specified with DECLARE ... CONDITION.
  - . It must be in the same stored program. See [DECLARE CONDITION](#).
- SQLWARNING is shorthand for the class of SQLSTATE values that begin with '01'.
- NOT FOUND is shorthand for the class of SQLSTATE values that begin with '02'. This is relevant only in the context of cursors and is used to control what happens when a cursor reaches the end of a data set. If no more rows are available, a No Data condition occurs with SQLSTATE value 02000. To detect this condition, you can set up a handler for it (or for a NOT FOUND condition). An example is shown in [Cursor Overview](#). This condition also occurs for SELECT ... INTO var\_list statements that retrieve no rows.
- SQLEXCEPTION is shorthand for the class of SQLSTATE values that do not begin with '00', '01', or '02'.

When an error raises, in some cases it could be handled by multiple

HANDLER

s. For example, there may be a handler for 1050 error, a separate handler for the 42S01 SQLSTATE, and another separate handler for the SQLEXCEPTION

class: in theory all occurrences of

HANDLER

may catch the 1050 error, but MariaDB chooses the

HANDLER

with the highest precedence. Here are the precedence rules:

- Handlers which refer to an error code have the highest precedence.
- Handlers which refer to a SQLSTATE come next.
- Handlers which refer to an error class have the lowest precedence.

In some cases, a statement could produce multiple errors. If this happens, in some cases multiple handlers could have the highest precedence. In such cases, the choice of the handler is indeterminate.

Note that if an error occurs within a

CONTINUE HANDLER

block, it can be handled by another

HANDLER

. However, a

HANDLER

which is already in the stack (that is, it has been called to handle an error and its execution didn't finish yet) cannot handle new errors — this prevents endless loops. For example, suppose that a stored procedure contains a

CONTINUE HANDLER

for

SQLWARNING

and another

CONTINUE HANDLER

for

NOT FOUND

. At some point, a NOT FOUND error occurs, and the execution enters the

NOT FOUND HANDLER

. But within that handler, a warning occurs, and the execution enters the

```

SQLWARNING HANDLER
. If another
NOT FOUND
error occurs, it cannot be handled again by the
NOT FOUND HANDLER
, because its execution is not finished.

```

When a

```

DECLARE HANDLER
block can handle more than one error condition, it may be useful to know which errors occurred. To do so, you can use the GET DIAGNOSTICS statement.

```

An error that is handled by a

```

DECLARE HANDLER
construct can be issued again using the RESIGNAL statement.

```

Below is an example using

```

DECLARE HANDLER
:
```

```

CREATE TABLE test.t (s1 INT, PRIMARY KEY (s1));

DELIMITER //

CREATE PROCEDURE handlerdemo ( )
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLSTATE '23000' SET @x2 = 1;
    SET @x = 1;
    INSERT INTO test.t VALUES (1);
    SET @x = 2;
    INSERT INTO test.t VALUES (1);
    SET @x = 3;
END;
// 

DELIMITER ;

CALL handlerdemo();

SELECT @x;
+-----+
| @x   |
+-----+
|   3  |
+-----+

```

## 1.1.6.6 DECLARE Variable

### Syntax

```
DECLARE var_name [, var_name] ... [[ROW] TYPE OF]] type [DEFAULT value]
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [TYPE OF / ROW TYPE OF](#)
3. [Examples](#)
4. [See Also](#)

### Description

This statement is used to declare local variables within [stored programs](#). To provide a default value for the variable, include a

```

DEFAULT
clause. The value can be specified as an expression (even subqueries are permitted); it need not be a constant. If the
DEFAULT
clause is missing, the initial value is
NULL

```

Local variables are treated like stored routine parameters with respect to data type and overflow checking. See [CREATE PROCEDURE](#).

Local variables must be declared before

CONDITION  
S, [CURSORS](#) and  
HANDLER  
S.

Local variable names are not case sensitive.

The scope of a local variable is within the

BEGIN ... END

block where it is declared. The variable can be referred to in blocks nested within the declaring block, except those blocks that declare a variable with the same name.

## TYPE OF / ROW TYPE OF

MariaDB starting with [10.3](#)

TYPE OF  
and  
ROW TYPE OF  
anchored data types for stored routines were introduced in [MariaDB 10.3](#).

Anchored data types allow a data type to be defined based on another object, such as a table row, rather than specifically set in the declaration. If the anchor object changes, so will the anchored data type. This can lead to routines being easier to maintain, so that if the data type in the table is changed, it will automatically be changed in the routine as well.

Variables declared with

ROW TYPE OF  
will have the same features as implicit [ROW](#) variables. It is not possible to use  
ROW TYPE OF  
variables in a [LIMIT](#) clause.

The real data type of

TYPE OF  
and  
ROW TYPE OF `table_name`

will become known at the very beginning of the stored routine call. [ALTER TABLE](#) or [DROP TABLE](#) statements performed inside the current routine on the tables that appear in anchors won't affect the data type of the anchored variables, even if the variable is declared after an [ALTER TABLE](#) or [DROP TABLE](#) statement.

The real data type of a

ROW TYPE OF `cursor_name`  
variable will become known when execution enters into the block where the variable is declared. Data type instantiation will happen only once. In a cursor  
ROW TYPE OF  
variable that is declared inside a loop, its data type will become known on the very first iteration and won't change on further loop iterations.

The tables referenced in

TYPE OF  
and  
ROW TYPE OF  
declarations will be checked for existence at the beginning of the stored routine call. [CREATE PROCEDURE](#) or [CREATE FUNCTION](#) will not check the referenced tables for existence.  
  
from MariaDB 10.3 :

```
DECLARE tmp TYPE OF t1.a; -- Get the data type from the column {{a}} in the table {{t1}}
DECLARE rec1 ROW TYPE OF t1; -- Get the row data type from the table {{t1}}
DECLARE rec2 ROW TYPE OF cur1; -- Get the row data type from the cursor {{cur1}}
```

## See Also

- [User-Defined variables](#)

## 1.1.6.7 FOR

MariaDB starting with [10.3](#)

FOR loops were introduced in [MariaDB 10.3](#).

### Syntax

Integer range FOR loop:

```
[begin_label:]  
FOR var_name IN [ REVERSE ] lower_bound .. upper_bound  
DO statement_list  
END FOR [ end_label ]
```

Explicit cursor FOR loop

```
[begin_label:]  
FOR record_name IN cursor_name [ ( cursor_actual_parameter_list )]  
DO statement_list  
END FOR [ end_label ]
```

Explicit cursor FOR loop ([Oracle mode](#))

```
[begin_label:]  
FOR record_name IN cursor_name [ ( cursor_actual_parameter_list )]  
LOOP  
    statement_list  
END LOOP [ end_label ]
```

Implicit cursor FOR loop

```
[begin_label:]  
FOR record_name IN ( select_statement )  
DO statement_list  
END FOR [ end_label ]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

FOR loops allow code to be executed a fixed number of times.

In an integer range FOR loop, MariaDB will compare the lower bound and upper bound values, and assign the lower bound value to a counter. If REVERSE is not specified, and the upper bound value is greater than or equal to the counter, the counter will be incremented and the statement will continue, after which the loop is entered again. If the upper bound value is greater than the counter, the loop will be exited.

If REVERSE is specified, the counter is decremented, and the upper bound value needs to be less than or equal for the loop to continue.

### Examples

Intger range FOR loop:

```

CREATE TABLE t1 (a INT);

DELIMITER //

FOR i IN 1..3
DO
    INSERT INTO t1 VALUES (i);
END FOR;
// 

DELIMITER ;

SELECT * FROM t1;
+---+
| a |
+---+
| 1 |
| 2 |
| 3 |
+---+

```

REVERSE integer range FOR loop:

```

CREATE OR REPLACE TABLE t1 (a INT);

DELIMITER //
FOR i IN REVERSE 4..12
DO
    INSERT INTO t1 VALUES (i);
END FOR;
// 
Query OK, 9 rows affected (0.422 sec)

DELIMITER ;

SELECT * FROM t1;
+---+
| a |
+---+
| 12 |
| 11 |
| 10 |
| 9 |
| 8 |
| 7 |
| 6 |
| 5 |
| 4 |
+---+

```

Explicit cursor in [Oracle mode](#) :

```

SET sql_mode=ORACLE;

CREATE OR REPLACE TABLE t1 (a INT, b VARCHAR(32));

INSERT INTO t1 VALUES (10, 'b0');
INSERT INTO t1 VALUES (11, 'b1');
INSERT INTO t1 VALUES (12, 'b2');

DELIMITER //

CREATE OR REPLACE PROCEDURE p1(pa INT) AS
  CURSOR cur(va INT) IS
    SELECT a, b FROM t1 WHERE a=va;
BEGIN
  FOR rec IN cur(pa)
  LOOP
    SELECT rec.a, rec.b;
  END LOOP;
END;
//

DELIMITER ;

CALL p1(10);
+-----+-----+
| rec.a | rec.b |
+-----+-----+
|     10 | b0      |
+-----+-----+

CALL p1(11);
+-----+-----+
| rec.a | rec.b |
+-----+-----+
|     11 | b1      |
+-----+-----+

CALL p1(12);
+-----+-----+
| rec.a | rec.b |
+-----+-----+
|     12 | b2      |
+-----+-----+

CALL p1(13);
Query OK, 0 rows affected (0.000 sec)

```

## See Also

- [LOOP](#)

## 1.1.6.8 GOTO

MariaDB starting with 10.3

The GOTO statement was introduced in [MariaDB 10.3](#) for Oracle compatibility.

## Syntax

```
GOTO label
```

### Contents

- [Syntax](#)
- [Description](#)
- [Example](#)

## Description

The

GOTO

statement causes the code to jump to the specified label, and continue operating from there. It is only accepted when in [Oracle mode](#) .

## Example

```
SET sql_mode=ORACLE;
DELIMITER //
CREATE OR REPLACE PROCEDURE p1 AS
BEGIN
  SELECT 1;
  GOTO label;
  SELECT 2;
  <<label>>
  SELECT 3;
END;
//
DELIMITER
call p1();
+---+
| 1 |
+---+
| 1 |
+---+
1 row in set (0.000 sec)

+---+
| 3 |
+---+
| 3 |
+---+
1 row in set (0.000 sec)
```

## 1.1.6.9 IF

### Syntax

```
IF search_condition THEN statement_list
  [ELSEIF search_condition THEN statement_list] ...
  [ELSE statement_list]
END IF;
```

### Description

The `IF` statement implements a basic conditional construct. If the `search_condition` evaluates to true, the corresponding SQL statement list is executed. If no `search_condition` matches, the statement list in the `ELSE` clause is executed. Each `statement_list` consists of one or more statements.

### See Also

- The [IF\(\) function](#) , which differs from the `IF` statement described above.
- [Changes in Oracle mode from MariaDB 10.3](#)

## 1.1.6.10 ITERATE

# Syntax

```
ITERATE label
```

ITERATE

can appear only within [LOOP](#), [REPEAT](#), and [WHILE](#) statements.

ITERATE

means "do the loop again", and uses the statement's [label](#) to determine which statements to repeat. The label must be in the same stored program, not in a caller procedure.

If you try to use

ITERATE

with a non-existing label, or if the label is associated to a construct which is not a loop, the following error will be produced:

```
ERROR 1308 (42000): ITERATE with no matching label: <label_name>
```

Below is an example of how

ITERATE

might be used:

```
CREATE PROCEDURE doiterate(p1 INT)
BEGIN
    label1: LOOP
        SET p1 = p1 + 1;
        IF p1 < 10 THEN ITERATE label1; END IF;
        LEAVE label1;
    END LOOP label1;
    SET @x = p1;
END
```

## See Also

- [LEAVE](#) - Exits a loop (or any labeled code block)

## 1.1.6.11 Labels

### Syntax

```
label: <construct>
[label]
```

Labels are MariaDB [identifiers](#) which can be used to identify a [BEGIN ... END](#) construct or a loop. They have a maximum length of 16 characters and can be quoted with backticks (i.e.,

).

Labels have a start part and an end part. The start part must precede the portion of code it refers to, must be followed by a colon (

:

) and can be on the same or different line. The end part is optional and adds nothing, but can make the code more readable. If used, the end part must precede the construct's delimiter (

;

). Constructs identified by a label can be nested. Each construct can be identified by only one label.

Labels need not be unique in the stored program they belong to. However, a label for an inner loop cannot be identical to a label for an outer loop. In this case, the following error would be produced:

```
ERROR 1309 (42000): Redefining label <label_name>
```

[LEAVE](#) and [ITERATE](#) statements can be used to exit or repeat a portion of code identified by a label. They must be in the same [Stored Routine](#), [Trigger](#) or [Event](#) which contains the target label.

Below is an example using a simple label that is used to exit a [LOOP](#):

```

CREATE PROCEDURE `test_sp`()
BEGIN
    `my_label`:
    LOOP
        SELECT 'looping';
        LEAVE `my_label`;
    END LOOP;
    SELECT 'out of loop';
END;

```

The following label is used to exit a procedure, and has an end part:

```

CREATE PROCEDURE `test_sp`()
`my_label`:
BEGIN
    IF @var = 1 THEN
        LEAVE `my_label`;
    END IF;
    DO something();
END `my_label`;

```

## 1.1.6.12 LEAVE

### Syntax

```
LEAVE label
```

This statement is used to exit the flow control construct that has the given `label`. The label must be in the same stored program, not in a caller procedure

`LEAVE`

can be used within `BEGIN ... END` or loop constructs (`LOOP`, `REPEAT`, `WHILE`). In [Stored Procedures](#), [Triggers](#) and [Events](#), `LEAVE` can refer to the outmost `BEGIN ... END` construct; in that case, the program exits the procedure. In [Stored Functions](#), `RETURN` can be used instead.

Note that `LEAVE` cannot be used to exit a [DECLARE HANDLER](#) block.

If you try to `LEAVE` a non-existing label, or if you try to `LEAVE` a `HANDLER` block, the following error will be produced:

```
ERROR 1308 (42000): LEAVE with no matching label: <label_name>
```

The following example uses

`LEAVE`

to exit the procedure if a condition is true:

```

CREATE PROCEDURE proc(IN p TINYINT)
CONTAINS SQL
`whole_proc`:
BEGIN
    SELECT 1;
    IF p < 1 THEN
        LEAVE `whole_proc`;
    END IF;
    SELECT 2;
END;

CALL proc(0);
+---+
| 1 |
+---+
| 1 |
+---+

```

### See Also

- [ITERATE](#) - Repeats a loop execution

## 1.1.6.13 LOOP

### Syntax

```
[begin_label:] LOOP  
    statement_list  
END LOOP [end_label]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

## Description

LOOP

implements a simple loop construct, enabling repeated execution of the statement list, which consists of one or more statements, each terminated by a semicolon (i.e.,

;

) statement delimiter. The statements within the loop are repeated until the loop is exited; usually this is accomplished with a [LEAVE](#) statement.

A

LOOP

statement can be [labeled](#).

end\_label

cannot be given unless

begin\_label

also is present. If both are present, they must be the same.

See [Delimiters](#) in the [mysql](#) client for more on delimiter usage in the client.

## See Also

- [LOOP in Oracle mode](#)
- [ITERATE](#)
- [LEAVE](#)
- [FOR Loops](#)

## 1.1.6.14 REPEAT LOOP

### Syntax

```
[begin_label:] REPEAT  
    statement_list  
UNTIL search_condition  
END REPEAT [end_label]
```

The statement list within a

REPEAT

statement is repeated until the search\_condition is true. Thus, a

REPEAT

always enters the loop at least once. statement\_list consists of one or more statements, each terminated by a semicolon (i.e.,

;

) statement delimiter.

A

REPEAT

statement can be [labeled](#). end\_label cannot be given unless begin\_label also is present. If both are present, they must be the same.

See [Delimiters](#) in the [mysql](#) client for more on client delimiter usage.

```

DELIMITER //

CREATE PROCEDURE dorepeat(p1 INT)
BEGIN
    SET @x = 0;
    REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
END
// 

CALL dorepeat(1000)//

SELECT @x//
+-----+
| @x   |
+-----+
| 1001 |
+-----+

```

## 1.1.6.15 RESIGNAL

### Syntax

```

RESIGNAL [error_condition]
    [SET error_property
    [, error_property] ...]

error_condition:
    SQLSTATE [VALUE] 'sqlstate_value'
    | condition_name

error_property:
    error_property_name = <error_property_value>

error_property_name:
    CLASS_ORIGIN
    | SUBCLASS_ORIGIN
    | MESSAGE_TEXT
    | MYSQL_ERRNO
    | CONSTRAINT_CATALOG
    | CONSTRAINT_SCHEMA
    | CONSTRAINT_NAME
    | CATALOG_NAME
    | SCHEMA_NAME
    | TABLE_NAME
    | COLUMN_NAME
    | CURSOR_NAME

```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

## Description

The syntax of

```

RESIGNAL
and its semantics are very similar to SIGNAL . This statement can only be used within an error HANDLER . It produces an error, like SIGNAL .
RESIGNAL
clauses are the same as SIGNAL, except that they all are optional, even SQLSTATE . All the properties which are not specified in
RESIGNAL
, will be identical to the properties of the error that was received by the error HANDLER . For a description of the clauses, see diagnostics area .

```

Note that

```

RESIGNAL
does not empty the diagnostics area: it just appends another error condition.

```

```

RESIGNAL
, without any clauses, produces an error which is identical to the error that was received by HANDLER .

```

If used out of a HANDLER construct, RESIGNAL produces the following error:

```
ERROR 1645 (0K000): RESIGNAL when handler not active
```

In MariaDB 5.5 , if a **HANDLER** contained a **CALL** to another procedure, that procedure could use **RESIGNAL** . Since MariaDB 10.0 , trying to do this raises the above error.

For a list of

SQLSTATE  
values and MariaDB error codes, see [MariaDB Error Codes](#) .

The following procedure tries to query two tables which don't exist, producing a 1146 error in both cases. Those errors will trigger the **HANDLER** . The first time the error will be ignored and the client will not receive it, but the second time, the error is re-signaled, so the client will receive it.

```
CREATE PROCEDURE test_error( )  
BEGIN  
    DECLARE CONTINUE HANDLER  
        FOR 1146  
    BEGIN  
        IF @hide_errors IS FALSE THEN  
            RESIGNAL;  
        END IF;  
    END;  
    SET @hide_errors = TRUE;  
    SELECT 'Next error will be ignored' AS msg;  
    SELECT `c` FROM `temptab_one`;  
    SELECT 'Next error won''t be ignored' AS msg;  
    SET @hide_errors = FALSE;  
    SELECT `c` FROM `temptab_two`;  
END;  
  
CALL test_error();  
  
+-----+  
| msg |  
+-----+  
| Next error will be ignored |  
+-----+  
  
+-----+  
| msg |  
+-----+  
| Next error won't be ignored |  
+-----+  
  
ERROR 1146 (42S02): Table 'test temptab_two' doesn't exist
```

The following procedure re-signals an error, modifying only the error message to clarify the cause of the problem.

```
CREATE PROCEDURE test_error()  
BEGIN  
    DECLARE CONTINUE HANDLER  
        FOR 1146  
    BEGIN  
        RESIGNAL SET  
        MESSAGE_TEXT = ``temptab` does not exist';  
    END;  
    SELECT `c` FROM `temptab`;  
END;  
  
CALL test_error();  
ERROR 1146 (42S02): `temptab` does not exist
```

As explained above, this works on [MariaDB 5.5](#) , but produces a 1645 error since 10.0.

```
CREATE PROCEDURE handle_error()  
BEGIN  
    RESIGNAL;  
END;  
CREATE PROCEDURE p()  
BEGIN  
    DECLARE EXIT HANDLER FOR SQLEXCEPTION CALL p();  
    SIGNAL SQLSTATE '45000';  
END;
```

## See Also

- [Diagnostics Area](#)
- [SIGNAL](#)
- [HANDLER](#)
- [Stored Routines](#)
- [MariaDB Error Codes](#)

## 1.1.6.16 RETURN

### Syntax

```
RETURN expr
```

The

```
RETURN  
statement terminates execution of a stored function and returns the value  
expr  
to the function caller. There must be at least one  
RETURN  
statement in a stored function. If the function has multiple exit points, all exit points must have a  
RETURN
```

This statement is not used in [stored procedures](#), [triggers](#), or [events](#). [LEAVE](#) can be used instead.

The following example shows that

```
RETURN  
can return the result of a scalar subquery:
```

```
CREATE FUNCTION users_count() RETURNS BOOL  
READS SQL DATA  
BEGIN  
    RETURN (SELECT COUNT(DISTINCT User) FROM mysql.user);  
END;
```

## 1.1.6.17 SELECT INTO

### Syntax

```
SELECT col_name [, col_name] ...  
    INTO var_name [, var_name] ...  
    table_expr
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

`SELECT ... INTO` enables selected columns to be stored directly into variables. No resultset is produced. The query should return a single row. If the query returns no rows, a warning with error code 1329 occurs (No data), and the variable values remain unchanged. If the query returns multiple rows, error 1172 occurs (Result consisted of more than one row). If it is possible that the statement may retrieve multiple rows, you can use

```
LIMIT 1  
to limit the result set to a single row.
```

The `INTO` clause can also be specified at the end of the statement.

In the context of such statements that occur as part of events executed by the Event Scheduler, diagnostics messages (not only errors, but also warnings) are written to the error log, and, on Windows, to the application event log.

This statement can be used with both [local variables](#) and [user-defined variables](#).

For the complete syntax, see [SELECT](#).

Another way to set a variable's value is the [SET](#) statement.

```
SELECT ... INTO
results are not stored in the query cache even if
SQL_CACHE
is specified.
```

## Examples

```
SELECT id, data INTO @x,@y
FROM test.t1 LIMIT 1;
```

## See Also

- [SELECT](#) - full SELECT syntax.
- [SELECT INTO OUTFILE](#) - formatting and writing the result to an external file.
- [SELECT INTO DUMPFILE](#) - binary-safe writing of the unformatted results to an external file.

## 1.1.6.18 SET Variable

## 1.1.6.19 SIGNAL

### Syntax

```
SIGNAL error_condition
  [SET error_property
  [, error_property] ...]

error_condition:
  SQLSTATE [VALUE] 'sqlstate_value'
  | condition_name

error_property:
  error_property_name = <error_property_value>

error_property_name:
  CLASS_ORIGIN
  | SUBCLASS_ORIGIN
  | MESSAGE_TEXT
  | MYSQL_ERRNO
  | CONSTRAINT_CATALOG
  | CONSTRAINT_SCHEMA
  | CONSTRAINT_NAME
  | CATALOG_NAME
  | SCHEMA_NAME
  | TABLE_NAME
  | COLUMN_NAME
  | CURSOR_NAME
```

## Contents

1. [Syntax](#)
2. [Errors](#)
3. [Examples](#)
4. [See Also](#)

#### SIGNAL

empties the [diagnostics area](#) and produces a custom error. This statement can be used anywhere, but is generally useful when used inside a [stored program](#). When the error is produced, it can be caught by a [HANDLER](#). If not, the current stored program, or the current statement, will terminate with the specified error.

Sometimes an error [HANDLER](#) just needs to [SIGNAL](#) the same error it received, optionally with some changes. Usually the [RESIGNAL](#) statement is the most convenient way to do this.

#### error\_condition

can be an [SQLSTATE](#) value or a named error condition defined via [DECLARE CONDITION](#). [SQLSTATE](#) must be a constant string consisting of five characters. These codes are standard to ODBC and ANSI SQL. For customized errors, the recommended [SQLSTATE](#) is '45000'. For a list of

SQLSTATE values used by MariaDB, see the [MariaDB Error Codes](#) page. The SQLSTATE can be read via the API method

```
mysql_sqlstate( )
```

To specify error properties user-defined variables and [local variables](#) can be used, as well as [character set conversions](#) (but you can't set a collation).

The error properties, their type and their default values are explained in the [diagnostics area](#) page.

## Errors

If the SQLSTATE is not valid, the following error like this will be produced:

```
ERROR 1407 (42000): Bad SQLSTATE: '123456'
```

If a property is specified more than once, an error like this will be produced:

```
ERROR 1641 (42000): Duplicate condition information item 'MESSAGE_TEXT'
```

If you specify a condition name which is not declared, an error like this will be produced:

```
ERROR 1319 (42000): Undefined CONDITION: cond_name
```

If MYSQL\_ERRNO is out of range, you will get an error like this:

```
ERROR 1231 (42000): Variable 'MYSQL_ERRNO' can't be set to the value of '0'
```

## Examples

Here's what happens if [SIGNAL](#) is used in the client to generate errors:

```
SIGNAL SQLSTATE '01000';
Query OK, 0 rows affected, 1 warning (0.00 sec)

SHOW WARNINGS;

+-----+-----+
| Level | Code  | Message          |
+-----+-----+
| Warning | 1642 | Unhandled user-defined warning condition |
+-----+-----+
1 row in set (0.06 sec)

SIGNAL SQLSTATE '02000';
ERROR 1643 (02000): Unhandled user-defined not found condition
```

How to specify MYSQL\_ERRNO and MESSAGE\_TEXT properties:

```
SIGNAL SQLSTATE '45000' SET MYSQL_ERRNO=30001, MESSAGE_TEXT='Hello, world!';

ERROR 30001 (45000): Hello, world!
```

The following code shows how to use user variables, local variables and character set conversion with SIGNAL:

```
CREATE PROCEDURE test_error(x INT)
BEGIN
    DECLARE errno SMALLINT UNSIGNED DEFAULT 31001;
    SET @errormsg = 'Hello, world!';
    IF x = 1 THEN
        SIGNAL SQLSTATE '45000' SET
        MYSQL_ERRNO = errno,
        MESSAGE_TEXT = @errormsg;
    ELSE
        SIGNAL SQLSTATE '45000' SET
        MYSQL_ERRNO = errno,
        MESSAGE_TEXT = _utf8'Hello, world!';
    END IF;
END;
```

How to use named error conditions:

```

CREATE PROCEDURE test_error(n INT)
BEGIN
    DECLARE `too_big` CONDITION FOR SQLSTATE '45000';
    IF n > 10 THEN
        SIGNAL `too_big`;
    END IF;
END;

```

In this example, we'll define a [HANDLER](#) for an error code. When the error occurs, we [SIGNAL](#) a more informative error which makes sense for our procedure:

```

CREATE PROCEDURE test_error()
BEGIN
    DECLARE EXIT HANDLER
    FOR 1146
    BEGIN
        SIGNAL SQLSTATE '45000' SET
        MESSAGE_TEXT = 'Temporary tables not found; did you call init() procedure?';
    END;
    -- this will produce a 1146 error
    SELECT `c` FROM `temptab`;
END;

```

## See Also

- [Diagnostics Area](#)
- [RESIGNAL](#)
- [HANDLER](#)
- [Stored Routines](#)
- [MariaDB Error Codes](#)

## 1.1.6.20 WHILE

### Syntax

```
[begin_label:] WHILE search_condition DO
    statement_list
END WHILE [end_label]
```

### Description

The statement list within a

```
WHILE
    statement is repeated as long as the
    search_condition
    is true. statement_list consists of one or more statements. If the loop must be executed at least once,
```

```
REPEAT ... LOOP
```

can be used instead.

A

```
WHILE
    statement can be labeled. end_label cannot be given unless begin_label also is present. If both are present, they must be the same.
```

### Examples

```

CREATE PROCEDURE dowhile()
BEGIN
    DECLARE v1 INT DEFAULT 5;

    WHILE v1 > 0 DO
        ...
        SET v1 = v1 - 1;
    END WHILE;
END

```

## 1.1.7 Stored Routine Statements

### 1.1.7.1 CALL

#### Syntax

```
CALL sp_name([parameter[,...]])  
CALL sp_name[()]
```

#### Description

The

CALL  
statement invokes a [stored procedure](#) that was defined previously with [CREATE PROCEDURE](#).

Stored procedure names can be specified as

database\_name.procedure\_name  
. Procedure names and database names can be quoted with backticks (). This is necessary if they are reserved words, or contain special characters. See [identifier qualifiers](#) for details.

CALL p()  
and  
CALL p  
are equivalent.

If parentheses are used, any number of spaces, tab characters and newline characters are allowed between the procedure's name and the open parenthesis.

CALL  
can pass back values to its caller using parameters that are declared as  
OUT  
or  
INOUT  
parameters. If no value is assigned to an  
OUT  
parameter,  
NULL  
is assigned (and its former value is lost). To pass such values from another stored program you can use [user-defined variables](#), [local variables](#) or routine's parameters; in other contexts, you can only use user-defined variables.

CALL  
can also be executed as a prepared statement. Placeholders can be used for  
IN  
parameters in all versions of MariaDB; for  
OUT  
and  
INOUT  
parameters, placeholders can be used since [MariaDB 5.5](#).

When the procedure returns, a client program can also obtain the number of rows affected for the final statement executed within the routine: At the SQL level, call the

`ROW_COUNT()`  
function; from the C API, call the  
`mysql_affected_rows()`  
function.

If the

`CLIENT_MULTI_RESULTS`  
API flag is set,  
CALL  
can return any number of resultsets and the called stored procedure can execute prepared statements. If it is not set, at most one resultset can be returned and prepared statements cannot be used within procedures.

### 1.1.7.2 DO

# Syntax

```
DO expr [, expr] ...
```

## Description

DO  
executes the expressions but does not return any results. In most respects,  
DO  
is shorthand for  
SELECT expr, ...,  
but has the advantage that it is slightly faster when you do not care about the result.

DO  
is useful primarily with functions that have side effects, such as

```
RELEASE_LOCK()
```

## 1.1.8 Table Statements

### 1.1.9 Transactions

#### 1.1.9.1 START TRANSACTION

## Syntax

```
START TRANSACTION [transaction_property [, transaction_property] ...] | BEGIN [WORK]
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
SET autocommit = {0 | 1}

transaction_property:
  WITH CONSISTENT SNAPSHOT
  | READ WRITE
  | READ ONLY
```

## Contents

- 1. [Syntax](#)
- 2. [Description](#)
  - 1. [Access Mode](#)
  - 2. [autocommit](#)
  - 3. [DDL Statements](#)
  - 4. [in\\_transaction](#)
  - 5. [WITH CONSISTENT SNAPSHOT](#)
- 3. [Examples](#)
- 4. [See Also](#)

## Description

The

```
START TRANSACTION
or
BEGIN
```

statement begins a new transaction. [COMMIT](#) commits the current transaction, making its changes permanent. [ROLLBACK](#) rolls back the current transaction, canceling its changes. The [SET autocommit](#) statement disables or enables the default autocommit mode for the current session.

START TRANSACTION and SET autocommit = 1 implicitly commit the current transaction, if any.

The optional

```
WORK
keyword is supported for
COMMIT
```

and  
ROLLBACK  
, as are the  
CHAIN  
and  
RELEASE  
clauses.  
CHAIN  
and  
RELEASE  
can be used for additional control over transaction completion. The value of the [completion\\_type](#) system variable determines the default completion behavior.

The  
AND CHAIN  
clause causes a new transaction to begin as soon as the current one ends, and the new transaction has the same isolation level as the just-terminated transaction. The  
RELEASE  
clause causes the server to disconnect the current client session after terminating the current transaction. Including the  
NO  
keyword suppresses  
CHAIN  
or  
RELEASE  
completion, which can be useful if the [completion\\_type](#) system variable is set to cause chaining or release completion by default.

## Access Mode

The access mode specifies whether the transaction is allowed to write data or not. By default, transactions are in  
READ WRITE  
mode (see the [tx\\_read\\_only](#) system variable).  
READ ONLY  
mode allows the storage engine to apply optimizations that cannot be used for transactions which write data. The only exception to this rule is that read only transactions can perform DDL statements on temporary tables.

It is not permitted to specify both  
READ WRITE  
and  
READ ONLY  
in the same statement.

READ WRITE  
and  
READ ONLY  
can also be specified in the

[SET TRANSACTION](#)

statement, in which case the specified mode is valid for all sessions, or for all subsequent transaction used by the current session.

## autocommit

By default, MariaDB runs with [autocommit](#) mode enabled. This means that as soon as you execute a statement that updates (modifies) a table, MariaDB stores the update on disk to make it permanent. To disable autocommit mode, use the following statement:

```
SET autocommit=0;
```

After disabling autocommit mode by setting the autocommit variable to zero, changes to transaction-safe tables (such as those for InnoDB or NDBCLUSTER ) are not made permanent immediately. You must use COMMIT to store your changes to disk or ROLLBACK to ignore the changes.

To disable autocommit mode for a single series of statements, use the

```
START TRANSACTION  
statement.
```

## DDL Statements

DDL statements (

```

CREATE
,
ALTER
,
DROP
) and administrative statements (
FLUSH
,
RESET
,
OPTIMIZE
,
ANALYZE
,
CHECK
,
REPAIR
,
CACHE INDEX
), transaction management statements (
BEGIN
,
START TRANSACTION
) and
LOAD DATA INFILE
, cause an implicit
COMMIT
and start a new transaction. An exception to this rule are the DDL that operate on temporary tables: you can
CREATE
,
ALTER
and
DROP
them without causing any
COMMIT
, but those actions cannot be rolled back. This means that if you call
ROLLBACK
, the temporary tables you created in the transaction will remain, while the rest of the transaction will be rolled back.

```

Transactions cannot be used in Stored Functions or Triggers. In Stored Procedures and Events BEGIN is not allowed, so you should use START TRANSACTION instead.

A transaction acquires a [metadata lock](#) on every table it accesses to prevent other connections from altering their structure. The lock is released at the end of the transaction. This happens even with non-transactional storage engines (like [MEMORY](#) or [CONNECT](#) ), so it makes sense to use transactions with non-transactional tables.

## in\_transaction

The [in\\_transaction](#) system variable is a session-only, read-only variable that returns

```

1
inside a transaction, and
0
if not in a transaction.

```

## WITH CONSISTENT SNAPSHOT

The

```
WITH CONSISTENT SNAPSHOT
```

option starts a consistent read for storage engines such as [InnoDB](#) that can do so, the same as if a START TRANSACTION followed by a SELECT from any InnoDB table was issued.

See [Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT](#).

## Examples

```

START TRANSACTION;
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summary=@A WHERE type=1;
COMMIT;

```

## See Also

- [Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT](#)
- [MyRocks and START TRANSACTION WITH CONSISTENT SNAPSHOT](#)

## 1.1.9.2 COMMIT

The

COMMIT

statement ends a transaction, saving any changes to the data so that they become visible to subsequent transactions. Also, [unlocks metadata](#) changed by current transaction. If [autocommit](#) is set to 1, an implicit commit is performed after each statement. Otherwise, all transactions which don't end with an explicit

COMMIT

are implicitly rolled back and the changes are lost. The

ROLLBACK

statement can be used to do this explicitly.

The required syntax for the

COMMIT

statement is as follows:

```
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
```

COMMIT

is the more important transaction terminator, as well as the more interesting one. The basic form of the

COMMIT

statement is simply the keyword

COMMIT

(the keyword

WORK

is simply noise and can be omitted without changing the effect).

The optional

AND CHAIN

clause is a convenience for initiating a new transaction as soon as the old transaction terminates. If

AND CHAIN

is specified, then there is effectively nothing between the old and new transactions, although they remain separate. The characteristics of the new transaction will be the same as the characteristics of the old one — that is, the new transaction will have the same access mode, isolation level and diagnostics area size (we'll discuss all of these shortly) as the transaction just terminated.

RELEASE

tells the server to disconnect the client immediately after the current transaction.

There are

NO RELEASE

and

AND NO CHAIN

options. By default, commits do not

RELEASE

or

CHAIN

, but it's possible to change this default behavior with the [completion\\_type](#) server system variable. In this case, the

AND NO CHAIN

and

NO RELEASE

options override the server default.

## See Also

- [autocommit](#) - server system variable that determines whether statements are automatically committed.
- [completion\\_type](#) - server system variable that determines whether COMMIT's are standard, COMMIT AND CHAIN or COMMIT RELEASE.
- [SQL statements that cause an implicit commit](#)

## 1.1.9.3 ROLLBACK

The

ROLLBACK

statement rolls back (ends) a transaction, destroying any changes to SQL-data so that they never become visible to subsequent transactions.

The required syntax for the

ROLLBACK

statement is as follows.

```
ROLLBACK [ WORK ] [ AND [ NO ] CHAIN ]
[ TO [ SAVEPOINT ] {<savepoint name> | <simple target specification>} ]
```

The

ROLLBACK

statement will either end a transaction, destroying all data changes that happened during any of the transaction, or it will just destroy any data changes that happened since you established a savepoint. The basic form of the

ROLLBACK

statement is just the keyword

ROLLBACK

(the keyword

WORK

is simply noise and can be omitted without changing the effect).

The optional

AND CHAIN

clause is a convenience for initiating a new transaction as soon as the old transaction terminates. If

AND CHAIN

is specified, then there is effectively nothing between the old and new transactions, although they remain separate. The characteristics of the new transaction will be the same as the characteristics of the old one — that is, the new transaction will have the same access mode, isolation level and diagnostics area size (we'll discuss all of these shortly) as the transaction just terminated. The

AND NO CHAIN

option just tells your DBMS to end the transaction — that is, these four SQL statements are equivalent:

```
ROLLBACK;
ROLLBACK WORK;
ROLLBACK AND NO CHAIN;
ROLLBACK WORK AND NO CHAIN;
```

All of them end a transaction without saving any transaction characteristics. The only other options, the equivalent statements:

```
ROLLBACK AND CHAIN;
ROLLBACK WORK AND CHAIN;
```

both tell your DBMS to end a transaction, but to save that transaction's characteristics for the next transaction.

ROLLBACK

is much simpler than

COMMIT

: it may involve no more than a few deletions (of Cursors, locks, prepared SQL statements and log-file entries). It's usually assumed that

ROLLBACK

can't fail, although such a thing is conceivable (for example, an encompassing transaction might reject an attempt to

ROLLBACK

because it's lining up for a

COMMIT

).

ROLLBACK

cancels all effects of a transaction. It does not cancel effects on objects outside the DBMS's control (for example the values in host program variables or the settings made by some SQL/CLI function calls). But in general, it is a convenient statement for those situations when you say "oops, this isn't working" or when you simply don't care whether your temporary work becomes permanent or not.

Here is a moot question. If all you've been doing is

SELECT

s, so that there have been no data changes, should you end the transaction with

ROLLBACK

or

COMMIT

? It shouldn't really matter because both

ROLLBACK

and

COMMIT

do the same transaction-terminating job. However, the popular conception is that

ROLLBACK

implies failure, so after a successful series of

```

SELECT
statements the convention is to end the transaction with
COMMIT
rather than
ROLLBACK

```

MariaDB (and most other DBMSs) supports rollback of SQL-data change statements, but not of SQL-Schema statements. This means that if you use any of

```

CREATE
,
ALTER
,
DROP
,
GRANT
,
REVOKE
, you are implicitly committing at execution time.

```

```

INSERT INTO Table_2 VALUES(5);
DROP TABLE Table_3 CASCADE;
ROLLBACK;

```

The result will be that both the

```

INSERT
and the
DROP
will go through as separate transactions so the
ROLLBACK
will have no effect.

```

## 1.1.9.4 SET TRANSACTION

## 1.1.9.5 LOCK TABLES

### Syntax

```

LOCK TABLE[S]
tbl_name [[AS] alias] lock_type
[, tbl_name [[AS] alias] lock_type] ...
[WAIT n|NOWAIT]

```

```

lock_type:
  READ [LOCAL]
  | [LOW_PRIORITY] WRITE
  | WRITE CONCURRENT

```

```
UNLOCK TABLES
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [WAIT/NOWAIT](#)
3. [Limitations](#)
4. [See Also](#)

### Description

The *lock\_type* can be one of:

Option	Description
READ	Read lock, no writes allowed
READ LOCAL	Read lock, but allow <a href="#">concurrent inserts</a>
WRITE	Exclusive write lock. No other connections can read or write to this table

LOW_PRIORITY WRITE	Exclusive write lock, but allow new read locks on the table until we get the write lock.
WRITE CONCURRENT	Exclusive write lock, but allow READ LOCAL locks to the table.

MariaDB enables client sessions to acquire table locks explicitly for the purpose of cooperating with other sessions for access to tables, or to prevent other sessions from modifying tables during periods when a session requires exclusive access to them. A session can acquire or release locks only for itself. One session cannot acquire locks for another session or release locks held by another session.

Locks may be used to emulate transactions or to get more speed when updating tables.

```
LOCK TABLES
explicitly acquires table locks for the current client session. Table locks can be acquired for base tables or views. To use
LOCK TABLES
, you must have the
LOCK TABLES
privilege, and the
SELECT
privilege for each object to be locked. See
```

[GRANT](#)

For view locking,

```
LOCK TABLES
adds all base tables used in the view to the set of tables to be locked and locks them automatically. If you lock a table explicitly with
LOCK TABLES
, any tables used in triggers are also locked implicitly, as described in Triggers and Implicit Locks.
```

[UNLOCK TABLES](#) explicitly releases any table locks held by the current session.

MariaDB starting with [10.3.0](#)

## WAIT/NOWAIT

Set the lock wait timeout. See [WAIT and NOWAIT](#).

## Limitations

- LOCK TABLES  
[doesn't work when using Galera cluster](#). You may experience crashes or locks when used with Galera.
- LOCK TABLES  
works on XtraDB/InnoDB tables only if the [innodb\\_table\\_locks](#) system variable is set to 1 (the default) and [autocommit](#) is set to 0 (1 is default). Please note that no error message will be returned on LOCK TABLES with innodb\_table\_locks = 0.
- LOCK TABLES  
[implicitly commits](#) the active transaction, if any. Also, starting a transaction always releases all table locks acquired with LOCK TABLES. This means that there is no way to have table locks and an active transaction at the same time. The only exceptions are the transactions in [autocommit](#) mode. To preserve the data integrity between transactional and non-transactional tables, the [GET\\_LOCK\(\)](#) function can be used.
- When using  
LOCK TABLES  
on a  
TEMPORARY  
table, it will always be locked with a  
WRITE  
lock.
- While a connection holds an explicit read lock on a table, it cannot modify it. If you try, the following error will be produced:

ERROR 1099 (HY000): Table 'tab\_name' was locked with a READ lock and can't be updated

- While a connection holds an explicit lock on a table, it cannot access a non-locked table. If you try, the following error will be produced:

ERROR 1100 (HY000): Table 'tab\_name' was not locked with LOCK TABLES

- While a connection holds an explicit lock on a table, it cannot issue the following: INSERT DELAYED, CREATE TABLE, CREATE TABLE ... LIKE, and DDL statements involving stored programs and views (except for triggers). If you try, the following error will be produced:

ERROR 1192 (HY000): Can't execute the given command because you have active locked tables or an active transaction

- **LOCK TABLES**

can not be used in stored routines - if you try, the following error will be produced on creation:

ERROR 1314 (0A000): LOCK is not allowed in stored procedures

## See Also

- [UNLOCK TABLES](#)

## 1.1.9.6 SAVEPOINT

### Syntax

```
SAVEPOINT identifier  
ROLLBACK [WORK] TO [SAVEPOINT] identifier  
RELEASE SAVEPOINT identifier
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Errors](#)

### Description

InnoDB supports the SQL statements

```
SAVEPOINT  
,  
ROLLBACK TO SAVEPOINT  
,  
RELEASE SAVEPOINT  
and the optional  
WORK  
keyword for  
ROLLBACK
```

Each savepoint must have a legal [MariaDB identifier](#). A savepoint is a named sub-transaction.

Normally **ROLLBACK** undoes the changes performed by the whole transaction. When used with the TO clause, it undoes the changes performed after the specified savepoint, and erases all subsequent savepoints. However, all locks that have been acquired after the save point will survive. **RELEASE SAVEPOINT** does not rollback or commit any changes, but removes the specified savepoint.

When the execution of a trigger or a stored function begins, it is not possible to use statements which reference a savepoint which was defined from out of that stored program.

When a **COMMIT** (including implicit commits) or a **ROLLBACK** statement (with no TO clause) is performed, they act on the whole transaction, and all savepoints are removed.

### Errors

If COMMIT or ROLLBACK is issued and no transaction was started, no error is reported.

If SAVEPOINT is issued and no transaction was started, no error is reported but no savepoint is created. When ROLLBACK TO SAVEPOINT or RELEASE SAVEPOINT is called for a savepoint that does not exist, an error like this is issued:

ERROR 1305 (42000): SAVEPOINT *svp\_name* does not exist

## 1.1.9.7 Metadata Locking

MariaDB supports metadata locking. This means that when a transaction (including [XA transactions](#)) uses a table, it locks its metadata until the end of transaction. Non-transactional tables are also locked, as well as views and objects which are related to locked tables/views (stored functions, triggers, etc). When a connection tries to use a DDL statement (like an [ALTER TABLE](#)) which modifies a table that is locked, that connection is queued, and has to wait until it's unlocked. Using savepoints and performing a partial rollback does not release metadata locks.

[LOCK TABLES ... WRITE](#) are also queued. Some wrong statements which produce an error may not need to wait for the lock to be freed.

The metadata lock's timeout is determined by the value of the `lock_wait_timeout` server system variable (in seconds). However, note that its default value is 31536000 (1 year, MariaDB <= 10.2.3), or 86400 (1 day, MariaDB >= 10.2.4). If this timeout is exceeded, the following error is returned:

```
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
```

If the `metadata_lock_info` plugin is installed, the `Information Schema metadata_lock_info` table stores information about existing metadata locks.

MariaDB starting with 10.5.2

From MariaDB 10.5, the `Performance Schema metadata_locks` table contains metadata lock information.

## Example

Let's use the following MEMORY (non-transactional) table:

```
CREATE TABLE t (a INT) ENGINE = MEMORY;
```

Connection 1 starts a transaction, and INSERTs a row into t:

```
START TRANSACTION;  
  
INSERT INTO t SET a=1;
```

t  
's metadata is now locked by connection 1. Connection 2 tries to alter  
t  
, but has to wait:

```
ALTER TABLE t ADD COLUMN b INT;
```

Connection 2's prompt is blocked now.

Now connection 1 ends the transaction:

```
COMMIT;
```

...and connection 2 finally gets the output of its command:

```
Query OK, 1 row affected (35.23 sec)  
Records: 1 Duplicates: 0 Warnings: 0
```

## 1.1.9.8 SQL statements That Cause an Implicit Commit

Some SQL statements cause an implicit commit. As a rule of thumb, such statements are DDL statements. The same statements (except for `SHUTDOWN`) produce a 1400 error (`SQLSTATE 'XAE09'`) if a XA transaction is in effect.

Here is the list:

```
ALTER DATABASE ... UPGRADE DATA DIRECTORY NAME
ALTER EVENT
ALTER FUNCTION
ALTER PROCEDURE
ALTER SERVER
ALTER TABLE
ALTER VIEW
ANALYZE TABLE
BEGIN
CACHE INDEX
CHANGE MASTER TO
CHECK TABLE
CREATE DATABASE
CREATE EVENT
CREATE FUNCTION
CREATE INDEX
CREATE PROCEDURE
CREATE ROLE
CREATE SERVER
CREATE TABLE
CREATE TRIGGER
CREATE USER
CREATE VIEW
DROP DATABASE
DROP EVENT
DROP FUNCTION
DROP INDEX
DROP PROCEDURE
DROP ROLE
DROP SERVER
DROP TABLE
DROP TRIGGER
DROP USER
DROP VIEW
FLUSH
GRANT
LOAD INDEX INTO CACHE
LOCK TABLES
OPTIMIZE TABLE
RENAME TABLE
RENAME USER
REPAIR TABLE
RESET
REVOKE
SET PASSWORD
SHUTDOWN
START SLAVE
START TRANSACTION
STOP SLAVE
TRUNCATE TABLE
```

```
SET autocommit = 1
causes an implicit commit if the value was 0.
```

All these statements cause an implicit commit before execution. This means that, even if the statement fails with an error, the transaction is committed. Some of them, like

```
CREATE TABLE ... SELECT
, also cause a commit immediately after execution. Such statements couldn't be rolled back in any case.
```

If you are not sure whether a statement has implicitly committed the current transaction, you can query the [in\\_transaction](#) server system variable.

Note that when a transaction starts (not in autocommit mode), all locks acquired with [LOCK TABLES](#) are released. And acquiring such locks always commits the current transaction. To preserve the data integrity between transactional and non-transactional tables, the [GET\\_LOCK\(\)](#) function can be used.

## Exceptions

These statements do not cause an implicit commit in the following cases:

- CREATE TABLE  
and  
DROP TABLE  
, when the

- TEMPORARY keyword is used.
- However, TRUNCATE TABLE causes an implicit commit even when used on a temporary table.
- CREATE FUNCTION and DROP FUNCTION, when used to create a UDF (instead of a stored function). However, CREATE INDEX and DROP INDEX cause commits even when used with temporary tables.
- UNLOCK TABLES causes a commit only if a LOCK TABLES was used on non-transactional tables.
- START SLAVE,  
STOP SLAVE,  
RESET SLAVE and CHANGE MASTER TO only cause implicit commit since MariaDB 10.0.

## 1.1.9.9 Transaction Timeouts

MariaDB has always had the `wait_timeout` and `interactive_timeout` settings, which close connections after a certain period of inactivity.

However, these are by default set to a long wait period. In situations where transactions may be started, but not committed or rolled back, more granular control and a shorter timeout may be desirable so as to avoid locks being held for too long.

MariaDB 10.3 introduced three new variables to handle this situation.

- `idle_transaction_timeout` (all transactions)
- `idle_write_transaction_timeout` (write transactions - called `idle_readwrite_transaction_timeout` until MariaDB 10.3.2)
- `idle_READONLY_transaction_timeout` (read transactions)

These accept a time in seconds to time out, by closing the connection, transactions that are idle for longer than this period. By default all are set to zero, or no timeout.

`idle_transaction_timeout` affects all transactions, `idle_write_transaction_timeout` affects write transactions only and `idle_READONLY_transaction_timeout` affects read transactions only. The latter two variables work independently. However, if either is set along with `idle_transaction_timeout`, the settings for `idle_write_transaction_timeout` or `idle_READONLY_transaction_timeout` will take precedence.

## Examples

```
SET SESSION idle_transaction_timeout=2;
BEGIN;
SELECT * FROM t;
Empty set (0.000 sec)
## wait 3 seconds
SELECT * FROM t;
ERROR 2006 (HY000): MySQL server has gone away
```

```
SET SESSION idle_write_transaction_timeout=2;
BEGIN;
SELECT * FROM t;
Empty set (0.000 sec)
## wait 3 seconds
SELECT * FROM t;
Empty set (0.000 sec)
INSERT INTO t VALUES(1);
## wait 3 seconds
SELECT * FROM t;
ERROR 2006 (HY000): MySQL server has gone away
```

```
SET SESSION idle_transaction_timeout=2, SESSION idle_READONLY_transaction_timeout=10;
BEGIN;
SELECT * FROM t;
Empty set (0.000 sec)
## wait 3 seconds
SELECT * FROM t;
Empty set (0.000 sec)
## wait 11 seconds
SELECT * FROM t;
ERROR 2006 (HY000): MySQL server has gone away
```

## 1.1.9.10 UNLOCK TABLES

### Syntax

[UNLOCK TABLES](#)

### Contents

1. [Syntax](#)
2. [Description](#)

### Description

[UNLOCK TABLES](#)

explicitly releases any table locks held by the current session. See

[LOCK TABLES](#)

for more information.

In addition to releasing table locks acquired by the

[LOCK TABLES](#)

statement, the

[UNLOCK TABLES](#)

statement also releases the global read lock acquired by the

[FLUSH TABLES WITH READ LOCK](#)

statement. The

[FLUSH TABLES WITH READ LOCK](#)

statement is very useful for performing backups. See

[FLUSH](#)

for more information about

[FLUSH TABLES WITH READ LOCK](#)

## 1.1.9.11 WAIT and NOWAIT

MariaDB starting with [10.3.0](#)

MariaDB 10.3.0 introduced extended syntax so that it is possible to set [innodb\\_lock\\_wait\\_timeout](#) and [lock\\_wait\\_timeout](#) for the following statements:

### Syntax

```
ALTER TABLE tbl_name [WAIT n|NOWAIT] ...
CREATE ... INDEX ON tbl_name (index_col_name, ...) [WAIT n|NOWAIT] ...
DROP INDEX ... [WAIT n|NOWAIT]
DROP TABLE tbl_name [WAIT n|NOWAIT] ...
LOCK TABLE ... [WAIT n|NOWAIT]
OPTIMIZE TABLE tbl_name [WAIT n|NOWAIT]
RENAME TABLE tbl_name [WAIT n|NOWAIT] ...
SELECT ... FOR UPDATE [WAIT n|NOWAIT]
SELECT ... LOCK IN SHARE MODE [WAIT n|NOWAIT]
TRUNCATE TABLE tbl_name [WAIT n|NOWAIT]
```

## Description

The lock wait timeout can be explicitly set in the statement by using either

```
WAIT n
(to set the wait in seconds) or
NOWAIT
, in which case the statement will immediately fail if the lock cannot be obtained.
WAIT 0
is equivalent to
NOWAIT
```

## See Also

- [Query Limits and Timeouts](#)
- [ALTER TABLE](#)
- [CREATE INDEX](#)
- [DROP INDEX](#)
- [DROP TABLE](#)
- [LOCK TABLES and UNLOCK TABLES](#)
- [OPTIMIZE TABLE](#)
- [RENAME TABLE](#)
- [SELECT](#)
- [TRUNCATE TABLE](#)

## 1.1.9.12 XA Transactions

### Contents

1. [Overview](#)
2. [Internal XA vs External XA](#)
3. [Transaction Coordinator Log](#)
4. [Syntax](#)
5. [XA RECOVER](#)
6. [Examples](#)
7. [Known Issues](#)
  1. [MariaDB Galera Cluster](#)

## Overview

The MariaDB XA implementation is based on the X/Open CAE document Distributed Transaction Processing: The XA Specification. This document is published by The Open Group and available at <http://www.opengroup.org/public/pubs/catalog/c193.htm>.

XA transactions are designed to allow distributed transactions, where a transaction manager (the application) controls a transaction which involves multiple resources. Such resources are usually DBMSs, but could be resources of any type. The whole set of required transactional operations is called a global transaction. Each subset of operations which involve a single resource is called a local transaction. XA used a 2-phases commit (2PC). With the first commit, the transaction manager tells each resource to prepare an effective commit, and waits for a confirm message. The changes are not still made effective at this point. If any of the resources encountered an error, the transaction manager will rollback the global transaction. If all resources communicate that the first commit is successful, the transaction manager can require a second commit, which makes the changes effective.

In MariaDB, XA transactions can only be used with storage engines that support them. At least [InnoDB](#) , [TokuDB](#) , [SPIDER](#) and [MyRocks](#) support them. For InnoDB, until [MariaDB 10.2](#) , XA transactions can be disabled by setting the `innodb_support_xa` server system variable to 0. From [MariaDB 10.3](#) , XA transactions are always supported.

Like regular transactions, XA transactions create [metadata locks](#) on accessed tables.

XA transactions require [REPEATABLE READ](#) as a minimum isolation level. However, distributed transactions should always use [SERIALIZABLE](#).

Trying to start more than one XA transaction at the same time produces a 1400 error ( [SQLSTATE 'XAE09'](#) ). The same error is produced when attempting to start an XA transaction while a regular transaction is in effect. Trying to start a regular transaction while an XA transaction is in effect produces a 1399 error ( [SQLSTATE 'XAE07'](#) ).

The [statements that cause an implicit COMMIT](#) for regular transactions produce a 1400 error ( [SQLSTATE 'XAE09'](#)) if a XA transaction is in effect.

## Internal XA vs External XA

XA transactions are an overloaded term in MariaDB. If a [storage engine](#) is XA-capable, it can mean one or both of these:

- It supports MariaDB's internal two-phase commit API. This is transparent to the user. Sometimes this is called "internal XA", since MariaDB's [internal transaction coordinator log](#) can handle coordinating these transactions.
- It supports XA transactions, with the

XA START

,

XA PREPARE

,

XA COMMIT

, etc. statements. Sometimes this is called "external XA", since it requires the use of an external transaction coordinator to use this feature properly.

## Transaction Coordinator Log

If you have two or more XA-capable storage engines enabled, then a transaction coordinator log must be available.

There are currently two implementations of the transaction coordinator log:

- Binary log-based transaction coordinator log
- Memory-mapped file-based transaction coordinator log

If the [binary log](#) is enabled on a server, then the server will use the binary log-based transaction coordinator log. Otherwise, it will use the memory-mapped file-based transaction coordinator log.

See [Transaction Coordinator Log](#) for more information.

## Syntax

```
XA {START|BEGIN} xid [JOIN|RESUME]  
  
XA END xid [SUSPEND [FOR MIGRATE]]  
  
XA PREPARE xid  
  
XA COMMIT xid [ONE PHASE]  
  
XA ROLLBACK xid  
  
XA RECOVER [FORMAT=['RAW' | 'SQL']]  
  
xid: gtrid [, bqual [, formatID ]]
```

The interface to XA transactions is a set of SQL statements starting with

XA

. Each statement changes a transaction's state, determining which actions it can perform. A transaction which does not exist is in the NON-EXISTING state.

XA START

(or

BEGIN

) starts a transaction and defines its

xid

(a transaction identifier). The

JOIN

or

RESUME

keywords have no effect. The new transaction will be in

ACTIVE

state.

The

xid

can have 3 components, though only the first one is mandatory.

gtrid

is a quoted string representing a global transaction identifier.

bqual  
is a quoted string representing a local transaction identifier.  
formatID  
is an unsigned integer indicating the format used for the first two components; if not specified, defaults to 1. MariaDB does not interpret in any way these components, and only uses them to identify a transaction.  
xid  
s of transactions in effect must be unique.

XA END  
declares that the specified  
ACTIVE  
transaction is finished and it changes its state to  
IDLE  
  
SUSPEND [FOR MIGRATE]  
has no effect.

XA PREPARE  
prepares an  
IDLE  
transaction for commit, changing its state to  
PREPARED  
. This is the first commit.

XA COMMIT  
definitely commits and terminates a transaction which has already been  
PREPARED  
. If the  
ONE PHASE  
clause is specified, this statement performs a 1-phase commit on an  
IDLE  
transaction.

XA ROLLBACK  
rolls back and terminates an  
IDLE  
or  
PREPARED  
transaction.

XA RECOVER  
shows information about all  
PREPARED  
transactions.

When trying to execute an operation which is not allowed for the transaction's current state, an error is produced:

```
XA COMMIT 'test' ONE PHASE;  
ERROR 1399 (XAE07): XAER_RMFAIL: The command cannot be executed when global transaction is in the ACTIVE state  
  
XA COMMIT 'test2';  
ERROR 1399 (XAE07): XAER_RMFAIL: The command cannot be executed when global transaction is in the NON-EXISTING state
```

## XA RECOVER

The  
XA RECOVER  
statement shows information about all transactions which are in the  
PREPARED  
state. It does not matter which connection created the transaction: if it has been  
PREPARED  
, it appears. But this does not mean that a connection can commit or rollback a transaction which was started by another connection. Note that transactions using a 1-phase commit are never in the  
PREPARED

state, so they cannot be shown by

XA RECOVER

.

XA RECOVER

produces four columns:

```
XA RECOVER;
+-----+-----+-----+-----+
| formatID | gtrid_length | bqual_length | data |
+-----+-----+-----+-----+
|      1 |          4 |          0 | test  |
+-----+-----+-----+
```

MariaDB starting with 10.3.3

You can use

```
XA RECOVER FORMAT='SQL'
to get the data in a human readable form that can be directly copy-pasted into
XA COMMIT
or
XA ROLLBACK
. This is particularly useful for binary
xid
generated by some transaction coordinators.
```

formatID  
is the  
formatID  
part of  
xid

.

data  
are the  
gtrid  
and  
bqual  
parts of  
xid  
, concatenated.

gtrid\_length  
and  
bqual\_length  
are the lengths of  
gtrid  
and  
bqual  
, respectively.

## Examples

2-phases commit:

```
XA START 'test';
INSERT INTO t VALUES (1,2);
XA END 'test';
XA PREPARE 'test';
XA COMMIT 'test';
```

1-phase commit:

```

XA START 'test';

INSERT INTO t VALUES (1,2);

XA END 'test';

XA COMMIT 'test' ONE PHASE;

```

Human-readable:

```

xa start '12\r34\t67\v78', 'abc\ndef', 3;

insert t1 values (40);

xa end '12\r34\t67\v78', 'abc\ndef', 3;

xa prepare '12\r34\t67\v78', 'abc\ndef', 3;

xa recover format='RAW';
+-----+-----+-----+
| formatID | gtrid_length | bqual_length | data           |
+-----+-----+-----+
| 34      |       11        |         7       | 12             |
def |
+-----+-----+-----+
xa recover format='SQL';
+-----+-----+-----+
| formatID | gtrid_length | bqual_length | data           |
+-----+-----+-----+
| 3       |       11        |         7       | X'31320d3334093637763738',X'6162630a646566',3 |
+-----+-----+-----+
xa rollback X'31320d3334093637763738',X'6162630a646566',3;

```

## Known Issues

### MariaDB Galera Cluster

[MariaDB Galera Cluster](#) does not support XA transactions.

However, [MariaDB Galera Cluster](#) builds include a built-in plugin called

wsrep

Prior to [MariaDB 10.4.3](#), this plugin was internally considered an [XA-capable storage engine](#). Consequently, these [MariaDB Galera Cluster](#) builds have multiple XA-capable storage engines by default, even if the only "real" storage engine that supports external [XA transactions](#) enabled on these builds by default is [InnoDB](#). Therefore, when using one of these builds MariaDB would be forced to use a [transaction coordinator log](#) by default, which could have performance implications.

See [Transaction Coordinator Log Overview: MariaDB Galera Cluster](#) for more information.

## 1.1.10 HELP Command

## 1.1.11 Comment Syntax

There are three supported comment styles in MariaDB:

- From a '#'

#  
' to the end of a line:

```
SELECT * FROM users; # This is a comment
```

- From a '--'

--  
' to the end of a line. The space after the two dashes is required (as in MySQL).

```
SELECT * FROM users; -- This is a comment
```

- C style comments from an opening '/\*'

/\*  
' to a closing '\*/'

```
/*
'. Comments of this form can span multiple lines:
```

```
SELECT * FROM users; /* This is a
multi-line
comment */
```

Nested comments are possible in some situations, but they are not supported or recommended.

## Executable Comments

As an aid to portability between different databases, MariaDB supports executable comments. These special comments allow you to embed SQL code which will not execute when run on other databases, but will execute when run on MariaDB.

MariaDB supports both MySQL's executable comment format, and a slightly modified version specific to MariaDB. This way, if you have SQL code that works on MySQL and MariaDB, but not other databases, you can wrap it in a MySQL executable comment, and if you have code that specifically takes advantage of features only available in MariaDB you can use the MariaDB specific format to hide the code from MySQL.

### Executable Comment Syntax

MySQL and MariaDB executable comment syntax:

```
/*! MySQL or MariaDB-specific code */
```

Code that should be executed only starting from a specific MySQL or MariaDB version:

```
/*!##### MySQL or MariaDB-specific code */
```

The numbers, represented by '

#####

' in the syntax examples above specify the specific the minimum versions of MySQL and MariaDB that should execute the comment. The first number is the major version, the second 2 numbers are the minor version and the last 2 is the patch level.

For example, if you want to embed some code that should only execute on MySQL or MariaDB starting from 5.1.0, you would do the following:

```
/*!50100 MySQL and MariaDB 5.1.0 (and above) code goes here. */
```

MariaDB-only executable comment syntax (starting from [MariaDB 5.3.1](#) ):

```
/*M! MariaDB-specific code */
/*M!##### MariaDB-specific code */
```

MariaDB ignores MySQL-style executable comments that have a version number in the range

50700..99999

. This is needed to skip features introduced in MySQL-5.7 that are not ported to MariaDB 10.x yet.

```
/*!50701 MariaDB-10.x ignores MySQL-5.7 specific code */
```

**Note:** comments which have a version number in the range

50700..99999

that use MariaDB-style executable comment syntax are still executed.

```
/*M!50701 MariaDB-10.x does not ignore this */
```

Statement delimiters cannot be used within executable comments.

## Examples

In MySQL all the following will return 2: In MariaDB, the last 2 queries would return 3.

```
SELECT 2 /* +1 */;
SELECT 1 /*! +1 */;
SELECT 1 /*!50101 +1 */;
SELECT 2 /*M! +1 */;
SELECT 2 /*M!50301 +1 */;
```

The following executable statement will not work due to the delimiter inside the executable portion:

```
/*M!100100 select 1 ; */
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for t
```

Instead, the delimiter should be placed outside the executable portion:

```
/*M!100100 select 1 *;
+---+
| 1 |
+---+
| 1 |
+---+
```

## 1.1.12 Built-in Functions

### 1.1.12.1 Function and Operator Reference

Name	Description
+	Addition operator
/	Division operator
*	Multiplication operator
%	Modulo operator. Returns the remainder of N divided by M
-	Subtraction operator
!=	Not equals
<	Less than
<=	Less than or equal
<>	NULL-safe equal
=	Equal
>	Greater than
>=	Greater than or equal
&	Bitwise AND
<<	Shift left
>>	Shift right
^	Bitwise XOR
!	Logical NOT
&&	Logical AND
XOR	Logical XOR
	Logical OR
	Bitwise OR
:=	Assignment operator
=	Assignment and comparison operator
~	Bitwise NOT
ABS	Returns an absolute value
ACOS	Returns an arc cosine
ADD_MONTHS	Add months to a date
ADDDATE	Add days or another interval to a date
ADDTIME	Adds a time to a time or datetime
AES_DECRYPT	Decryption data encrypted with AES_ENCRYPT
AES_ENCRYPT	Encrypts a string with the AES algorithm
AREA	Synonym for ST_AREA
AsBinary	Synonym for ST_AsBinary
ASCII	Numeric ASCII value of leftmost character

ASIN	Returns the arc sine
AsText	Synonym for ST_AsText
AsWKB	Synonym for ST_AsBinary
AsWKT	Synonym for ST_AsText
ATAN	Returns the arc tangent
ATAN2	Returns the arc tangent of two variables
AVG	Returns the average value
BENCHMARK	Executes an expression repeatedly
BETWEEN AND	True if expression between two values
BIN	Returns binary value
BINARY OPERATOR	Casts to a binary string
BINLOG_GTID_POS	Returns a string representation of the corresponding GTID position
BIT_AND	Bitwise AND
BIT_COUNT	Returns the number of set bits
BIT_LENGTH	Returns the length of a string in bits
BIT_OR	Bitwise OR
BIT_XOR	Bitwise XOR
BOUNDARY	Synonym for ST_BOUNDARY
BUFFER	Synonym for ST_BUFFER
CASE	Returns the result where value=compare_value or for the first condition that is true
CAST	Casts a value of one type to another type
CEIL	Synonym for CEILING()
CEILING	Returns the smallest integer not less than X
CENTROID	Synonym for ST_CENTROID
CHAR Function	Returns string based on the integer values for the individual characters
CHARACTER_LENGTH	Synonym for CHAR_LENGTH()
CHAR_LENGTH	Length of the string in characters
CHARSET	Returns the character set
CHR	Returns a string consisting of the character given by the code values of the integer
COALESCE	Returns the first non-NULL parameter
COERCIBILITY	Returns the collation coercibility value
COLLATION	Collation of the string argument
COLUMN_ADD	Adds or updates dynamic columns
COLUMN_CHECK	Checks if a dynamic column blob is valid
COLUMN_CREATE	Returns a dynamic columns blob
COLUMN_DELETE	Deletes a dynamic column
COLUMN_EXISTS	Checks if a column exists
COLUMN_GET	Gets a dynamic column value by name
COLUMN_JSON	Returns a JSON representation of dynamic column blob data
COLUMN_LIST	Returns comma-separated list
COMPRESS	Returns a binary, compressed string
CONCAT	Returns concatenated string
CONCAT_WS	Concatenate with separator
CONNECTION_ID	Connection thread ID
CONTAINS	Whether one geometry contains another
CONVERT	Convert a value from one type to another type
CONV	Converts numbers between different number bases
CONVERT_TZ	Converts a datetime from one time zone to another
CONVEXHULL	Synonym for ST_CONVEXHULL
COS	Returns the cosine
COT	Returns the cotangent

COUNT	Returns count of non-null values
COUNT DISTINCT	Returns count of number of different non-NULL values
CRC32	Computes a cyclic redundancy check value
CROSSES	Whether two geometries spatially cross
CUME_DIST	Window function that returns the cumulative distribution of a given row
CURDATE	Returns the current date
CURRENT_DATE	Synonym for CURDATE()
CURRENT_ROLE	Current role name
CURRENT_TIME	Synonym for CURTIME()
CURRENT_TIMESTAMP	Synonym for NOW()
CURRENT_USER	Username/host that authenticated the current client
CURTIME	Returns the current time
DATABASE	Current default database
DATE FUNCTION	Extracts the date portion of a datetime
DATEDIFF	Difference in days between two date/time values
DATE_ADD	Date arithmetic - addition
DATE_FORMAT	Formats the date value according to the format string
DATE_SUB	Date arithmetic - subtraction
DAY	Synonym for DAYOFMONTH()
DAYNAME	Return the name of the weekday
DAYOFMONTH	Returns the day of the month
DAYOFWEEK	Returns the day of the week index
DAYOFYEAR	Returns the day of the year
DECODE	Decrypts a string encoded with ENCODE()
DECODE_HISTOGRAM	Returns comma separated numerics corresponding to a probability distribution represented by a histogram
DEFAULT	Returns column default
DEGREES	Converts from radians to degrees
DENSE_RANK	Rank of a given row with identical values receiving the same result, no skipping
DES_DECRYPT	Decrypts a string encrypted with DES_ENCRYPT()
DES_ENCRYPT	Encrypts a string using the Triple-DES algorithm
DIMENSION	Synonym for ST_DIMENSION
DISJOINT	Whether the two elements do not intersect
DIV	Integer division
ELT	Returns the N'th element from a set of strings
ENCODE	Encrypts a string
ENCRYPT	Encrypts a string with Unix crypt()
ENDPOINT	Synonym for ST_ENDPOINT
ENVELOPE	Synonym for ST_ENVELOPE
EQUALS	Indicates whether two geometries are spatially equal
EXP	e raised to the power of the argument
EXPORT_SET	Returns an on string for every bit set, an off string for every bit not set
ExteriorRing	Synonym for ST_ExteriorRing
EXTRACT	Extracts a portion of the date
EXTRACTVALUE	Returns the text of the first text node matched by the XPath expression
FIELD	Returns the index position of a string in a list
FIND_IN_SET	Returns the position of a string in a set of strings
FLOOR	Largest integer value not greater than the argument
FORMAT	Formats a number
FOUND_ROWS	Number of (potentially) returned rows
FROM_BASE64	Given a base-64 encoded string, returns the decoded result as a binary string

FROM_DAYS	Returns a date given a day
FROM_UNIXTIME	Returns a datetime from a Unix timestamp
GeomCollFromText	Synonym for ST_GeomCollFromText
GeomCollFromWKB	Synonym for ST_GeomCollFromWKB
GeometryCollectionFromText	Synonym for ST_GeomCollFromText
GeometryCollectionFromWKB	Synonym for ST_GeomCollFromWKB
GeometryFromText	Synonym for ST_GeomFromText
GeometryFromWKB	Synonym for ST_GeomFromWKB
GeomFromText	Synonym for ST_GeomFromText
GeomFromWKB	Synonym for ST_GeomFromWKB
GeometryN	Synonym for ST_GeometryN
GEOMETRYCOLLECTION	Constructs a WKB GeometryCollection
GeometryType	Synonym for ST_GeometryType
GET_FORMAT	Returns a format string
GET_LOCK	Obtain LOCK
GLENGTH	Length of a LineString value
GREATEST	Returns the largest argument
GROUP_CONCAT	Returns string with concatenated values from a group
HEX	Returns hexadecimal value
HOUR	Returns the hour
IF	If expr1 is TRUE, returns expr2; otherwise returns expr3
IFNULL	Check whether an expression is NULL
IN	True if expression equals any of the values in the list
INTERVAL	Index of the argument that is less than the first argument
INET6_ATON	Given an IPv6 or IPv4 network address, returns a VARBINARY numeric value
INET6_NTOA	Given an IPv6 or IPv4 network address, returns the address as a nonbinary string
INET_ATON	Returns numeric value of IPv4 address
INET_NTOA	Returns dotted-quad representation of IPv4 address
INSERT Function	Replaces a part of a string with another string
INSTR	Returns the position of a string within a string
InteriorRingN	Synonym for ST_InteriorRingN
INTERSECTS	Indicates whether two geometries spatially intersect
IS	Tests whether a boolean is TRUE, FALSE, or UNKNOWN
IsClosed	Synonym for ST_IsClosed
IsEmpty	Synonym for ST_IsEmpty
IS_FREE_LOCK	Checks whether lock is free to use
IS_IPV4	Whether or not an expression is a valid IPv4 address
IS_IPV4_COMPAT	Whether or not an IPv6 address is IPv4-compatible
IS_IPV4_MAPPED	Whether an IPv6 address is a valid IPv4-mapped address
IS_IPV6	Whether or not an expression is a valid IPv6 address
IS NOT	Tests whether a boolean value is not TRUE, FALSE, or UNKNOWN
IS NOT NULL	Tests whether a value is not NULL
IS NULL	Tests whether a value is NULL
ISNULL	Checks if an expression is NULL
IsRing	Synonym for ST_IsRing
IsSimple	Synonym for ST_IsSimple
IS_USED_LOCK	Check if lock is in use
JSON_ARRAY	Returns a JSON array containing the listed values
JSON_ARRAY_APPEND	Appends values to the end of the given arrays within a JSON document
JSON_ARRAY_INSERT	Inserts a value into a JSON document

JSON_COMPACT	Removes all unnecessary spaces so the json document is as short as possible
JSON_CONTAINS	Whether a value is found in a given JSON document or at a specified path within the document
JSON_CONTAINS_PATH	Indicates whether the given JSON document contains data at the specified path or paths
JSON_DEPTH	Maximum depth of a JSON document
JSON_DETAILED	Represents JSON in the most understandable way emphasizing nested structures
JSON_EQUALS	Check for equality between JSON objects.
JSON_EXISTS	Determines whether a specified JSON value exists in the given data
JSON_EXTRACT	Extracts data from a JSON document.
JSON_INSERT	Inserts data into a JSON document
JSON_KEYS	Returns keys from top-level value of a JSON object or top-level keys from the path
JSON_LENGTH	Returns the length of a JSON document, or the length of a value within the document
JSON_LOOSE	Adds spaces to a JSON document to make it look more readable
JSON_MERGE	Merges the given JSON documents
JSON_MERGE_PATCH	RFC 7396-compliant merge of the given JSON documents
JSON_MERGE_PRESERVE	Synonym for <code>JSON_MERGE_PATCH</code> .
JSON_NORMALIZE	Recursively sorts keys and removes spaces, allowing comparison of json documents for equality
JSON_OBJECT	Returns a JSON object containing the given key/value pairs
JSON_OBJECTAGG	Returns a JSON object containing key-value pairs
JSON_OVERLAPS	Compares two json documents for overlaps
JSON_QUERY	Given a JSON document, returns an object or array specified by the path
JSON_QUOTE	Quotes a string as a JSON value
JSON_REMOVE	Removes data from a JSON document
JSON_REPLACE	Replaces existing values in a JSON document
JSON_SEARCH	Returns the path to the given string within a JSON document
JSON_SET	Updates or inserts data into a JSON document
JSON_TABLE	Returns a representation of a JSON document as a relational table
JSON_TYPE	Returns the type of a JSON value
JSON_UNQUOTE	Unquotes a JSON value, returning a string
JSON_VALID	Whether a value is a valid JSON document or not
JSON_VALUE	Given a JSON document, returns the specified scalar
LAST_DAY	Returns the last day of the month
LAST_INSERT_ID	Last inserted autoinc value
LAST_VALUE	Returns the last value in a list
LASTVAL	Get last value generated from a sequence
LCASE	Synonym for <code>[LOWER()]</code>
LEAST	Returns the smallest argument
LEFT	Returns the leftmost characters from a string
LENGTH	Length of the string in bytes
LIKE	Whether expression matches a pattern
LineFromText	Synonym for <code>ST_LineFromText</code>
LineFromWKB	Synonym for <code>ST_LineFromWKB</code>
LINESTRING	Constructs a WKB LineString value from a number of WKB Point arguments
LineStringFromText	Synonym for <code>ST_LineFromText</code>
LineStringFromWKB	Synonym for <code>ST_LineFromWKB</code>
LN	Returns natural logarithm
LOAD_FILE	Returns file contents as a string
LOCALTIME	Synonym for <code>NOW()</code>
LOCALTIMESTAMP	Synonym for <code>NOW()</code>
LOCATE	Returns the position of a substring in a string
LOG	Returns the natural logarithm

LOG10	Returns the base-10 logarithm
LOG2	Returns the base-2 logarithm
LOWER	Returns a string with all characters changed to lowercase
LPAD	Returns the string left-padded with another string to a given length
LTRIM	Returns the string with leading space characters removed
MAKE_SET	Make a set of strings that matches a bitmask
MAKEDATE	Returns a date given a year and day
MAKETIME	Returns a time
MASTER_GTID_WAIT	Wait until slave reaches the GTID position
MASTER_POS_WAIT	Blocks until the slave has applied all specified updates
MATCH AGAINST	Perform a fulltext search on a fulltext index
MAX	Returns the maximum value
MBRContains	Indicates one Minimum Bounding Rectangle contains another
MBRDisjoint	Indicates whether the Minimum Bounding Rectangles of two geometries are disjoint
MBREqual	Whether the Minimum Bounding Rectangles of two geometries are the same.
MBRIntersects	Indicates whether the Minimum Bounding Rectangles of the two geometries intersect
MBROverlaps	Whether the Minimum Bounding Rectangles of two geometries overlap.
MBRTouches	Whether the Minimum Bounding Rectangles of two geometries touch.
MBRWithin	Indicates whether one Minimum Bounding Rectangle is within another
MD5	MD5 checksum
MEDIAN	Window function that returns the median value of a range of values
MICROSECOND	Returns microseconds from a date or datetime
MID	Synonym for SUBSTRING(str,pos,len)
MIN	Returns the minimum value
MINUTE	Returns a minute from 0 to 59
MLineFromText	Constructs MULTILINESTRING using its WKT representation and SRID
MLineFromWKB	Constructs a MULTILINESTRING
MOD	Modulo operation. Remainder of N divided by M
MONTH	Returns a month from 1 to 12
MONTHNAME	Returns the full name of the month
MPointFromText	Constructs a MULTIPOINT value using its WKT and SRID
MPointFromWKB	Constructs a MULTIPOINT value using its WKB representation and SRID
MPolyFromText	Constructs a MULTIPOLYGON value
MPolyFromWKB	Constructs a MULTIPOLYGON value using its WKB representation and SRID
MultiLineStringFromText	Synonym for MLineFromText
MultiLineStringFromWKB	A synonym for MLineFromWKB
MULTIPOINT	Constructs a WKB MultiPoint value
MultiPointFromText	Synonym for MPointFromText
MultiPointFromWKB	Synonym for MPointFromWKB
MULTIPOLYGON	Constructs a WKB MultiPolygon
MultiPolygonFromText	Synonym for MPolyFromText
MultiPolygonFromWKB	Synonym for MPolyFromWKB
MULTILINESTRING	Constructs a MultiLineString value
NAME_CONST	Returns the given value
NATURAL_SORT_KEY	Sorting that is more more similar to natural human sorting
NOT LIKE	Same as NOT(expr LIKE pat [ESCAPE 'escape_char'])
NOT REGEXP	Same as NOT (expr REGEXP pat)
NULLIF	Returns NULL if expr1 = expr2
NEXTVAL	Generate next value for sequence
NOT BETWEEN	Same as NOT (expr BETWEEN min AND max)
NOT IN	Same as NOT (expr IN (value,...))

NOW	Returns the current date and time
NTILE	Returns an integer indicating which group a given row falls into
NumGeometries	Synonym for ST_NumGeometries
NumInteriorRings	Synonym for NumInteriorRings
NumPoints	Synonym for ST_NumPoints
OCT	Returns octal value
OCTET_LENGTH	Synonym for LENGTH()
OLD_PASSWORD	Pre MySQL 4.1 password implementation
ORD	Return ASCII or character code
OVERLAPS	Indicates whether two elements spatially overlap
PASSWORD	Calculates a password string
PERCENT_RANK	Window function that returns the relative percent rank of a given row
PERCENTILE_CONT	Returns a value which corresponds to the given fraction in the sort order.
PERCENTILE_DISC	Returns the first value in the set whose ordered position is the same or more than the specified fraction.
PERIOD_ADD	Add months to a period
PERIOD_DIFF	Number of months between two periods
PI	Returns the value of π (pi)
POINT	Constructs a WKB Point
PointFromText	Synonym for ST_PointFromText
PointFromWKB	Synonym for PointFromWKB
PointN	Synonym for PointN
PointOnSurface	Synonym for ST_PointOnSurface
POLYGON	Constructs a WKB Polygon value from a number of WKB LineString arguments
PolyFromText	Synonym for ST_PolyFromText
PolyFromWKB	Synonym for ST_PolyFromWKB
PolygonFromText	Synonym for ST_PolyFromText
PolygonFromWKB	Synonym for ST_PolyFromWKB
POSITION	Returns the position of a substring in a string
POW	Returns X raised to the power of Y
POWER	Synonym for POW()
QUARTER	Returns year quarter from 1 to 4
QUOTE	Returns quoted, properly escaped string
RADIANS	Converts from degrees to radians
RAND	Random floating-point value
RANK	Rank of a given row with identical values receiving the same result
REGEXP	Performs pattern matching
REGEXP_INSTR	Position of the first appearance of a regex
REGEXP_REPLACE	Replaces all occurrences of a pattern
REGEXP_SUBSTR	Returns the matching part of a string
RELEASE_LOCK	Releases lock obtained with GET_LOCK()
REPEAT Function	Returns a string repeated a number of times
REPLACE Function	Replace occurrences of a string
REVERSE	Reverses the order of a string
RIGHT	Returns the rightmost N characters from a string
RLIKE	Synonym for REGEXP()
RPAD	Returns the string right-padded with another string to a given length
ROUND	Rounds a number
ROW_COUNT	Number of rows affected by previous statement
ROW_NUMBER	Row number of a given row with identical values receiving a different result
RTRIM	Returns the string with trailing space characters removed

SCHEMA	Synonym for DATABASE()
SECOND	Returns the second of a time
SEC_TO_TIME	Converts a second to a time
SETVAL	Set the next value to be returned by a sequence
SESSION_USER	Synonym for USER()
SHA	Synonym for SHA1()
SHA1	Calculates an SHA-1 checksum
SHA2	Calculates an SHA-2 checksum
SIGN	Returns 1, 0 or -1
SIN	Returns the sine
SLEEP	Pauses for the given number of seconds
SOUNDEX	Returns a string based on how the string sounds
SOUNDS LIKE	SOUNDEX(expr1) = SOUNDEX(expr2)
SPACE	Returns a string of space characters
SPIDER_BG_DIRECT_SQL	Background SQL execution
SPIDER_COPY_TABLES	Copy table data
SPIDER_DIRECT_SQL	Execute SQL on the remote server
SPIDER_FLUSH_TABLE_MON_CACHE	Refreshing Spider monitoring server information
SQRT	Square root
SRID	Synonym for ST_SRID
ST_AREA	Area of a Polygon
ST_AsBinary	Converts a value to its WKB representation
ST_AsText	Converts a value to its WKT-Definition
ST_AsWKB	Synonym for ST_AsBinary
ST_ASWKT	Synonym for ST_ASTEXT()
ST_BOUNDARY	Returns a geometry that is the closure of a combinatorial boundary
ST_BUFFER	A new geometry with a buffer added to the original geometry
ST_CENTROID	The mathematical centroid (geometric center) for a MultiPolygon
ST_CONTAINS	Whether one geometry is contained by another
ST_CONVEXHULL	The minimum convex geometry enclosing all geometries within the set
ST_CROSSES	Whether two geometries spatially cross
ST_DIFFERENCE	Point set difference
ST_DIMENSION	Inherent dimension of a geometry value
ST_DISJOINT	Whether one geometry is spatially disjoint from another
ST_DISTANCE	The distance between two geometries
ST_DISTANCE_SPHERE	The spherical distance between two geometries
ST_ENDPOINT	Returns the endpoint of a LineString
ST_ENVELOPE	Returns the Minimum Bounding Rectangle for a geometry value
ST_EQUALS	Whether two geometries are spatioially equal
ST_ExteriorRing	Returns the exterior ring of a Polygon as a LineString
ST_GeomCollFromText	Constructs a GEOMETRYCOLLECTION value
ST_GeomCollFromWKB	Constructs a GEOMETRYCOLLECTION value from a WKB
ST_GeometryCollectionFromText	Synonym for ST_GeomCollFromText
ST_GeometryCollectionFromWKB	Synonym for ST_GeomCollFromWKB
ST_GeometryFromText	Synonym for ST_GeomFromText
ST_GeometryFromWKB	Synonym for ST_GeomFromWKB
ST_GEOMETRYN	Returns the N-th geometry in a GeometryCollection
ST_GEOMETRYTYPE	Returns name of the geometry type of which a given geometry instance is a member
ST_GeomFromText	Constructs a geometry value using its WKT and SRID
ST_GeomFromWKB	Constructs a geometry value using its WKB representation and SRID
ST_InteriorRingN	Returns the N-th interior ring for a Polygon

ST_INTERSECTION	The intersection, or shared portion, of two geometries
ST_INTERSECTS	Whether two geometries spatially intersect
ST_ISCLOSED	Returns true if a given LINESTRING's start and end points are the same
ST_ISEMPTY	Indicated validity of geometry value
ST_IsRing	Returns true if a given LINESTRING is both ST_IsClosed and ST_IsSimple
ST_IsSimple	Returns true if the given Geometry has no anomalous geometric points
ST_LENGTH	Length of a LineString value
ST_LineFromText	Creates a linestring value
ST_LineFromWKB	Constructs a LINESTRING using its WKB and SRID
ST_LineStringFromText	Synonym for ST_LineFromText
ST_LineStringFromWKB	Synonym for ST_LineFromWKB
ST_NUMGEOMETRIES	Number of geometries in a GeometryCollection
ST_NumInteriorRings	Number of interior rings in a Polygon
ST_NUMPOINTS	Returns the number of Point objects in a LineString
ST_OVERLAPS	Whether two geometries overlap
ST_PointFromText	Constructs a POINT value
ST_PointFromWKB	Constructs POINT using its WKB and SRID
ST_POINTN	Returns the N-th Point in the LineString
ST_POINTONSURFACE	Returns a POINT guaranteed to intersect a surface
ST_PolyFromText	Constructs a POLYGON value
ST_PolyFromWKB	Constructs POLYGON value using its WKB representation and SRID
ST_PolygonFromText	Synonym for ST_PolyFromText
ST_PolygonFromWKB	Synonym for ST_PolyFromWKB
ST_RELATE	Returns true if two geometries are related
ST_SRID	Returns a Spatial Reference System ID
ST_STARTPOINT	Returns the start point of a LineString
ST_SYMDIFFERENCE	Portions of two geometries that don't intersect
ST_TOUCHES	Whether one geometry g1 spatially touches another
ST_UNION	Union of two geometries
ST_WITHIN	Whether one geometry is within another
ST_X	X-coordinate value for a point
ST_Y	Y-coordinate for a point
STARTPOINT	Synonym for ST_StartPoint
STD	Population standard deviation
STDDEV	Population standard deviation
STDDEV_POP	Returns the population standard deviation
STDDEV_SAMP	Standard deviation
STR_TO_DATE	Converts a string to date
STRCMP	Compares two strings in sort order
SUBDATE	Subtract a date unit or number of days
SUBSTR	Returns a substring from string starting at a given position
SUBSTRING	Returns a substring from string starting at a given position
SUBSTRING_INDEX	Returns the substring from string before count occurrences of a delimiter
SUBTIME	Subtracts a time from a date/time
SUM	Sum total
SYS_GUID	Generates a globally unique identifier
SYSDATE	Returns the current date and time
SYSTEM_USER	Synonym for USER()
TAN	Returns the tangent
TIME function	Extracts the time
TIMEDIFF	Returns the difference between two date/times

TIMESTAMP FUNCTION	Return the datetime, or add a time to a date/time
TIMESTAMPADD	Add interval to a date or datetime
TIMESTAMPDIFF	Difference between two datetimes
TIME_FORMAT	Formats the time value according to the format string
TIME_TO_SEC	Returns the time argument, converted to seconds
TO_BASE64	Converts a string to its base-64 encoded form
TO_CHAR	Converts a date/time type to a char
TO_DAYS	Number of days since year 0
TO_SECONDS	Number of seconds since year 0
TOUCHES	Whether two geometries spatially touch
TRIM	Returns a string with all given prefixes or suffixes removed
TRUNCATE	Truncates X to D decimal places
UCASE	Synonym for UPPER]]()
UNHEX	Interprets pairs of hex digits as a number and converts to the character represented by the number
UNCOMPRESS	Uncompresses string compressed with COMPRESS()
UNCOMPRESSED_LENGTH	Returns length of a string before being compressed with COMPRESS()
UNIX_TIMESTAMP	Returns a Unix timestamp
UPDATEXML	Replace XML
UPPER	Changes string to uppercase
USER	Current user/host
UTC_DATE	Returns the current UTC date
UTC_TIME	Returns the current UTC time
UTC_TIMESTAMP	Returns the current UTC date and time
UUID	Returns a Universal Unique Identifier
UUID_SHORT	Return short universal identifier
VALUES or VALUE	Refer to columns in INSERT ... ON DUPLICATE KEY UPDATE
VAR_POP	Population standard variance
VAR_SAMP	Returns the sample variance
VARIANCE	Population standard variance
VERSION	MariaDB server version
WEEK	Returns the week number
WEEKDAY	Returns the weekday index
WEEKOFYEAR	Returns the calendar week of the date as a number in the range from 1 to 53
WEIGHT_STRING	Weight of the input string
WITHIN	Indicate whether a geographic element is spacially within another
WSREP_LAST_SEEN_GTID	Returns the Global Transaction ID of the most recent write transaction observed by the client.
WSREP_LAST_WRITTEN_GTID	Returns the Global Transaction ID of the most recent write transaction performed by the client.
WSREP_SYNC_WAIT_UPTO_GTID	Blocks the client until the transaction specified by the given Global Transaction ID is applied and committed by the node
X	Synonym for ST_X
Y	Synonym for ST_Y
YEAR	Returns the year for the given date
YEARWEEK	Returns year and week for a date

## 1.1.12.2 String Functions

### 1.1.12.2.1 Regular Expressions Functions

#### 1.1.12.2.1.1 Regular Expressions Overview

## Contents

1. [Special Characters](#)
  1. ^
  2. \$
  3. .
  4. \*
  5. +
  6. ?
  7. ()
  8. {}
  9. []
    1. ^
    2. [Word boundaries](#)
    3. [Character Classes](#)
    4. [Character Names](#)
10. [Combining](#)
11. [Escaping](#)

Regular Expressions allow MariaDB to perform complex pattern matching on a string. In many cases, the simple pattern matching provided by [LIKE](#) is sufficient.

`LIKE`

performs two kinds of matches:

- \_
  - the underscore, matching a single character
- %
  - the percentage sign, matching any number of characters.

In other cases you may need more control over the returned matches, and will need to use regular expressions.

MariaDB starting with [10.0.5](#)

Until [MariaDB 10.0.5](#), MariaDB used the POSIX 1003.2 compliant regular expression library. The new PCRE library is mostly backwards compatible with what is described below - see the [PCRE Regular Expressions](#) article for the enhancements made in 10.0.5.

Regular expression matches are performed with the [REGEXP](#) function.

`RLIKE`

is a synonym for

`REGEXP`

Comparisons are performed on the byte value, so characters that are treated as equivalent by a collation, but do not have the same byte-value, such as accented characters, could evaluate as unequal. Also note that until [MariaDB 10.0.5](#), regular expressions were not multi-byte safe, and therefore could produce unexpected results in multi-byte character sets.

Without any special characters, a regular expression match is true if the characters match. The match is case-insensitive, except in the case of `BINARY` strings.

```
SELECT 'Maria' REGEXP 'Maria';
+-----+
| 'Maria' REGEXP 'Maria' |
+-----+
|           1 |
+-----+  
  
SELECT 'Maria' REGEXP 'maria';
+-----+
| 'Maria' REGEXP 'maria' |
+-----+
|           1 |
+-----+  
  
SELECT BINARY 'Maria' REGEXP 'maria';
+-----+
| BINARY 'Maria' REGEXP 'maria' |
+-----+
|           0 |
+-----+
```

Note that the word being matched must match the whole pattern:

```
SELECT 'Maria' REGEXP 'Mari';
```

```
+-----+
| 'Maria' REGEXP 'Mari' |
+-----+
|           1 |
+-----+
```

```
SELECT 'Mari' REGEXP 'Maria';
```

```
+-----+
| 'Mari' REGEXP 'Maria' |
+-----+
|           0 |
+-----+
```

The first returns true because the pattern "Mari" exists in the expression "Maria". When the order is reversed, the result is false, as the pattern "Maria" does not exist in the expression "Mari"

A match can be performed against more than one word with the

|

character. For example:

```
SELECT 'Maria' REGEXP 'Monty|Maria';
```

```
+-----+
| 'Maria' REGEXP 'Monty|Maria' |
+-----+
|           1 |
+-----+
```

## Special Characters

The above examples introduce the syntax, but are not very useful on their own. It's the special characters that give regular expressions their power.

^

^

matches the beginning of a string (inside square brackets it can also mean NOT - see below):

```
SELECT 'Maria' REGEXP '^Ma';
```

```
+-----+
| 'Maria' REGEXP '^Ma' |
+-----+
|           1 |
+-----+
```

\$

\$

matches the end of a string:

```
SELECT 'Maria' REGEXP 'ia$';
```

```
+-----+
| 'Maria' REGEXP 'ia$' |
+-----+
|           1 |
+-----+
```

matches any single character:

```
SELECT 'Maria' REGEXP 'Ma.ia';
```

```
+-----+  
| 'Maria' REGEXP 'Ma.ia' |  
+-----+  
| 1 |  
+-----+
```

```
SELECT 'Maria' REGEXP 'Ma..ia';
```

```
+-----+  
| 'Maria' REGEXP 'Ma..ia' |  
+-----+  
| 0 |  
+-----+
```

\*

x\*

matches zero or more of a character

x

. In the examples below, it's the

r

character.

```
SELECT 'Maria' REGEXP 'Mar*ia';
```

```
+-----+  
| 'Maria' REGEXP 'Mar*ia' |  
+-----+  
| 1 |  
+-----+
```

```
SELECT 'Maia' REGEXP 'Mar*ia';
```

```
+-----+  
| 'Maia' REGEXP 'Mar*ia' |  
+-----+  
| 1 |  
+-----+
```

```
SELECT 'Marrria' REGEXP 'Mar*ia';
```

```
+-----+  
| 'Marrria' REGEXP 'Mar*ia' |  
+-----+  
| 1 |  
+-----+
```

+

x+

matches one or more of a character

x

. In the examples below, it's the

r

character.

```

SELECT 'Maria' REGEXP 'Mar+ia';
+-----+
| 'Maria' REGEXP 'Mar+ia' |
+-----+
|           1 |
+-----+

SELECT 'Maia' REGEXP 'Mar+ia';
+-----+
| 'Maia' REGEXP 'Mar+ia' |
+-----+
|           0 |
+-----+

SELECT 'Marrria' REGEXP 'Mar+ia';
+-----+
| 'Marrria' REGEXP 'Mar+ia' |
+-----+
|           1 |
+-----+

```

?

x?  
 matches zero or one of a character  
 x  
 . In the examples below, it's the  
 r  
 character.

```

SELECT 'Maria' REGEXP 'Mar?ia';
+-----+
| 'Maria' REGEXP 'Mar?ia' |
+-----+
|           1 |
+-----+

SELECT 'Maia' REGEXP 'Mar?ia';
+-----+
| 'Maia' REGEXP 'Mar?ia' |
+-----+
|           1 |
+-----+

SELECT 'Marrria' REGEXP 'Mar?ia';
+-----+
| 'Marrria' REGEXP 'Mar?ia' |
+-----+
|           0 |
+-----+

```

()

(xyz)  
 - combine a sequence, for example  
 (xyz)+  
 or  
 (xyz)\*

```

SELECT 'Maria' REGEXP '(ari)+';
+-----+
| 'Maria' REGEXP '(ari)+' |
+-----+
|           1 |
+-----+

```

{}

```

x{n}
and
x{m,n}
This notation is used to match many instances of the
x
. In the case of
x{n}
the match must be exactly that many times. In the case of
x{m,n}
, the match can occur from
m
to
n
times. For example, to match zero or one instance of the string
ari
(which is identical to
(ari)?
), the following can be used:

```

```

SELECT 'Maria' REGEXP '(ari){0,1}';
+-----+
| 'Maria' REGEXP '(ari){0,1}' |
+-----+
|           1 |
+-----+

```

□

```

[xy]
groups characters for matching purposes. For example, to match either the
p
or the
r
character:

```

```

SELECT 'Maria' REGEXP 'Ma[pr]ia';
+-----+
| 'Maria' REGEXP 'Ma[pr]ia' |
+-----+
|           1 |
+-----+

```

The square brackets also permit a range match, for example, to match any character from a-z,

```

[a-z]
is used. Numeric ranges are also permitted.

```

```

SELECT 'Maria' REGEXP 'Ma[a-z]ia';
+-----+
| 'Maria' REGEXP 'Ma[a-z]ia' |
+-----+
|           1 |
+-----+

```

The following does not match, as

```

r
falls outside of the range
a-p
.
```

```

SELECT 'Maria' REGEXP 'Ma[a-p]ia';
+-----+
| 'Maria' REGEXP 'Ma[a-p]ia' |
+-----+
|           0 |
+-----+

```

^

The

^

character means does

NOT

match, for example:

```
SELECT 'Maria' REGEXP 'Ma[^p]ia';
+-----+
| 'Maria' REGEXP 'Ma[^p]ia' |
+-----+
|           1 |
+-----+

SELECT 'Maria' REGEXP 'Ma[^r]ia';
+-----+
| 'Maria' REGEXP 'Ma[^r]ia' |
+-----+
|           0 |
+-----+
```

The

[  
and  
]  
characters on their own can be literally matched inside a  
[]

block, without escaping, as long as they immediately match the opening bracket:

```
SELECT '[Maria' REGEXP '[[[';
+-----+
| '[Maria' REGEXP '[[[' |
+-----+
|           1 |
+-----+

SELECT '[Maria' REGEXP '][]';
+-----+
| '[Maria' REGEXP '][]' |
+-----+
|           0 |
+-----+

SELECT ']Maria' REGEXP '[]]';
+-----+
| ']Maria' REGEXP '[]]' |
+-----+
|           1 |
+-----+

SELECT ']Maria' REGEXP '[]a]';
+-----+
| ']Maria' REGEXP '[]a]' |
+-----+
|           1 |
+-----+
```

Incorrect order, so no match:

```
SELECT ']Maria' REGEXP '[a]]';
+-----+
| ']Maria' REGEXP '[a]]' |
+-----+
|           0 |
+-----+
```

The

- character can also be matched in the same way:

```

SELECT '-Maria' REGEXP '[1-10]';
+-----+
| '-Maria' REGEXP '[1-10]' |
+-----+
|          0 |
+-----+

SELECT '-Maria' REGEXP '[-1-10]';
+-----+
| '-Maria' REGEXP '[-1-10]' |
+-----+
|          1 |
+-----+

```

## Word boundaries

The

:<:

and

:>:

patterns match the beginning and the end of a word respectively. For example:

```

SELECT 'How do I upgrade MariaDB?' REGEXP '[[[:<:]]MariaDB[[:>:]]';
+-----+
| 'How do I upgrade MariaDB?' REGEXP '[[[:<:]]MariaDB[[:>:]]' |
+-----+
|          1 |
+-----+

SELECT 'How do I upgrade MariaDB?' REGEXP '[[[:<:]]Maria[[:>:]]';
+-----+
| 'How do I upgrade MariaDB?' REGEXP '[[[:<:]]Maria[[:>:]]' |
+-----+
|          0 |
+-----+

```

## Character Classes

There are a number of shortcuts to match particular preset character classes. These are matched with the

[**:character\_class:**]  
pattern (inside a  
[]  
set). The following character classes exist:

Character Class	Description
alnum	Alphanumeric
alpha	Alphabetic
blank	Whitespace
cntrl	Control characters
digit	Digits
graph	Graphic characters
lower	Lowercase alphabetic
print	Graphic or space characters
punct	Punctuation
space	Space, tab, newline, and carriage return
upper	Uppercase alphabetic
xdigit	Hexadecimal digit

For example:

```
SELECT 'Maria' REGEXP 'Mar[:alnum:]*' ;
+-----+
| 'Maria' REGEXP 'Mar[:alnum:]*' |
+-----+
| 1 |
+-----+
```

Remember that matches are by default case-insensitive, unless a binary string is used, so the following example, specifically looking for an uppercase, counter-intuitively matches a lowercase character:

```
SELECT 'Mari' REGEXP 'Mar[:upper:]+' ;
+-----+
| 'Mari' REGEXP 'Mar[:upper:]+' |
+-----+
| 1 |
+-----+

SELECT BINARY 'Mari' REGEXP 'Mar[:upper:]+' ;
+-----+
| BINARY 'Mari' REGEXP 'Mar[:upper:]+' |
+-----+
| 0 |
+-----+
```

## Character Names

There are also number of shortcuts to match particular preset character names. These are matched with the

[ .character . ]  
pattern (inside a  
[ ]  
set). The following character classes exist:

Name	Character
NUL	0
SOH	001
STX	002
ETX	003
EOT	004
ENQ	005
ACK	006
BEL	007
alert	007
BS	010
backspace	'\b'
HT	011
tab	'\t'
LF	012
newline	'\n'
VT	013
vertical-tab	'\v'
FF	014
form-feed	'\f'
CR	015
carriage-return	'\r'
SO	016
SI	017
DLE	020

DC1	021
DC2	022
DC3	023
DC4	024
NAK	025
SYN	026
ETB	027
CAN	030
EM	031
SUB	032
ESC	033
IS4	034
FS	034
IS3	035
GS	035
IS2	036
RS	036
IS1	037
US	037
space	' '
exclamation-mark	'!'
quotation-mark	'''
number-sign	'#'
dollar-sign	'\$'
percent-sign	'%'
ampersand	'&'
apostrophe	'\'
left-parenthesis	'('
right-parenthesis	')'
asterisk	'*'
plus-sign	'+'
comma	' ,'
hyphen	' -'
hyphen-minus	' -'
period	' .'
full-stop	' .'
slash	' /'
solidus	' /'
zero	' 0'
one	' 1'
two	' 2'
three	' 3'
four	' 4'
five	' 5'
six	' 6'

seven	'7'
eight	'8'
nine	'9'
colon	::
semicolon	::
less-than-sign	'<'
equals-sign	'='
greater-than-sign	'>'
question-mark	'?'
commercial-at	'@'
left-square-bracket	['
backslash	'\'
reverse-solidus	'\'
right-square-bracket	']'
circumflex	'^'
circumflex-accent	'^'
underscore	'_'
low-line	'—'
grave-accent	'`'
left-brace	'{'
left-curly-bracket	'{'
vertical-line	' '
right-brace	'}'
right-curly-bracket	'}'
tilde	"~"
DEL	177

For example:

```
SELECT '|'|' REGEXP '[[.vertical-line.]]';
+-----+
| '|' REGEXP '[[.vertical-line.]]' |
+-----+
|                               1 |
+-----+
```

## Combining

The true power of regular expressions is unleashed when the above is combined, to form more complex examples. Regular expression's reputation for complexity stems from the seeming complexity of multiple combined regular expressions, when in reality, it's simply a matter of understanding the characters and how they apply:

The first example fails to match, as while the

Ma  
matches, either  
i  
or  
r  
only matches once before the  
ia  
characters at the end.

```
SELECT 'Maria' REGEXP 'Ma[ir]{2}ia';
+-----+
| 'Maria' REGEXP 'Ma[ir]{2}ia' |
+-----+
|          0 |
+-----+
```

This example matches, as either

i  
or  
r  
match exactly twice after the  
Ma  
, in this case one  
r  
and one  
i  
.

```
SELECT 'Maria' REGEXP 'Ma[ir]{2}';
+-----+
| 'Maria' REGEXP 'Ma[ir]{2}' |
+-----+
|          1 |
+-----+
```

## Escaping

With the large number of special characters, care needs to be taken to properly escape characters. Two backslash characters,

(one for the MariaDB parser, one for the regex library), are required to properly escape a character. For example:

To match the literal

(Ma  
:

```
SELECT '(Maria)' REGEXP '(Ma';
ERROR 1139 (42000): Got error 'parentheses not balanced' from regexp
```

```
SELECT '(Maria)' REGEXP '\(Ma';
ERROR 1139 (42000): Got error 'parentheses not balanced' from regexp
```

```
SELECT '(Maria)' REGEXP '\\\\(Ma';
+-----+
| '(Maria)' REGEXP '\\\\(Ma' |
+-----+
|          1 |
+-----+
```

To match

r+  
: The first two examples are incorrect, as they match  
r  
one or more times, not  
r+  
:

```
SELECT 'Mar+ia' REGEXP 'r+';
+-----+
| 'Mar+ia' REGEXP 'r+' |
+-----+
|          1 |
+-----+

SELECT 'Maria' REGEXP 'r+';
+-----+
| 'Maria' REGEXP 'r+' |
+-----+
|          1 |
+-----+

SELECT 'Maria' REGEXP 'r\\\'';
+-----+
| 'Maria' REGEXP 'r\\\'' |
+-----+
|          0 |
+-----+

SELECT 'Maria' REGEXP 'r\'+';
+-----+
| 'Maria' REGEXP 'r\'' |
+-----+
|          1 |
+-----+
```

## 1.1.12.2.1.2 Perl Compatible Regular Expressions (PCRE) Documentation

## Contents

1. PCRE Versions
2. PCRE Enhancements
3. New Regular Expression Functions
4. PCRE Syntax
  1. Special Characters
  2. Character Classes
  3. Generic Character Types
  4. Unicode Character Properties
    1. General Category Properties For \p and \P
    2. Special Category Properties For \p and \P
    3. Script Names For \p and \P
5. Extended Unicode Grapheme Sequence
6. Simple Assertions
7. Option Setting
8. Multiline Matching
9. Newline Conventions
10. Newline Sequences
11. Comments
12. Quoting
13. Resetting the Match Start
14. Non-Capturing Groups
15. Non-Greedy Quantifiers
16. Atomic Groups
17. Possessive quantifiers
18. Absolute and Relative Numeric Backreferences
19. Named Subpatterns and Backreferences
20. Positive and Negative Look-Ahead and Look-Behind Assertions
21. Subroutine Reference and Recursive Patterns
22. Defining Subpatterns For Use By Reference
23. Conditional Subpatterns
  1. Conditions With Subpattern References
  2. Other Kinds of Conditions
24. Matching Zero Bytes (0x00)
25. Other PCRE Features
26. `default_regex_flags` Examples
27. See Also

## PCRE Versions

PCRE Version	Introduced	Maturity
PCRE2 10.34	MariaDB 10.5.1	Stable
PCRE 8.43	MariaDB 10.1.39	Stable
PCRE 8.42	MariaDB 10.2.15 , MariaDB 10.1.33 , MariaDB 10.0.35	Stable
PCRE 8.41	MariaDB 10.2.8 , MariaDB 10.1.26 , MariaDB 10.0.32	Stable
PCRE 8.40	MariaDB 10.2.5 , MariaDB 10.1.22 , MariaDB 10.0.30	Stable
PCRE 8.39	MariaDB 10.1.15 , MariaDB 10.0.26	Stable
PCRE 8.38	MariaDB 10.1.10 , MariaDB 10.0.23	Stable
PCRE 8.37	MariaDB 10.1.5 , MariaDB 10.0.18	Stable
PCRE 8.36	MariaDB 10.1.2 , MariaDB 10.0.15	Stable
PCRE 8.35	MariaDB 10.1.0 , MariaDB 10.0.12	Stable
PCRE 8.34	MariaDB 10.0.8	Stable

## PCRE Enhancements

MariaDB 10.0.5 switched to the PCRE library, which significantly improved the power of the [REGEXP/RLIKE](#) operator.

The switch to PCRE added a number of features, including recursive patterns, named capture, look-ahead and look-behind assertions, non-capturing groups, non-greedy quantifiers, Unicode character properties, extended syntax for characters and character classes, multi-line matching, and many other.

Additionally, MariaDB 10.0.5 introduced three new functions that work with regular expressions: [REGEXP\\_REPLACE\(\)](#), [REGEXP\\_INSTR\(\)](#) and [REGEXP\\_SUBSTR\(\)](#).

Also, REGEXP/RLIKE, and the new functions, now work correctly with all multi-byte [character sets](#) supported by MariaDB, including East-Asian character sets (big5, gb2313, gbk, eucjp, eucjpm, cp932, ujis, euckr), and Unicode character sets (utf8, utf8mb4, ucs2, utf16, utf16le, utf32). In earlier versions of MariaDB (and all MySQL versions) REGEXP/RLIKE works correctly only with 8-bit character sets.

## New Regular Expression Functions

- [REGEXP\\_REPLACE\(subject, pattern, replace\)](#) - Replaces all occurrences of a pattern.
- [REGEXP\\_INSTR\(subject, pattern\)](#) - Position of the first appearance of a regex .
- [REGEXP\\_SUBSTR\(subject,pattern\)](#) - Returns the matching part of a string.

See the individual articles for more details and examples.

## PCRE Syntax

In most cases PCRE is backward compatible with the old POSIX 1003.2 compliant regexp library (see [Regular Expressions Overview](#) ), so you won't need to change your applications that use SQL queries with the REGEXP/RLIKE predicate.

MariaDB 10.0.11 introduced the [default\\_regex\\_flags](#) variable to address the remaining compatibilities between PCRE and the old regex library.

This section briefly describes the most important extended PCRE features. For more details please refer to the documentation on the [PCRE site](#) , or to the documentation which is bundled in the /pcre/doc/html/ directory of a MariaDB sources distribution. The pages pcresyntax.html and pcrepattern.html should be a good start. [Regular-Expressions.Info](#) is another good resource to learn about PCRE and regular expressions generally.

## Special Characters

PCRE supports the following escape sequences to match special characters:

Sequence	Description
\a	0x07 (BEL)
\cx	"control-x", where x is any ASCII character
\e	0x1B (escape)
\f	0x0C (form feed)
\n	0x0A (newline)
\r	0x0D (carriage return)
\t	0x09 (TAB)
\ddd	character with octal code ddd
\xhh	character with hex code hh
\x{hhh..}	character with hex code hhh..

Note, the backslash characters (here, and in all examples in the sections below) must be escaped with another backslash, unless you're using the [SQL\\_MODE](#) NO\_BACKSLASH\_ESCAPES

This example tests if a character has hex code 0x61:

```
SELECT 'a' RLIKE '\x{61}';  
-> 1
```

## Character Classes

PCRE supports the standard POSIX character classes such as

```
alnum  
,  
alpha  
,  
blank  
,  
cntrl
```

```
,
```

```
digit
```

```
,
```

```
graph
```

```
,
```

```
lower
```

```
,
```

```
print
```

```
,
```

```
punct
```

```
,
```

```
space
```

```
,
```

```
upper
```

```
,
```

```
xdigit
```

, with the following additional classes:

Class	Description
ascii	any ASCII character (0x00..0x7F)
word	any "word" character (a letter, a digit, or an underscore)

This example checks if the string consists of ASCII characters only:

```
SELECT 'abc' RLIKE '^[:ascii:]+$';
-> 1
```

## Generic Character Types

Generic character types complement the POSIX character classes and serve to simplify writing patterns:

Class	Description
\d	a decimal digit (same as [:digit:])
\D	a character that is not a decimal digit
\h	a horizontal white space character
\H	a character that is not a horizontal white space character
\N	a character that is not a new line
\R	a newline sequence
\s	a white space character
\S	a character that is not a white space character
\v	a vertical white space character
\V	a character that is not a vertical white space character
\w	a "word" character (same as [:word:])
\W	a "non-word" character

This example checks if the string consists of "word" characters only:

```
SELECT 'abc' RLIKE '^\\w+$';
-> 1
```

## Unicode Character Properties

```
\p{xx}
```

is a character with the

```
xx
```

property, and

```
\P{xx}
```

is a character without the

```
xx
```

property.

The property names represented by

above are limited to the Unicode script names, the general category properties, and "Any", which matches any character (including newline). Those that are not part of an identified script are lumped together as "Common".

## General Category Properties For \p and \P

Property	Description
C	Other
Cc	Control
Cf	Format
Cn	Unassigned
Co	Private use
Cs	Surrogate
L	Letter
Li	Lower case letter
Lm	Modifier letter
Lo	Other letter
Lt	Title case letter
Lu	Upper case letter
L&	Li, Lu, or Lt
M	Mark
Mc	Spacing mark
Me	Enclosing mark
Mn	Non-spacing mark
N	Number
Nd	Decimal number
NI	Letter number
No	Other number
P	Punctuation
Pc	Connector punctuation
Pd	Dash punctuation
Pe	Close punctuation
Pf	Final punctuation
Pi	Initial punctuation
Po	Other punctuation
Ps	Open punctuation
S	Symbol
Sc	Currency symbol
Sk	Modifier symbol
Sm	Mathematical symbol
So	Other symbol
Z	Separator
Zl	Line separator
Zp	Paragraph separator
Zs	Space separator

This example checks if the string consists only of characters with property N (number):

```
SELECT '1%' RLIKE '^\\p{N}+$';
-> 1
```

## Special Category Properties For \p and \P

Property	Description
Xan	Alphanumeric: union of properties L and N
Xps	POSIX space: property Z or tab, NL, VT, FF, CR
Xsp	Perl space: property Z or tab, NL, FF, CR
Xuc	A character than can be represented by a Universal Character Name
Xwd	Perl word: property Xan or underscore

The property

Xuc

matches any character that can be represented by a Universal Character Name (in C++ and other programming languages). These include

\$

,

@

,

, and all characters with Unicode code points greater than

U+00A0

, excluding the surrogates

U+D800

..

U+DFFF

## Script Names For \p and \P

Arabic, Armenian, Avestan, Balinese, Bamum, Batak, Bengali, Bopomofo, Brahmi, Braille, Buginese, Buhid, Canadian\_Aboriginal, Carian, Chakma, Cham, Cherokee, Common, Coptic, Cuneiform, Cypriot, Cyrillic, Deseret, Devanagari, Egyptian\_Hieroglyphs, Ethiopic, Georgian, Glagolitic, Gothic, Greek, Gujarati, Gurmukhi, Han, Hangul, Hanunoo, Hebrew, Hiragana, Imperial\_Aramaic, Inherited, Inscriptional\_Pahlavi, Inscriptional\_Parthian, Javanese, Kaithi, Kannada, Katakana, Kayah\_Li, Kharoshthi, Khmer, Lao, Latin, Lepcha, Limbu, Linear\_B, Lisu, Lycian, Lydian, Malayalam, Mandaic, Meetei\_Mayek, Meroitic\_Cursive, Meroitic\_Hieroglyphs, Miao, Mongolian, Myanmar, New\_Tai\_Lue, Nko, Ogham, Old\_Italic, Old\_Persian, Old\_South\_Arabian, Old\_Turkic, Ol\_Chiki, Oriya, Osmany, Phags\_Pa, Phoenician, Rejang, Runic, Samaritan, Saurashtra, Sharada, Shavian, Sinhala, Sora\_Sompeng, Sundanese, Syloti\_Nagri, Syriac, Tagalog, Tagbanwa, Tai\_Le, Tai\_Tham, Tai\_Viet, Takri, Tamil, Telugu, Thaana, Thai, Tibetan, Tifinagh, Ugaritic, Vai, Yi.

This example checks if the string consists only of Greek characters:

```
SELECT 'ΣΦΩ' RLIKE '^\\p{Greek}+$';
-> 1
```

## Extended Unicode Grapheme Sequence

The

\X

escape sequence matches a character sequence that makes an "extended grapheme cluster", i.e. a composite character that consists of multiple Unicode code points.

One of the examples of a composite character can be a letter followed by non-spacing accent marks. This example demonstrates that

U+0045 LATIN CAPITAL LETTER E

followed by

U+0302 COMBINING CIRCUMFLEX ACCENT

followed by

U+0323 COMBINING DOT BELOW

together form an extended grapheme cluster:

```
SELECT _ucs2 0x004503020323 RLIKE '^\\X$';
-> 1
```

See the [PCRE documentation](#) for the other types of extended grapheme clusters.

## Simple Assertions

An assertion specifies a certain condition that must match at a particular point, but without consuming characters from the subject string. In addition to the

standard POSIX simple assertions

^  
(that matches at the beginning of a line) and  
\$  
(that matches at the end of a line), PCRE supports a number of other assertions:

Assertion	Description
\b	matches at a word boundary
\B	matches when not at a word boundary
\A	matches at the start of the subject
\Z	matches at the end of the subject, also matches before a newline at the end of the subject
\z	matches only at the end of the subject
\G	matches at the first matching position in the subject

This example cuts a word that consists only of 3 characters from a string:

```
SELECT REGEXP_SUBSTR('---abcd---xyz---', '\b\w{3}\b');
-> xyz
```

Notice that the two

\b  
assertions checked the word boundaries but did not get into the matching pattern.

The

\b  
assertions work well in the beginning and the end of the subject string:

```
SELECT REGEXP_SUBSTR('xyz', '\b\w{3}\b');
-> xyz
```

By default, the

^  
and  
\$  
assertions have the same meaning with  
\A  
,  
\Z  
, and  
\z  
. However, the meanings of  
^  
and  
\$  
can change in multiline mode (see below). By contrast, the meanings of  
\A  
,  
\Z  
, and  
\z  
are always the same; they are independent of the multiline mode.

## Option Setting

A number of options that control the default match behavior can be changed within the pattern by a sequence of option letters enclosed between

(?  
and  
)

Option	Description
(?i)	case insensitive match
(?m)	multiline mode
(?s)	dotall mode (dot matches newline characters)

(?x)	extended (ignore white space)
(?U)	ungreedy (lazy) match
(?J)	allow named subpatterns with duplicate names
(?X)	extra PCRE functionality (e.g. force error on unknown escaped character)
(?-...)	unset option(s)

For example,

```
(?im)
sets case insensitive multiline matching.
```

A hyphen followed by the option letters unset the options. For example,

```
(?-im)
means case sensitive single line match.
```

A combined setting and unsetting is also possible, e.g.

```
(?im-sx)
```

If an option is set outside of subpattern parentheses, the option applies to the remainder of the pattern that follows the option. If an option is set inside a subpattern, it applies to the part of this subpattern that follows the option.

In this example the pattern

```
(?i)m((?-i)aria)db
matches the words
MariaDB
,
Mariadb
,
mariadb
, but not
MARIADB
:
```

```
SELECT 'MariaDB' RLIKE '(?i)m((?-i)aria)db';
-> 1

SELECT 'mariaDB' RLIKE '(?i)m((?-i)aria)db';
-> 1

SELECT 'Mariadb' RLIKE '(?i)m((?-i)aria)db';
-> 1

SELECT 'MARIADB' RLIKE '(?i)m((?-i)aria)db';
-> 0
```

This example demonstrates that the

```
(?x)
option makes the regexp engine ignore all white spaces in the pattern (other than in a class).
```

```
SELECT 'ab' RLIKE '(?x)a b';
-> 1
```

Note, putting spaces into a pattern in combination with the (?x) option can be useful to split different logical parts of a complex pattern, to make it more readable.

## Multiline Matching

Multiline matching changes the meaning of

```
^
and
$
```

from "the beginning of the subject string" and "the end of the subject string" to "the beginning of any line in the subject string" and "the end of any line in the subject string" respectively.

This example checks if the subject string contains two consequent lines that fully consist of digits:

```
SELECT 'abc\n123\n456\nxyz\n' RLIKE '(?m)^\\d+\\R\\d+$';
-> 1
```

Notice the

(?m)

option in the beginning of the pattern, which switches to the multiline matching mode.

## Newline Conventions

PCRE supports five line break conventions:

- CR (\r)  
- a single carriage return character
- LF (\n)  
- a single linefeed character
- CRLF (\r\n)  
- a carriage return followed by a linefeed
- any of the previous three
- any Unicode newline sequence

By default, the newline convention is set to any Unicode newline sequence, which includes:

Sequence	Description
LF	(U+000A, carriage return)
CR	(U+000D, carriage return)
CRLF	(a carriage return followed by a linefeed)
VT	(U+000B, vertical tab)
FF	(U+000C, form feed)
NEL	(U+0085, next line)
LS	(U+2028, line separator)
PS	(U+2029, paragraph separator)

The newline convention can be set by starting a pattern with one of the following sequences:

Sequence	Description
(*CR)	carriage return
(*LF)	linefeed
(*CRLF)	carriage return followed by linefeed
(*ANYCRLF)	any of the previous three
(*ANY)	all Unicode newline sequences

The newline conversion affects the

^

and

\$

assertions, the interpretation of the dot metacharacter, and the behavior of

\N

Note, the new line convention does not affect the meaning of

\R

This example demonstrates that the dot metacharacter matches

\n

, because it is not a newline sequence anymore:

```
SELECT 'a\nb' RLIKE '(*CR)a.b';
-> 1
```

## Newline Sequences

By default, the escape sequence

\R

matches any Unicode newline sequences.

The meaning of

\R

can be set by starting a pattern with one of the following sequences:

Sequence	Description
(*BSR_ANYCRLF)	any of CR, LF or CRLF
(*BSR_UNICODE)	any Unicode newline sequence

## Comments

It's possible to include comments inside a pattern. Comments do not participate in the pattern matching. Comments start at the

(?

# sequence and continue up to the next closing parenthesis:

```
SELECT 'ab12' RLIKE 'ab(?#expect digits)12';
-> 1
```

## Quoting

POSIX uses the backslash to remove a special meaning from a character. PCRE introduces a syntax to remove special meaning from a sequence of characters. The characters inside

\Q

...

\E

are treated literally, without their special meaning.

This example checks if the string matches a dollar sign followed by a parenthesized name (a variable reference in some languages):

```
SELECT '$(abc)' RLIKE '^\\Q$(\\E\\w+\\Q)\\E$';
-> 1
```

Note that the leftmost dollar sign and the parentheses are used literally, while the rightmost dollar sign is still used to match the end of the string.

## Resetting the Match Start

The escape sequence

\K

causes any previously matched characters to be excluded from the final matched sequence. For example, the pattern:

(foo)\Kbar

matches

foobar

, but reports that it has matched

bar

. This feature is similar to a look-behind assertion. However, in this case, the part of the subject before the real match does not have to be of fixed length:

```
SELECT REGEXP_SUBSTR('aaa123', '[a-z]*\\K[0-9]*');
-> 123
```

## Non-Capturing Groups

The question mark and the colon after the opening parenthesis create a non-capturing group:

(?:...)

This example removes an optional article from a word, for example for better sorting of the results.

```
SELECT REGEXP_REPLACE('The King', '(?:the|an|a)[^a-z]([a-z]+)', '\\1');
-> King
```

Note that the articles are listed inside the left parentheses using the alternation operator

|

but they do not produce a captured subpattern, so the word followed by the article is referenced by

'

1'

in the third argument to the function. Using non-capturing groups can be useful to save numbers on the sup-patterns that won't be used in the third argument of [REGEXP\\_REPLACE\(\)](#), as well as for performance purposes.

## Non-Greedy Quantifiers

By default, the repetition quantifiers

```
?  
,  
*  
,  
+  
and  
{n,m}
```

are "greedy", that is, they try to match as much as possible. Adding a question mark after a repetition quantifier makes it "non-greedy", so the pattern matches the minimum number of times possible.

This example cuts C comments from a line:

```
SELECT REGEXP_REPLACE('/* Comment1 */ i+= 1; /* Comment2 */', '/[*].*?[*]/', '');  
-> i+= 1;
```

The pattern without the non-greedy flag to the quantifier

```
/[*].*[*]/  
would match the entire string between the leftmost  
/*  
and the rightmost  
*/
```

## Atomic Groups

A sequence inside

```
(?>  
...  
)
```

makes an atomic group. Backtracking inside an atomic group is prevented once it has matched; however, backtracking past to the previous items works normally.

Consider the pattern

```
\d+foo  
applied to the subject string  
123bar  
. Once the engine scans  
123  
and fails on the letter  
b  
, it would normally backtrack to  
2  
and try to match again, then fail and backtrack to  
1  
and try to match and fail again, and finally fail the entire pattern. In case of an atomic group  
(?>\d+)foo  
with the same subject string  
123bar  
, the engine gives up immediately after the first failure to match  
foo  
. An atomic group with a quantifier can match all or nothing.
```

Atomic groups produce faster false results (i.e. in case when a long subject string does not match the pattern), because the regexp engine saves performance on backtracking. However, don't hurry to put everything into atomic groups. This example demonstrates the difference between atomic and non-atomic match:

```
SELECT 'abcc' RLIKE 'a(?>bc|b)c' AS atomic1;  
-> 1  
  
SELECT 'abc' RLIKE 'a(?>bc|b)c' AS atomic2;  
-> 0  
  
SELECT 'abcc' RLIKE 'a(bc|b)c' AS non_atomic1;  
-> 1  
  
SELECT 'abc' RLIKE 'a(bc|b)c' AS non_atomic2;  
-> 1
```

The non-atomic pattern matches both

```
abbc
and
abc
, while the atomic pattern matches
abbc
only.
```

The atomic group

```
(?>bc|b)
```

in the above example can be "translated" as "if there is  
bc  
, then don't try to match as  
b  
. So  
b  
can match only if  
bc  
is not found.

Atomic groups are not capturing. To make an atomic group capturing, put it into parentheses:

```
SELECT REGEXP_REPLACE('abcc','a((?>bc|b))c','\\1');
-> bc
```

## Possessive quantifiers

An atomic group which ends with a quantifier can be rewritten using a so called "possessive quantifier" syntax by putting an additional  
+  
sign following the quantifier.

The pattern

```
(?>\d+)foo
```

from the previous section's example can be rewritten as  
\d++foo

## Absolute and Relative Numeric Backreferences

Backreferences match the same text as previously matched by a capturing group. Backreferences can be written using:

- a backslash followed by a digit
- the  
    \g  
    escape sequence followed by a positive or negative number
- the  
    \g  
    escape sequence followed by a positive or negative number enclosed in braces

The following backreferences are identical and refer to the first capturing group:

- \1
- \g1
- \g{1}

This example demonstrates a pattern that matches "sense and sensibility" and "response and responsibility", but not "sense and responsibility":

```
SELECT 'sense and sensibility' RLIKE '(sens|respons)e and \\1ibility';
-> 1
```

This example removes doubled words that can unintentionally creep in when you edit a text in a text editor:

```
SELECT REGEXP_REPLACE('using using the the regexp regexp',
  '\\b(\\w+)\\s+\\1\\b','\\1');
-> using the regexp
```

Note that all double words were removed, in the beginning, in the middle and in the end of the subject string.

A negative number in a

\g

sequence means a relative reference. Relative references can be helpful in long patterns, and also in patterns that are created by joining fragments together that contain references within themselves. The sequence

\g{-1}

is a reference to the most recently started capturing subpattern before

\g

In this example

\g{-1}

is equivalent to

\2

:

```
SELECT 'abc123def123' RLIKE '(abc(123)def)\\g{-1}';  
-> 1
```

```
SELECT 'abc123def123' RLIKE '(abc(123)def)\\2';  
-> 1
```

## Named Subpatterns and Backreferences

Using numeric backreferences for capturing groups can be hard to track in a complicated regular expression. Also, the numbers can change if an expression is modified. To overcome these difficulties, PCRE supports named subpatterns.

A subpattern can be named in one of three ways:

(?<name>

...

)

or

(?'name'

...

)

as in Perl, or

(?P<name>

...

)

as in Python. References to capturing subpatterns from other parts of the pattern, can be made by name as well as by number.

Backreferences to a named subpattern can be written using the .NET syntax

\k{name}

, the Perl syntax

\k<name>

or

\k'name'

or

\g{name}

, or the Python syntax

(?P=name)

This example tests if the string is a correct HTML tag:

```
SELECT '<a href="..">Up</a>' RLIKE '<(?<tag>[a-z][a-z0-9]*[^>]*[^<]*</?P=tag>)>';  
-> 1
```

## Positive and Negative Look-Ahead and Look-Behind Assertions

Look-ahead and look-behind assertions serve to specify the context for the searched regular expression pattern. Note that the assertions only check the context, they do not capture anything themselves!

This example finds the letter which is not followed by another letter (negative look-ahead):

```
SELECT REGEXP_SUBSTR('ab1','[a-z](?! [a-z])');  
-> b
```

This example finds the letter which is followed by a digit (positive look-ahead):

```
SELECT REGEXP_SUBSTR('ab1','[a-z](?=[0-9])');  
-> b
```

This example finds the letter which does not follow a digit character (negative look-behind):

```
SELECT REGEXP_SUBSTR('1ab','(?![0-9])[a-z]');
-> b
```

This example finds the letter which follows another letter character (positive look-behind):

```
SELECT REGEXP_SUBSTR('1ab','(?<=[a-z])[a-z]');
-> b
```

Note that look-behind assertions can only be of fixed length; you cannot have repetition operators or alternations with different lengths:

```
SELECT 'aaa' RLIKE '(?<=(a|bc))a';
ERROR 1139 (42000): Got error 'lookbehind assertion is not fixed length at offset 10' from regexp
```

## Subroutine Reference and Recursive Patterns

PCRE supports a special syntax to recourse the entire pattern or its individual subpatterns:

Syntax	Description
(?R)	Recurse the entire pattern
(?n)	call subpattern by absolute number
(?+n)	call subpattern by relative number
(?-n)	call subpattern by relative number
(?&name)	call subpattern by name (Perl)
(?P>name)	call subpattern by name (Python)
\g<name>	call subpattern by name (Oniguruma)
\g'name'	call subpattern by name (Oniguruma)
\g<n>	call subpattern by absolute number (Oniguruma)
\g'n'	call subpattern by absolute number (Oniguruma)
\g<+n>	call subpattern by relative number
\g<-n>	call subpattern by relative number
\g'+n'	call subpattern by relative number
\g'-n'	call subpattern by relative number

This example checks for a correct additive arithmetic expression consisting of numbers, unary plus and minus, binary plus and minus, and parentheses:

```
SELECT '1+2-3+(+4-1)+(-2)+(+1)' RLIKE '^(([+-]?(\\d+|([(\\)?1][)]))(([+-](\\)?1))*$';
-> 1
```

The recursion is done using

(?1)

to call for the first parenthesized subpattern, which includes everything except the leading

^

and the trailing

\$

.

The regular expression in the above example implements the following BNF grammar:

1. 

```
<expression> ::= <term> [(<sign> <term>)...]
```
2. 

```
<term> ::= [ <sign> ] <primary>
```
3. 

```
<primary> ::= <number> | <left paren> <expression> <right paren>
```
4. 

```
<sign> ::= <plus sign> | <minus sign>
```

## Defining Subpatterns For Use By Reference

Use the

```
(?(DEFINE)
...
)
syntax to define subpatterns that can be referenced from elsewhere.
```

This example defines a subpattern with the name

```
letters
that matches one or more letters, which is further reused two times:
```

```
SELECT 'abc123xyz' RLIKE '^^(?<DEFINE>(?<letters>[a-z]+))(&letters)[0-9]+(&letters)$';
-> 1
```

The above example can also be rewritten to define the digit part as a subpattern as well:

```
SELECT 'abc123xyz' RLIKE
'^^(?<DEFINE>(?<letters>[a-z]+)(?<digits>[0-9]+))(&letters)(&digits)(&letters)$';
-> 1
```

## Conditional Subpatterns

There are two forms of conditional subpatterns:

```
(?(condition)yes-pattern)
(?(condition)yes-pattern|no-pattern)
```

The

```
yes-pattern
is used if the condition is satisfied, otherwise the
no-pattern
(if any) is used.
```

## Conditions With Subpattern References

If a condition consists of a number, it makes a condition with a subpattern reference. Such a condition is true if a capturing subpattern corresponding to the number has previously matched.

This example finds an optionally parenthesized number in a string:

```
SELECT REGEXP_SUBSTR('a(123)b', '([()]?[0-9]+(?:1)[)])');
-> (123)
```

The

```
([()])?
part makes a capturing subpattern that matches an optional opening parenthesis; the
[0-9]+
part matches a number, and the
(?:1)[)]
part matches a closing parenthesis, but only if the opening parenthesis has been previously found.
```

## Other Kinds of Conditions

The other possible condition kinds are: recursion references and assertions. See the [PCRE documentation](#) for details.

## Matching Zero Bytes (0x00)

PCRE correctly works with zero bytes in the subject strings:

```
SELECT 'a\0b' RLIKE '^a.b$';
-> 1
```

Zero bytes, however, are not supported literally in the pattern strings and should be escaped using the

```
\xhh
or
\x{hh}
syntax:
```

```
SELECT 'a\0b' RLIKE '^a\\x{00}b$';
-> 1
```

## Other PCRE Features

PCRE provides other extended features that were not covered in this document, such as duplicate subpattern numbers, backtracking control, breaking utf-8 sequences into individual bytes, setting the match limit, setting the recursion limit, optimization control, recursion conditions, assertion conditions and more types of extended grapheme clusters. Please refer to the [PCRE documentation](#) for details.

Enhanced regex was implemented as a GSoC 2013 project by Sudheera Palihakkara.

## default\_regex\_flags Examples

MariaDB starting with 10.0.11

The `default_regex_flags` variable was introduced in **MariaDB 10.0.11**

The `default_regex_flags` variable was introduced to address the remaining incompatibilities between PCRE and the old regex library. Here are some examples of its usage:

The default behaviour (multiline match is off)

```
SELECT 'a\nb\n' RLIKE '^b$';
+-----+
| '(?m)a\nb\n' RLIKE '^b$' |
+-----+
|          0 |
+-----+
```

Enabling the multiline option using the PCRE option syntax:

```
SELECT 'a\nb\n' RLIKE '(?m)^b$';
+-----+
| 'a\nb\n' RLIKE '(?m)^b$' |
+-----+
|          1 |
+-----+
```

Enabling the multiline option using `default_regex_flags`

```
SET default_regex_flags='MULTILINE';
SELECT 'a\nb\n' RLIKE '^b$';
+-----+
| 'a\nb\n' RLIKE '^b$' |
+-----+
|          1 |
+-----+
```

## See Also

- [MariaDB upgrades to PCRE-8.34](#)

## 1.1.12.2.1.3 NOT REGEXP

### Syntax

```
expr NOT REGEXP pat, expr NOT RLIKE pat
```

### Description

This is the same as [NOT \(expr REGEXP pat\)](#).

## 1.1.12.2.1.4 REGEXP

### Syntax

```
expr REGEXP pat, expr RLIKE pat
```

## Description

Performs a pattern match of a string expression

```
expr  
against a pattern  
pat
```

. The pattern can be an extended regular expression. See [Regular Expressions Overview](#) for details on the syntax for regular expressions (see also [PCRE Regular Expressions](#) ).

Returns

```
1  
if  
expr  
matches  
pat  
or  
0  
if it doesn't match. If either  
expr  
or  
pat  
are NULL, the result is NULL.
```

The negative form [NOT REGEXP](#) also exists, as an alias for

```
NOT (string REGEXP pattern)  
. RLIKE and NOT RLIKE are synonyms for REGEXP and NOT REGEXP, originally provided for mSQL compatibility.
```

The pattern need not be a literal string. For example, it can be specified as a string expression or table column.

**Note:** Because MariaDB uses the C escape syntax in strings (for example, "\n" to represent the newline character), you must double any "\" that you use in your REGEXP strings.

REGEXP is not case sensitive, except when used with binary strings.

[MariaDB 10.0.5](#) moved to the PCRE regex library - see [PCRE Regular Expressions](#) for enhancements to REGEXP introduced in [MariaDB 10.0.5](#) .

The [default\\_regex\\_flags](#) variable addresses the remaining compatibilities between PCRE and the old regex library.

## Examples

```

SELECT 'Monty!' REGEXP 'm%y%%';
+-----+
| 'Monty!' REGEXP 'm%y%%' |
+-----+
|          0 |
+-----+

SELECT 'Monty!' REGEXP '.';
+-----+
| 'Monty!' REGEXP '.' |
+-----+
|          1 |
+-----+

SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';
+-----+
| 'new*\n*line' REGEXP 'new\\*.\\*line' |
+-----+
|          1 |
+-----+

SELECT 'a' REGEXP 'A', 'a' REGEXP BINARY 'A';
+-----+
| 'a' REGEXP 'A' | 'a' REGEXP BINARY 'A' |
+-----+
|          1 |          0 |
+-----+

SELECT 'a' REGEXP '^[a-d]';
+-----+
| 'a' REGEXP '^[a-d]' |
+-----+
|          1 |
+-----+

```

## default\_regex\_flags examples

MariaDB 10.0.11 introduced the [default\\_regex\\_flags](#) variable to address the remaining compatibilities between PCRE and the old regex library.

The default behaviour (multiline match is off)

```

SELECT 'a\nb\nc' RLIKE '^b$';
+-----+
| '(?m)a\nb\nc' RLIKE '^b$' |
+-----+
|          0 |
+-----+

```

Enabling the multiline option using the PCRE option syntax:

```

SELECT 'a\nb\nc' RLIKE '(?m)^b$';
+-----+
| 'a\nb\nc' RLIKE '(?m)^b$' |
+-----+
|          1 |
+-----+

```

Enabling the multiline option using default\_regex\_flags

```

SET default_regex_flags='MULTILINE';
SELECT 'a\nb\nc' RLIKE '^b$';
+-----+
| 'a\nb\nc' RLIKE '^b$' |
+-----+
|          1 |
+-----+

```

## 1.1.12.2.1.5 REGEXP\_INSTR

### Syntax

```
REGEXP_INSTR(subject, pattern)
```

Returns the position of the first occurrence of the regular expression

pattern  
in the string  
subject  
, or 0 if pattern was not found.

The positions start with 1 and are measured in characters (i.e. not in bytes), which is important for multi-byte character sets. You can cast a multi-byte character set to [BINARY](#) to get offsets in bytes.

The function follows the case sensitivity rules of the effective [collation](#). Matching is performed case insensitively for case insensitive collations, and case sensitively for case sensitive collations and for binary data.

The collation case sensitivity can be overwritten using the (?i) and (?-i) PCRE flags.

MariaDB uses the [PCRE regular expression](#) library for enhanced regular expression performance, and REGEXP\_INSTR was introduced as part of this enhancement.

## Examples

```
SELECT REGEXP_INSTR('abc','b');  
-> 2  
  
SELECT REGEXP_INSTR('abc','x');  
-> 0  
  
SELECT REGEXP_INSTR('BJÖRN','N');  
-> 5
```

Casting a multi-byte character set as BINARY to get offsets in bytes:

```
SELECT REGEXP_INSTR(BINARY 'BJÖRN','N') AS cast_utf8_to_binary;  
-> 6
```

Case sensitivity:

```
SELECT REGEXP_INSTR('ABC','b');  
-> 2  
  
SELECT REGEXP_INSTR('ABC' COLLATE utf8_bin,'b');  
-> 0  
  
SELECT REGEXP_INSTR(BINARY'ABC','b');  
-> 0  
  
SELECT REGEXP_INSTR('ABC','(?-i)b');  
-> 0  
  
SELECT REGEXP_INSTR('ABC' COLLATE utf8_bin,'(?i)b');  
-> 2
```

## 1.1.12.2.1.6 REGEXP\_REPLACE

### Syntax

```
REGEXP_REPLACE(subject, pattern, replace)
```

### Description

REGEXP\_REPLACE  
returns the string  
subject  
with all occurrences of the regular expression  
pattern  
replaced by the string  
replace

. If no occurrences are found, then  
subject  
is returned as is.

The replace string can have backreferences to the subexpressions in the form \N, where N is a number from 1 to 9.

The function follows the case sensitivity rules of the effective [collation](#). Matching is performed case insensitively for case insensitive collations, and case sensitively for case sensitive collations and for binary data.

The collation case sensitivity can be overwritten using the (?i) and (?-i) PCRE flags.

MariaDB uses the [PCRE regular expression](#) library for enhanced regular expression performance, and  
REGEXP\_REPLACE  
was introduced as part of this enhancement.

The [default\\_regex\\_flags](#) variable addresses the remaining compatibilities between PCRE and the old regex library.

## Examples

```
SELECT REGEXP_REPLACE('ab12cd','[0-9]', '') AS remove_digits;  
-> abcd  
  
SELECT REGEXP_REPLACE('<html><head><title>title</title><body>body</body></html>', '<.+?>', ' ')  
AS strip_html;  
-> title body
```

Backreferences to the subexpressions in the form

\N  
, where N is a number from 1 to 9:

```
SELECT REGEXP_REPLACE('James Bond','^(.*) (.*)$', '\2, \1') AS reorder_name;  
-> Bond, James
```

Case insensitive and case sensitive matches:

```
SELECT REGEXP_REPLACE('ABC','b','-') AS case_insensitive;  
-> A-C  
  
SELECT REGEXP_REPLACE('ABC' COLLATE utf8_bin,'b','-') AS case_sensitive;  
-> ABC  
  
SELECT REGEXP_REPLACE(BINARY 'ABC','b','-') AS binary_data;  
-> ABC
```

Overwriting the collation case sensitivity using the (?i) and (?-i) PCRE flags.

```
SELECT REGEXP_REPLACE('ABC','(?-i)b','-') AS force_case_sensitive;  
-> ABC  
  
SELECT REGEXP_REPLACE(BINARY 'ABC','(?i)b','-') AS force_case_insensitive;  
-> A-C
```

## 1.1.12.2.1.7 REGEXP\_SUBSTR

### Syntax

```
REGEXP_SUBSTR(subject,pattern)
```

### Description

Returns the part of the string

subject  
that matches the regular expression  
pattern  
, or an empty string if  
pattern  
was not found.

The function follows the case sensitivity rules of the effective [collation](#). Matching is performed case insensitively for case insensitive collations, and case

sensitively for case sensitive collations and for binary data.

The collation case sensitivity can be overwritten using the (?i) and (?-i) PCRE flags.

MariaDB uses the [PCRE regular expression](#) library for enhanced regular expression performance, and `REGEXP_SUBSTR` was introduced as part of this enhancement.

The `default_regex_flags` variable addresses the remaining compatibilities between PCRE and the old regex library.

## Examples

```
SELECT REGEXP_SUBSTR('ab12cd','[0-9]+');
-> 12

SELECT REGEXP_SUBSTR(
  'See https://mariadb.org/en/foundation/ for details',
  'https?://[^/]*');
-> https://mariadb.org
```

```
SELECT REGEXP_SUBSTR('ABC','b');
-> B

SELECT REGEXP_SUBSTR('ABC' COLLATE utf8_bin,'b');
->

SELECT REGEXP_SUBSTR(BINARY'ABC','b');
->

SELECT REGEXP_SUBSTR('ABC','(?i)b');
-> B

SELECT REGEXP_SUBSTR('ABC' COLLATE utf8_bin,'(?+i)b');
-> B
```

## 1.1.12.2.1.8 RLIKE

### Syntax

```
expr REGEXP pat, expr RLIKE pat
```

### Description

`RLIKE`  
is a synonym for [REGEXP](#).

## 1.1.12.2.2 Dynamic Columns Functions

### 1.1.12.2.2.1 COLUMN\_ADD

#### Syntax

```
COLUMN_ADD(dyncol_blob, column_nr, value [as type], [column_nr, value [as type]]...);
COLUMN_ADD(dyncol_blob, column_name, value [as type], [column_name, value [as type]]...);
```

### Description

Adds or updates [dynamic columns](#).

- - `dyncol_blob`  
must be either a valid dynamic columns blob (for example, `COLUMN_CREATE` returns such blob), or an empty string.

- **column\_name**  
specifies the name of the column to be added. If **dyncol\_blob** already has a column with this name, it will be overwritten.
- **value**  
specifies the new value for the column. Passing a NULL value will cause the column to be deleted.
- **as type**  
is optional. See [#datatypes](#) section for a discussion about types.

The return value is a dynamic column blob after the modifications.

## Examples

```
UPDATE t1 SET dyncol_blob=COLUMN_ADD(dyncol_blob, "column_name", "value") WHERE id=1;
```

Note:

`COLUMN_ADD()`

is a regular function (just like

`CONCAT()`

), hence, in order to update the value in the table you have to use the

```
UPDATE ... SET dynamic_col=COLUMN_ADD(dynamic_col,
....)
pattern.
```

## 1.1.12.2.2.2 COLUMN\_CHECK

### Syntax

```
COLUMN_CHECK(dyncol_blob);
```

### Description

Check if

`dyncol_blob`

is a valid packed dynamic columns blob. Return value of 1 means the blob is valid, return value of 0 means it is not.

**Rationale:** Normally, one works with valid dynamic column blobs. Functions like `COLUMN_CREATE`, `COLUMN_ADD`, `COLUMN_DELETE` always return valid dynamic column blobs. However, if a dynamic column blob is accidentally truncated, or transcoded from one character set to another, it will be corrupted. This function can be used to check if a value in a blob field is a valid dynamic column blob.

## 1.1.12.2.2.3 COLUMN\_CREATE

### Syntax

```
COLUMN_CREATE(column_nr, value [as type], [column_nr, value [as type]]...);
COLUMN_CREATE(column_name, value [as type], [column_name, value [as type]]...);
```

### Description

Returns a [dynamic columns](#) blob that stores the specified columns with values.

The return value is suitable for

- storing in a table
- further modification with other dynamic columns functions

The

`as type`

part allows one to specify the value type. In most cases, this is redundant because MariaDB will be able to deduce the type of the value. Explicit type specification may be needed when the type of the value is not apparent. For example, a literal

`'2012-12-01'`

has a CHAR type by default, one will need to specify

'2012-12-01' AS DATE  
to have it stored as a date. See [Dynamic Columns:Datatypes](#) for further details.

## Examples

```
INSERT INTO tbl SET dyncol_blob=COLUMN_CREATE("column_name", "value");
```

### 1.1.12.2.2.4 COLUMN\_DELETE

#### Syntax

```
COLUMN_DELETE(dyncol_blob, column_nr, column_nr...);  
COLUMN_DELETE(dyncol_blob, column_name, column_name...);
```

#### Description

Deletes a [dynamic column](#) with the specified name. Multiple names can be given. The return value is a dynamic column blob after the modification.

### 1.1.12.2.2.5 COLUMN\_EXISTS

#### Syntax

```
COLUMN_EXISTS(dyncol_blob, column_nr);  
COLUMN_EXISTS(dyncol_blob, column_name);
```

#### Description

Checks if a column with name

column\_name  
exists in  
dyncol\_blob  
. If yes, return  
1  
, otherwise return  
0  
. See [dynamic columns](#) for more information.

### 1.1.12.2.2.6 COLUMN\_GET

#### Syntax

```
COLUMN_GET(dyncol_blob, column_nr as type);  
COLUMN_GET(dyncol_blob, column_name as type);
```

#### Description

Gets the value of a [dynamic column](#) by its name. If no column with the given name exists,

NULL  
will be returned.

column\_name as type  
requires that one specify the datatype of the dynamic column they are reading.

This may seem counter-intuitive: why would one need to specify which datatype they're retrieving? Can't the dynamic columns system figure the datatype from the data being stored?

The answer is: SQL is a statically-typed language. The SQL interpreter needs to know the datatypes of all expressions before the query is run (for example, when one is using prepared statements and runs

"select COLUMN\_GET(...)"  
, the prepared statement API requires the server to inform the client about the datatype of the column being read before the query is executed and the server can see what datatype the column actually has).

## Lengths

If you're running queries like:

```
SELECT COLUMN_GET(blob, 'colname' as CHAR) ...
```

without specifying a maximum length (i.e. using

as CHAR

, not

as CHAR(n)

), MariaDB will report the maximum length of the resultset column to be 16,777,216. This may cause excessive memory usage in some client libraries, because they try to pre-allocate a buffer of maximum resultset width. To avoid this problem, use CHAR(n) whenever you're using COLUMN\_GET in the select list.

See [Dynamic Columns:Datatypes](#) for more information about datatypes.

## 1.1.12.2.2.7 COLUMN\_JSON

### Syntax

```
COLUMN_JSON(dyncol_blob)
```

### Description

Returns a JSON representation of data in

dyncol\_blob

. Can also be used to display nested columns. See [dynamic columns](#) for more information.

### Example

```
select item_name, COLUMN_JSON(dynamic_cols) from assets;
+-----+-----+
| item_name      | COLUMN_JSON(dynamic_cols)          |
+-----+-----+
| MariaDB T-shirt | {"size":"XL","color":"blue"}        |
| Thinkpad Laptop | {"color":"black","warranty":"3 years"} |
+-----+-----+
```

Limitation:

COLUMN\_JSON

will decode nested dynamic columns at a nesting level of not more than 10 levels deep. Dynamic columns that are nested deeper than 10 levels will be shown as BINARY string, without encoding.

## 1.1.12.2.3 ASCII

### Syntax

```
ASCII(str)
```

### Description

Returns the numeric ASCII value of the leftmost character of the string argument. Returns

0

if the given string is empty and

NULL

if it is

NULL

.

ASCII()

works for 8-bit characters.

### Examples

```
SELECT ASCII(9);
+-----+
| ASCII(9) |
+-----+
|      57   |
+-----+

SELECT ASCII('9');
+-----+
| ASCII('9') |
+-----+
|      57   |
+-----+

SELECT ASCII('abc');
+-----+
| ASCII('abc') |
+-----+
|      97   |
+-----+
```

## 1.1.12.2.4 BIN

### Syntax

```
BIN(N)
```

### Description

Returns a string representation of the binary value of the given longlong (that is,

`BIGINT`

) number. This is equivalent to

`CONV(N,10,2)`

. The argument should be positive. If it is a

`FLOAT`

, it will be truncated. Returns

`NULL`

if the argument is

`NULL`

.

### Examples

```
SELECT BIN(12);
+-----+
| BIN(12) |
+-----+
| 1100     |
+-----+
```

### See Also

- [Binary literals](#)
- [CONV\(\)](#)
- [OCT\(\)](#)
- [HEX\(\)](#)

## 1.1.12.2.5 BINARY Operator

This page describes the BINARY operator. For details about the data type, see [Binary Data Type](#).

# Syntax

BINARY

## Description

The

BINARY

operator casts the string following it to a binary string. This is an easy way to force a column comparison to be done byte by byte rather than character by character. This causes the comparison to be case sensitive even if the column isn't defined as

BINARY

or

BLOB

BINARY

also causes trailing spaces to be significant.

## Examples

```
SELECT 'a' = 'A';
+-----+
| 'a' = 'A' |
+-----+
|      1 |
+-----+

SELECT BINARY 'a' = 'A';
+-----+
| BINARY 'a' = 'A' |
+-----+
|      0 |
+-----+

SELECT 'a' = 'a ';
+-----+
| 'a' = 'a ' |
+-----+
|      1 |
+-----+

SELECT BINARY 'a' = 'a ';
+-----+
| BINARY 'a' = 'a ' |
+-----+
|      0 |
+-----+
```

## 1.1.12.2.6 BIT\_LENGTH

### Syntax

BIT\_LENGTH(str)

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [Compatibility](#)

## Description

Returns the length of the given string argument in bits. If the argument is not a string, it will be converted to string. If the argument is

NULL  
, it returns  
NULL

## Examples

```
SELECT BIT_LENGTH('text');
+-----+
| BIT_LENGTH('text') |
+-----+
|          32 |
+-----+
```

```
SELECT BIT_LENGTH('');
+-----+
| BIT_LENGTH('') |
+-----+
|          0 |
+-----+
```

## Compatibility

PostgreSQL and Sybase support `BIT_LENGTH()`.

### 1.1.12.2.7 CAST

#### Syntax

```
CAST(expr AS type)
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

#### Description

The

`CAST()`

function takes a value of one `type` and produces a value of another type, similar to the `CONVERT()` function.

The type can be one of the following values:

- `BINARY`
- `CHAR`
- `DATE`
- `DATETIME`
- `DECIMAL[(M,D)]`
- `DOUBLE`
- `FLOAT` (from [MariaDB 10.4.5](#) )
- `INTEGER`
  - Short for  
`SIGNED INTEGER`
  - `SIGNED [INTEGER]`
  - `UNSIGNED [INTEGER]`
  - `TIME`
  - `VARCHAR` (in [Oracle mode](#) , from [MariaDB 10.3](#) )

The main difference between

`CAST`

and `CONVERT()` is that

`CONVERT(expr,type)`

is ODBC syntax while

`CAST(expr as type)`

and

`CONVERT(... USING ...)`

are SQL92 syntax.

In MariaDB 10.4 and later, you can use the

`CAST()`  
function with the  
`INTERVAL`  
keyword.

Until MariaDB 5.5.31 ,

`X'HHHH'`  
, the standard SQL syntax for binary string literals, erroneously worked in the same way as  
`0xHHHH`  
. In 5.5.31 it was intentionally changed to behave as a string in all contexts (and never as a number).

This introduced an incompatibility with previous versions of MariaDB, and all versions of MySQL (see the example below).

## Examples

Simple casts:

```
SELECT CAST("abc" AS BINARY);
SELECT CAST("1" AS UNSIGNED INTEGER);
SELECT CAST(123 AS CHAR CHARACTER SET utf8)
```

Note that when one casts to `CHAR` without specifying the character set, the `collation_connection` character set collation will be used. When used with `CHAR CHARACTER SET`, the default collation for that character set will be used.

```
SELECT COLLATION(CAST(123 AS CHAR));
+-----+
| COLLATION(CAST(123 AS CHAR)) |
+-----+
| latin1_swedish_ci           |
+-----+

SELECT COLLATION(CAST(123 AS CHAR CHARACTER SET utf8));
+-----+
| COLLATION(CAST(123 AS CHAR CHARACTER SET utf8)) |
+-----+
| utf8_general_ci             |
+-----+
```

If you also want to change the collation, you have to use the

`COLLATE`

operator:

```
SELECT COLLATION(CAST(123 AS CHAR CHARACTER SET utf8)
    COLLATE utf8_unicode_ci);
+-----+
| COLLATION(CAST(123 AS CHAR CHARACTER SET utf8) COLLATE utf8_unicode_ci) |
+-----+
| utf8_unicode_ci               |
+-----+
```

Using

`CAST()`

to order an

`ENUM`

field as a

`CHAR`

rather than the internal numerical value:

```

CREATE TABLE enum_list (enum_field enum('c','a','b'));

INSERT INTO enum_list (enum_field)
VALUES('c'),('a'),('c'),('b');

SELECT * FROM enum_list
ORDER BY enum_field;
+-----+
| enum_field |
+-----+
| c          |
| c          |
| a          |
| b          |
+-----+

SELECT * FROM enum_list
ORDER BY CAST(enum_field AS CHAR);
+-----+
| enum_field |
+-----+
| a          |
| b          |
| c          |
| c          |
+-----+

```

From [MariaDB 5.5.31](#), the following will trigger warnings, since

x'aa'  
and  
'X'aa'

no longer behave as a number. Previously, and in all versions of MySQL, no warnings are triggered since they did erroneously behave as a number:

```

SELECT CAST(0xAA AS UNSIGNED), CAST(x'aa' AS UNSIGNED), CAST(X'aa' AS UNSIGNED);
+-----+-----+-----+
| CAST(0xAA AS UNSIGNED) | CAST(x'aa' AS UNSIGNED) | CAST(X'aa' AS UNSIGNED) |
+-----+-----+-----+
|          170 |             0 |            0 |
+-----+-----+-----+
1 row in set, 2 warnings (0.00 sec)

Warning (Code 1292): Truncated incorrect INTEGER value: '\xAA'
Warning (Code 1292): Truncated incorrect INTEGER value: '\xAA'

```

Casting to intervals:

```

SELECT CAST('2019-01-04 INTERVAL AS DAY_SECOND(2)' AS "Cast";

+-----+
| Cast      |
+-----+
| 00:20:17.00 |
+-----+

```

## See Also

- [Supported data types](#)
- [Microseconds in MariaDB](#)
- [String literals](#)
- [COLLATION\(\)](#)
- [CONVERT\(\)](#)

## 1.1.12.2.8 CHAR Function

### Syntax

```
CHAR(N,... [USING charset_name])
```

# Description

CHAR()  
interprets each argument as an

INT

and returns a string consisting of the characters given by the code values of those integers.

NULL  
values are skipped. By default,  
CHAR()  
returns a binary string. To produce a string in a given character set , use the optional  
USING  
clause:

```
SELECT CHARSET(CHAR(0x65)), CHARSET(CHAR(0x65 USING utf8));  
+-----+  
| CHARSET(CHAR(0x65)) | CHARSET(CHAR(0x65 USING utf8)) |  
+-----+  
| binary           | utf8                |  
+-----+
```

If

USING  
is given and the result string is illegal for the given character set, a warning is issued. Also, if strict SQL mode is enabled, the result from  
CHAR()  
becomes  
NULL

## Examples

```
SELECT CHAR(77,97,114,'105',97,'68',66);  
+-----+  
| CHAR(77,97,114,'105',97,'68',66) |  
+-----+  
| MariaDB          |  
+-----+  
  
SELECT CHAR(77,77.3,'77.3');  
+-----+  
| CHAR(77,77.3,'77.3') |  
+-----+  
| MMM              |  
+-----+  
1 row in set, 1 warning (0.00 sec)  
  
Warning (Code 1292): Truncated incorrect INTEGER value: '77.3'
```

## See Also

- [Character Sets and Collations](#)
- [ASCII\(\)](#) - Return ASCII value of first character
- [ORD\(\)](#) - Return value for character in single or multi-byte character sets
- [CHR](#) - Similar, Oracle-compatible, function

## 1.1.12.2.9 CHAR\_LENGTH

### Syntax

```
CHAR_LENGTH(str)  
CHARACTER_LENGTH(str)
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Returns the length of the given string argument, measured in characters. A multi-byte character counts as a single character. This means that for a string containing five two-byte characters, `LENGTH()` (or `OCTET_LENGTH()` in Oracle mode) returns 10, whereas

```
CHAR_LENGTH()  
returns 5. If the argument is  
NULL  
, it returns  
NULL
```

If the argument is not a string value, it is converted into a string.

It is synonymous with the

```
CHARACTER_LENGTH()  
function.
```

## Examples

```
SELECT CHAR_LENGTH('MariaDB');  
+-----+  
| CHAR_LENGTH('MariaDB') |  
+-----+  
| 7 |  
+-----+
```

When Oracle mode from MariaDB 10.3 is not set:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');  
+-----+-----+-----+-----+  
| CHAR_LENGTH('π') | LENGTH('π') | LENGTHB('π') | OCTET_LENGTH('π') |  
+-----+-----+-----+-----+  
| 1 | 2 | 2 | 2 |  
+-----+-----+-----+-----+
```

In Oracle mode from MariaDB 10.3 :

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');  
+-----+-----+-----+-----+  
| CHAR_LENGTH('π') | LENGTH('π') | LENGTHB('π') | OCTET_LENGTH('π') |  
+-----+-----+-----+-----+  
| 1 | 1 | 2 | 2 |  
+-----+-----+-----+-----+
```

## See Also

- [LENGTH\(\)](#)
- [LENGTHB\(\)](#)
- [OCTET\\_LENGTH\(\)](#)
- [Oracle mode from MariaDB 10.3](#)

## 1.1.12.2.10 CHARACTER\_LENGTH

### Syntax

```
CHARACTER_LENGTH(str)
```

### Description

`CHARACTER_LENGTH()`  
is a synonym for

`CHAR_LENGTH()`

## 1.1.12.2.11 CHR

MariaDB starting with [10.3.1](#)

The

`CHR()`  
function was introduced in [MariaDB 10.3.1](#) to provide Oracle compatibility

### Syntax

`CHR(N)`

### Description

`CHR()`  
interprets each argument N as an integer and returns a

`VARCHAR(1)`

string consisting of the character given by the code values of the integer. The character set and collation of the string are set according to the values of the

`character_set_database`

and

`collation_database`

system variables.

`CHR()`  
is similar to the

`CHAR()`

function, but only accepts a single argument.

`CHR()`  
is available in all `sql_modes`.

### Examples

```

SELECT CHR(67);
+-----+
| CHR(67) |
+-----+
| C       |
+-----+

SELECT CHR('67');
+-----+
| CHR('67') |
+-----+
| C          |
+-----+

SELECT CHR('C');
+-----+
| CHR('C') |
+-----+
|          |
+-----+
1 row in set, 1 warning (0.000 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message           |
+-----+-----+
| Warning | 1292 | Truncated incorrect INTEGER value: 'C' |
+-----+-----+

```

## See Also

- [Character Sets and Collations](#)
- [ASCII\(\)](#) - Return ASCII value of first character
- [ORD\(\)](#) - Return value for character in single or multi-byte character sets
- [CHAR\(\)](#) - Similar function which accepts multiple integers

## 1.1.12.2.12 CONCAT

### Syntax

```
CONCAT(str1,str2,...)
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [Oracle Mode](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the string that results from concatenating the arguments. May have one or more arguments. If all arguments are non-binary strings, the result is a non-binary string. If the arguments include any binary strings, the result is a binary string. A numeric argument is converted to its equivalent binary string form; if you want to avoid that, you can use an explicit type cast, as in this example:

```
SELECT CONCAT(CAST(int_col AS CHAR), char_col);
```

```

CONCAT()
returns
NULL
if any argument is
NULL
.
```

A

NULL  
parameter hides all information contained in other parameters from the result. Sometimes this is not desirable; to avoid this, you can:

- Use the

[CONCAT\\_WS\(\)](#)

function with an empty separator, because that function is  
NULL  
-safe.

- Use

[IFNULL\(\)](#)

to turn NULLs into empty strings.

## Oracle Mode

MariaDB starting with [10.3](#)

In [Oracle mode from MariaDB 10.3](#),

CONCAT  
ignores **NULL**.

## Examples

```
SELECT CONCAT('Ma', 'ria', 'DB');
+-----+
| CONCAT('Ma', 'ria', 'DB') |
+-----+
| MariaDB |
+-----+

SELECT CONCAT('Ma', 'ria', NULL, 'DB');
+-----+
| CONCAT('Ma', 'ria', NULL, 'DB') |
+-----+
| NULL |
+-----+

SELECT CONCAT(42.0);
+-----+
| CONCAT(42.0) |
+-----+
| 42.0 |
+-----+
```

Using

[IFNULL\(\)](#)

to handle NULLs:

```
SELECT CONCAT('The value of @v is: ', IFNULL(@v, ''));
+-----+
| CONCAT('The value of @v is: ', IFNULL(@v, '')) |
+-----+
| The value of @v is: |
+-----+
```

In [Oracle mode](#), from [MariaDB 10.3](#):

```
SELECT CONCAT('Ma', 'ria', NULL, 'DB');
+-----+
| CONCAT('Ma', 'ria', NULL, 'DB') |
+-----+
| MariaDB |
+-----+
```

## See Also

- [GROUP\\_CONCAT\(\)](#)
- [Oracle mode from MariaDB 10.3](#)

## 1.1.12.2.13 CONCAT\_WS

### Syntax

```
CONCAT_WS(separator,str1,str2,...)
```

### Description

CONCAT\_WS()

stands for Concatenate With Separator and is a special form of

[CONCAT\(\)](#)

. The first argument is the separator for the rest of the arguments. The separator is added between the strings to be concatenated. The separator can be a string, as can the rest of the arguments.

If the separator is

NULL

, the result is

NULL

; all other

NULL

values are skipped. This makes

CONCAT\_WS()

suitable when you want to concatenate some values and avoid losing all information if one of them is

NULL

### Examples

```
SELECT CONCAT_WS(',','First name','Second name','Last Name');
+-----+
| CONCAT_WS(',',$First name','Second name','Last Name') |
+-----+
| First name,Second name,Last Name |
```

```
SELECT CONCAT_WS('-', 'Floor', NULL, 'Room');
+-----+
| CONCAT_WS('-', 'Floor', NULL, 'Room') |
+-----+
| Floor-Room |
```

In some cases, remember to include a space in the separator string:

```
SET @a = 'gnu', @b = 'penguin', @c = 'sea lion';
Query OK, 0 rows affected (0.00 sec)
```

```
SELECT CONCAT_WS(' ', @a, @b, @c);
+-----+
| CONCAT_WS(' ', @a, @b, @c) |
+-----+
| gnu, penguin, sea lion |
```

Using

CONCAT\_WS()

to handle

NULL

s:

```

SET @a = 'a', @b = NULL, @c = 'c';
SELECT CONCAT_WS('', @a, @b, @c);
+-----+
| CONCAT_WS('', @a, @b, @c) |
+-----+
| ac                                |
+-----+

```

## See Also

- [GROUP\\_CONCAT\(\)](#)

## 1.1.12.2.14 CONVERT

### Syntax

```
CONVERT(expr,type), CONVERT(expr USING transcoding_name)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

`CONVERT()`  
and `CAST()` functions take a value of one type and produce a value of another type.

The type can be one of the following values:

- [BINARY](#)
- [CHAR](#)
- [DATE](#)
- [DATETIME](#)
- [DECIMAL\[\(M,D\)\]](#)
- [DOUBLE](#)
- [FLOAT \(from MariaDB 10.4.5 \)](#)
- [INTEGER](#)
  - Short for  
`SIGNED INTEGER`
- [SIGNED \[INTEGER\]](#)
- [UNSIGNED \[INTEGER\]](#)
- [TIME](#)
- [VARCHAR \(in Oracle mode , from MariaDB 10.3 \)](#)

Note that in MariaDB,

`INT`  
and  
`INTEGER`  
are the same thing.

`BINARY`  
produces a string with the `BINARY` data type. If the optional length is given,  
`BINARY(N)`  
causes the cast to use no more than  
`N`  
bytes of the argument. Values shorter than the given number in bytes are padded with 0x00 bytes to make them equal the length value.

`CHAR(N)`  
causes the cast to use no more than the number of characters given in the argument.

The main difference between the `CAST()` and

`CONVERT()`  
is that

`CONVERT(expr,type)`  
is ODBC syntax while `CAST(expr as type)` and  
`CONVERT(... USING ...)`  
are SQL92 syntax.

`CONVERT()`  
with  
`USING`

is used to convert data between different [character sets](#). In MariaDB, transcoding names are the same as the corresponding character set names. For example, this statement converts the string 'abc' in the default character set to the corresponding string in the utf8 character set:

```
SELECT CONVERT('abc' USING utf8);
```

## Examples

```
SELECT enum_col FROM tbl_name
ORDER BY CAST(enum_col AS CHAR);
```

Converting a [BINARY](#) to string to permit the [LOWER](#) function to work:

```
SET @x = 'AardVark';
SET @x = BINARY 'AardVark';

SELECT LOWER(@x), LOWER(CONVERT (@x USING latin1));
+-----+-----+
| LOWER(@x) | LOWER(CONVERT (@x USING latin1)) |
+-----+-----+
| AardVark | aardvark
+-----+-----+
```

## See Also

- [Character Sets and Collations](#)

## 1.1.12.2.15 ELT

### Syntax

```
ELT(N, str1[, str2, str3,...])
```

### Description

Takes a numeric argument and a series of string arguments. Returns the string that corresponds to the given numeric position. For instance, it returns

`str1`  
if  
`N`  
is 1,  
`str2`  
if  
`N`  
is 2, and so on. If the numeric argument is a

[FLOAT](#)

, MariaDB rounds it to the nearest

[INTEGER](#)

. If the numeric argument is less than 1, greater than the total number of arguments, or not a number,

`ELT()`

returns

`NULL`

. It must have at least two arguments.

It is complementary to the

[FIELD\(\)](#)

function.

## Examples

```
SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
+-----+
| ELT(1, 'ej', 'Heja', 'hej', 'foo') |
+-----+
| ej                                |
+-----+

SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');
+-----+
| ELT(4, 'ej', 'Heja', 'hej', 'foo') |
+-----+
| foo                               |
+-----+
```

## See also

- [FIND\\_IN\\_SET\(\)](#) function. Returns the position of a string in a set of strings.
- [FIELD\(\)](#) function. Returns the index position of a string in a list.

## 1.1.12.2.16 EXPORT\_SET

### Syntax

```
EXPORT_SET(bits, on, off[, separator[, number_of_bits]])
```

### Description

Takes a minimum of three arguments. Returns a string where each bit in the given

bits  
argument is returned, with the string values given for  
on  
and  
off

Bits are examined from right to left, (from low-order to high-order bits). Strings are added to the result from left to right, separated by a separator string (defaults as ' )

' ). You can optionally limit the number of bits the  
EXPORT\_SET()  
function examines using the  
number\_of\_bits  
option.

If any of the arguments are set as

NULL  
, the function returns  
NULL

## Examples

```

SELECT EXPORT_SET(5,'Y','N','','',4);
+-----+
| EXPORT_SET(5,'Y','N','','',4) |
+-----+
| Y,N,Y,N |
+-----+

SELECT EXPORT_SET(6,'1','0','','',10);
+-----+
| EXPORT_SET(6,'1','0','','',10) |
+-----+
| 0,1,1,0,0,0,0,0,0,0 |
+-----+

```

## 1.1.12.2.17 EXTRACTVALUE

### Syntax

```
EXTRACTVALUE(xml_frag, xpath_expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [Invalid Arguments](#)
  2. [Explicit text\(\) Expressions](#)
  3. [Count Matches](#)
  4. [Matches](#)
3. [Examples](#)

### Description

The

`EXTRACTVALUE()`

function takes two string arguments: a fragment of XML markup and an XPath expression, (also known as a locator). It returns the text (That is, CDDATA), of the first text node which is a child of the element or elements matching the XPath expression.

In cases where a valid XPath expression does not match any text nodes in a valid XML fragment, (including the implicit

```
/text()
expression), the
EXTRACTVALUE()
function returns an empty string.
```

### Invalid Arguments

When either the XML fragment or the XPath expression is

```
NULL
, the
EXTRACTVALUE()
function returns
NULL
```

. When the XML fragment is invalid, it raises a warning Code 1525:

```
Warning (Code 1525): Incorrect XML value: 'parse error at line 1 pos 11: unexpected END-OF-INPUT'
```

When the XPath value is invalid, it generates an Error 1105:

```
ERROR 1105 (HY000): XPATH syntax error: ''
```

### Explicit

`text()`  
`Expressions`

This function is the equivalent of performing a match using the XPath expression after appending

`/text()`

. In other words:

```

SELECT
EXTRACTVALUE('<cases><case>example</case></cases>', '/cases/case') AS 'Base Example',
EXTRACTVALUE('<cases><case>example</case></cases>', '/cases/case/text()') AS 'text() Example';

+-----+
| Base Example | text() Example |
+-----+
| example      | example       |
+-----+

```

## Count Matches

When

```
EXTRACTVALUE()
```

returns multiple matches, it returns the content of the first child text node of each matching element, in the matched order, as a single, space-delimited string.

By design, the

```
EXTRACTVALUE()
```

function makes no distinction between a match on an empty element and no match at all. If you need to determine whether no matching element was found in the XML fragment or if an element was found that contained no child text nodes, use the XPath

```
count()
function.
```

For instance, when looking for a value that exists, but contains no child text nodes, you would get a count of the number of matching instances:

```

SELECT
EXTRACTVALUE('<cases><case/></cases>', '/cases/case') AS 'Empty Example',
EXTRACTVALUE('<cases><case/></cases>', 'count(/cases/case)') AS 'count() Example';

+-----+
| Empty Example | count() Example |
+-----+
|                 |          1 |
+-----+

```

Alternatively, when looking for a value that doesn't exist,

```
count()
returns 0.
```

```

SELECT
EXTRACTVALUE('<cases><case/></cases>', '/cases/person') AS 'No Match Example',
EXTRACTVALUE('<cases><case/></cases>', 'count(/cases/person)') AS 'count() Example';

+-----+
| No Match Example | count() Example |
+-----+
|                   |          0 |
+-----+

```

## Matches

**Important :** The

```
EXTRACTVALUE()
```

function only returns CDDATA. It does not return tags that the element might contain or the text that these child elements contain.

```

SELECT EXTRACTVALUE('<cases><case>Person<email>x@example.com</email></case></cases>', '/cases') AS Case;

+-----+
| Case   |
+-----+
| Person |
+-----+

```

Note, in the above example, while the XPath expression matches to the parent

```
<case>
instance, it does not return the contained
<email>
tag or its content.
```

## Examples

```
SELECT
    ExtractValue('<a>ccc<b>ddd</b></a>', '/a')           AS val1,
    ExtractValue('<a>ccc<b>ddd</b></a>', '/a/b')        AS val2,
    ExtractValue('<a>ccc<b>ddd</b></a>', '//b')         AS val3,
    ExtractValue('<a>ccc<b>ddd</b></a>', '/b')          AS val4,
    ExtractValue('<a>ccc<b>ddd</b><b>eee</b></a>', '//b') AS val5;
+-----+-----+-----+-----+
| val1 | val2 | val3 | val4 | val5   |
+-----+-----+-----+-----+
| ccc  | ddd  | ddd  |      | ddd eee |
+-----+-----+-----+-----+
```

## 1.1.12.2.18 FIELD

### Syntax

```
FIELD(pattern, str1[,str2,...])
```

### Description

Returns the index position of the string or number matching the given pattern. Returns

0

in the event that none of the arguments match the pattern. Raises an Error 1582 if not given at least two arguments.

When all arguments given to the

```
FIELD()
```

function are strings, they are treated as case-insensitive. When all the arguments are numbers, they are treated as numbers. Otherwise, they are treated as doubles.

If the given pattern occurs more than once, the

```
FIELD()
```

function only returns the index of the first instance. If the given pattern is

NULL

, the function returns

0

, as a

NULL

pattern always fails to match.

This function is complementary to the

```
ELT()
```

function.

## Examples

```

SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo')
  AS 'Field Results';
+-----+
| Field Results |
+-----+
|      2 |
+-----+

SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo')
  AS 'Field Results';
+-----+
| Field Results |
+-----+
|      0 |
+-----+

SELECT FIELD(1, 2, 3, 4, 5, 1) AS 'Field Results';
+-----+
| Field Results |
+-----+
|      5 |
+-----+

SELECT FIELD(NULL, 2, 3) AS 'Field Results';
+-----+
| Field Results |
+-----+
|      0 |
+-----+

SELECT FIELD('fail') AS 'Field Results';
Error 1582 (42000): Incorrect parameter count in call
to native function 'field'

```

## See also

- [ELT\(\)](#) function. Returns the N'th element from a set of strings.

## 1.1.12.2.19 FIND\_IN\_SET

### Syntax

```
FIND_IN_SET(pattern, strlist)
```

### Description

Returns the index position where the given pattern occurs in a string list. The first argument is the pattern you want to search for. The second argument is a string containing comma-separated variables. If the second argument is of the

[SET](#)

data-type, the function is optimized to use bit arithmetic.

If the pattern does not occur in the string list or if the string list is an empty string, the function returns

0

. If either argument is

NULL

, the function returns

NULL

. The function does not return the correct result if the pattern contains a comma ("

,

") character.

## Examples

```
SELECT FIND_IN_SET('b','a,b,c,d') AS "Found Results";
+-----+
| Found Results |
+-----+
|      2      |
+-----+
```

## See Also

- [ELT\(\)](#) function. Returns the N'th element from a set of strings.

## 1.1.12.2.20 FORMAT

### Syntax

```
FORMAT(num, decimal_position[, locale])
```

### Description

Formats the given number for display as a string, adding separators to appropriate position and rounding the results to the given decimal position. For instance, it would format

```
15233.345
to
15,233.35
```

If the given decimal position is

```
0
```

, it rounds to return no decimal point or fractional part. You can optionally specify a [locale](#) value to format numbers to the pattern appropriate for the given region.

### Examples

```
SELECT FORMAT(1234567890.09876543210, 4) AS 'Format';
+-----+
| Format          |
+-----+
| 1,234,567,890.0988 |
+-----+
```

```
SELECT FORMAT(1234567.89, 4) AS 'Format';
+-----+
| Format          |
+-----+
| 1,234,567.8900 |
+-----+
```

```
SELECT FORMAT(1234567.89, 0) AS 'Format';
+-----+
| Format          |
+-----+
| 1,234,568      |
+-----+
```

```
SELECT FORMAT(123456789,2,'rm_CH') AS 'Format';
+-----+
| Format          |
+-----+
| 123'456'789,00 |
+-----+
```

## 1.1.12.2.21 FROM\_BASE64

### Syntax

```
FROM_BASE64(str)
```

## Description

Decodes the given base-64 encode string, returning the result as a binary string. Returns

NULL  
if the given string is  
NULL  
or if it's invalid.

It is the reverse of the

```
TO_BASE64
```

function.

There are numerous methods to base-64 encode a string. MariaDB uses the following:

- It encodes alphabet value 64 as '  
+'  
'.'
- It encodes alphabet value 63 as '  
/'  
'.'
- It codes output in groups of four printable characters. Each three byte of data encoded uses four characters. If the final group is incomplete, it pads the difference with the '=' character.
- It divides long output, adding a new line every 76 characters.
- In decoding, it recognizes and ignores newlines, carriage returns, tabs and space whitespace characters.

```
SELECT TO_BASE64('Maria') AS 'Input';
+-----+
| Input   |
+-----+
| TWFyaWE= |
+-----+
SELECT FROM_BASE64('TWFyaWE=') AS 'Output';
+-----+
| Output |
+-----+
| Maria  |
+-----+
```

## 1.1.12.2.22 HEX

### Syntax

```
HEX(N_or_S)
```

## Description

If

N\_or\_S  
is a number, returns a string representation of the hexadecimal value of  
N  
, where  
N  
is a  
longlong  
(  
  
BIGINT  
  
) number. This is equivalent to

```
CONV
```

```
(N,10,16)

If
N_or_S
is a string, returns a hexadecimal string representation of
N_or_S
where each byte of each character in
N_or_S
is converted to two hexadecimal digits. If
N_or_S
is NULL, returns NULL. The inverse of this operation is performed by the UNHEX \(\) function.
```

MariaDB starting with [10.5.0](#)

`HEX()` with an `INET6` argument returns a hexadecimal representation of the underlying 16-byte binary string.

## Examples

```
SELECT HEX(255);
+-----+
| HEX(255) |
+-----+
| FF         |
+-----+

SELECT 0x4D617269614442;
+-----+
| 0x4D617269614442 |
+-----+
| MariaDB          |
+-----+

SELECT HEX('MariaDB');
+-----+
| HEX('MariaDB') |
+-----+
| 4D617269614442 |
+-----+
```

From [MariaDB 10.5.0](#) :

```
SELECT HEX(CAST('2001:db8::ff00:42:8329' AS INET6));
+-----+
| HEX(CAST('2001:db8::ff00:42:8329' AS INET6)) |
+-----+
| 20010DB800000000000FF0000428329               |
+-----+
```

## See Also

- [Hexadecimal literals](#)
- [UNHEX\(\)](#)
- [CONV\(\)](#)
- [BIN\(\)](#)
- [OCT\(\)](#)

## 1.1.12.2.23 INSERT Function

### Syntax

```
INSERT(str,pos,len,newstr)
```

### Description

Returns the string

```
str  
, with the substring beginning at position  
pos  
and  
len  
characters long replaced by the string  
newstr  
. Returns the original string if  
pos  
is not within the length of the string. Replaces the rest of the string from position  
pos  
if  
len  
is not within the length of the rest of the string. Returns NULL if any argument is NULL.
```

## Examples

```
SELECT INSERT('Quadratic', 3, 4, 'What');  
+-----+  
| INSERT('Quadratic', 3, 4, 'What') |  
+-----+  
| QuWhattic |  
+-----+  
  
SELECT INSERT('Quadratic', -1, 4, 'What');  
+-----+  
| INSERT('Quadratic', -1, 4, 'What') |  
+-----+  
| Quadratic |  
+-----+  
  
SELECT INSERT('Quadratic', 3, 100, 'What');  
+-----+  
| INSERT('Quadratic', 3, 100, 'What') |  
+-----+  
| QuWhat |  
+-----+
```

## 1.1.12.2.24 LENGTH

### Syntax

```
LENGTH(str)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the length of the string

```
str
```

In the default mode, when [Oracle mode from MariaDB 10.3](#) is not set, the length is measured in bytes. In this case, a multi-byte character counts as multiple bytes. This means that for a string containing five two-byte characters,

```
LENGTH()  
returns 10, whereas CHAR_LENGTH() returns 5.
```

When running [Oracle mode from MariaDB 10.3](#), the length is measured in characters, and

```
LENGTH  
is a synonym for CHAR_LENGTH().
```

If

```
str
```

is not a string value, it is converted into a string. If

```
str  
is  
NULL  
, the function returns  
NULL
```

## Examples

```
SELECT LENGTH('MariaDB');  
+-----+  
| LENGTH('MariaDB') |  
+-----+  
| 7 |  
+-----+
```

When [Oracle mode](#) from MariaDB 10.3 is not set:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');  
+-----+-----+-----+  
| CHAR_LENGTH('π') | LENGTH('π') | LENGTHB('π') | OCTET_LENGTH('π') |  
+-----+-----+-----+  
| 1 | 2 | 2 | 2 |  
+-----+-----+-----+
```

In [Oracle mode](#) from MariaDB 10.3 :

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');  
+-----+-----+-----+  
| CHAR_LENGTH('π') | LENGTH('π') | LENGTHB('π') | OCTET_LENGTH('π') |  
+-----+-----+-----+  
| 1 | 1 | 2 | 2 |  
+-----+-----+-----+
```

## See Also

- [CHAR\\_LENGTH\(\)](#)
- [LENGTHB\(\)](#)
- [OCTET\\_LENGTH\(\)](#)
- [Oracle mode from MariaDB 10.3](#)

## 1.1.12.2.25 LENGTHB

MariaDB starting with [10.3.1](#)

Introduced in [MariaDB 10.3.1](#) as part of the [Oracle compatibility enhancements](#).

## Syntax

```
LENGTHB(str)
```

## Description

`LENGTHB()`

returns the length of the given string, in bytes. When [Oracle mode](#) is not set, this is a synonym for [LENGTH](#).

A multi-byte character counts as multiple bytes. This means that for a string containing five two-byte characters,

```
LENGTHB()  
returns 10, whereas CHAR_LENGTH() returns 5.
```

If

```
str  
is not a string value, it is converted into a string. If  
str  
is
```

NULL  
, the function returns  
NULL

## Examples

When Oracle mode from MariaDB 10.3 is not set:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+-----+-----+-----+
| CHAR_LENGTH('π') | LENGTH('π') | LENGTHB('π') | OCTET_LENGTH('π') |
+-----+-----+-----+
|          1 |         2 |         2 |         2 |
+-----+-----+-----+
```

In Oracle mode from MariaDB 10.3 :

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+-----+-----+-----+
| CHAR_LENGTH('π') | LENGTH('π') | LENGTHB('π') | OCTET_LENGTH('π') |
+-----+-----+-----+
|          1 |         1 |         2 |         2 |
+-----+-----+-----+
```

## See Also

- [CHAR\\_LENGTH\(\)](#)
- [LENGTH\(\)](#)
- [OCTET\\_LENGTH\(\)](#)

## 1.1.12.2.26 LIKE

### Syntax

```
expr LIKE pat [ESCAPE 'escape_char']
expr NOT LIKE pat [ESCAPE 'escape_char']
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [Optimizing LIKE](#)
5. [See Also](#)

## Description

Tests whether *expr* matches the pattern *pat*. Returns either 1 (

TRUE  
) or 0 (

FALSE

). Both *expr* and *pat* may be any valid expression and are evaluated to strings. Patterns may use the following wildcard characters:

- %  
matches any number of characters, including zero.
- \_  
matches any single character.

### Use

NOT LIKE

to test if a string does not match a pattern. This is equivalent to using the

NOT

operator on the entire  
LIKE

expression.

If either the expression or the pattern is

NULL  
, the result is  
NULL

LIKE

performs case-insensitive substring matches if the collation for the expression and pattern is case-insensitive. For case-sensitive matches, declare either argument to use a binary collation using

COLLATE

, or coerce either of them to a

BINARY

string using

CAST

. Use

SHOW COLLATION

to get a list of available collations. Collations ending in

\_bin

are case-sensitive.

Numeric arguments are coerced to binary strings.

The

— wildcard matches a single character, not byte. It will only match a multi-byte character if it is valid in the expression's character set. For example

— will match  
\_utf8"€"  
, but it will not match  
\_latin1"€"  
because the Euro sign is not a valid latin1 character. If necessary, use

CONVERT

to use the expression in a different character set.

If you need to match the characters

—  
or  
%

, you must escape them. By default, you can prefix the wildcard characters the backslash character

\

to escape them. The backslash is used both to encode special characters like newlines when a string is parsed as well as to escape wildcards in a pattern after parsing. Thus, to match an actual backslash, you sometimes need to double-escape it as

"\

\

\

\"

To avoid difficulties with the backslash character, you can change the wildcard escape character using

ESCAPE  
in a  
LIKE  
expression. The argument to  
ESCAPE  
must be a single-character string.

# Examples

Select the days that begin with "T":

```
CREATE TABLE t1 (d VARCHAR(16));
INSERT INTO t1 VALUES ("Monday"), ("Tuesday"), ("Wednesday"), ("Thursday"), ("Friday"), ("Saturday"), ("Sunday");
SELECT * FROM t1 WHERE d LIKE "T%";
```

```
SELECT * FROM t1 WHERE d LIKE "T%";
+-----+
| d      |
+-----+
| Tuesday |
| Thursday|
+-----+
```

Select the days that contain the substring "es":

```
SELECT * FROM t1 WHERE d LIKE "%es%";
```

```
SELECT * FROM t1 WHERE d LIKE "%es%";
+-----+
| d      |
+-----+
| Tuesday |
| Wednesday|
+-----+
```

Select the six-character day names:

```
SELECT * FROM t1 WHERE d like "__day";
```

```
SELECT * FROM t1 WHERE d like "__day%";
+-----+
| d      |
+-----+
| Monday |
| Friday |
| Sunday |
+-----+
```

With the default collations,

LIKE

is case-insensitive:

```
SELECT * FROM t1 where d like "t%";
```

```
SELECT * FROM t1 where d like "t%";
+-----+
| d      |
+-----+
| Tuesday |
| Thursday|
+-----+
```

Use

COLLATE

to specify a binary collation, forcing case-sensitive matches:

```
SELECT * FROM t1 WHERE d like "t%" COLLATE latin1_bin;
```

```
SELECT * FROM t1 WHERE d like "t%" COLLATE latin1_bin;
Empty set (0.00 sec)
```

You can include functions and operators in the expression to match. Select dates based on their day name:

```
CREATE TABLE t2 (d DATETIME);
INSERT INTO t2 VALUES
  ("2007-01-30 21:31:07"),
  ("1983-10-15 06:42:51"),
  ("2011-04-21 12:34:56"),
  ("2011-10-30 06:31:41"),
  ("2011-01-30 14:03:25"),
  ("2004-10-07 11:19:34");
SELECT * FROM t2 WHERE DAYNAME(d) LIKE "T%";
```

```
SELECT * FROM t2 WHERE DAYNAME(d) LIKE "T%";
+-----+
| d      |
+-----+
| 2007-01-30 21:31 |
| 2011-04-21 12:34 |
| 2004-10-07 11:19 |
+-----+
3 rows in set, 7 warnings (0.00 sec)
```

## Optimizing LIKE

- MariaDB can use indexes for LIKE on string columns in the case where the LIKE doesn't start with % or \_.
- Starting from [MariaDB 10.0](#), one can set the `optimizer_use_condition_selectivity` variable to 5. If this is done, then the optimizer will read `optimizer_selectivity_sampling_limit` rows to calculate the selectivity of the LIKE expression before starting to calculate the query plan. This can help speed up some LIKE queries by providing the optimizer with more information about your data.

## See Also

- For searches on text columns, with results sorted by relevance, see [full-text](#) indexes.
- For more complex searches and operations on strings, you can use [regular expressions](#), which were enhanced in MariaDB 10 (see [PCRE Regular Expressions](#)).

## 1.1.12.2.27 LOAD\_FILE

## 1.1.12.2.28 LOCATE

### Syntax

```
LOCATE(substr,str), LOCATE(substr,str,pos)
```

### Description

The first syntax returns the position of the first occurrence of substring

substr  
in string  
str  
. The second syntax returns the position of the first occurrence of substring  
substr  
in string  
str  
, starting at position  
pos  
. Returns 0 if  
substr  
is not in  
str  
.

`LOCATE()`  
performs a case-insensitive search.

If any argument is

`NULL`  
, returns  
`NULL`.

`INSTR()` is the same as the two-argument form of  
`LOCATE()`, except that the order of the arguments is reversed.

## Examples

```
SELECT LOCATE('bar', 'foobarbar');
+-----+
| LOCATE('bar', 'foobarbar') |
+-----+
|          4 |
+-----+  
  
SELECT LOCATE('My', 'Maria');
+-----+
| LOCATE('My', 'Maria') |
+-----+
|          0 |
+-----+  
  
SELECT LOCATE('bar', 'foobarbar', 5);
+-----+
| LOCATE('bar', 'foobarbar', 5) |
+-----+
|          7 |
+-----+
```

## See Also

- [INSTR\(\)](#) ; Returns the position of a string within a string
- [SUBSTRING\\_INDEX\(\)](#) ; Returns the substring from string before count occurrences of a delimiter

## 1.1.12.2.29 LOWER

### Syntax

```
LOWER(str)
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

### Description

Returns the string

`str`

with all characters changed to lowercase according to the current character set mapping. The default is latin1 (cp1252 West European).

## Examples

```
SELECT LOWER('QUADRATICALLY');
+-----+
| LOWER('QUADRATICALLY') |
+-----+
| quadratically         |
+-----+
```

LOWER()  
(and  
UPPER()

) are ineffective when applied to binary strings (

BINARY

,

VARBINARY

,

BLOB

). To perform lowercase conversion, [CONVERT](#) the string to a non-binary string:

```
SET @str = BINARY 'North Carolina';

SELECT LOWER(@str), LOWER(CONVERT(@str USING latin1));
+-----+-----+
| LOWER(@str) | LOWER(CONVERT(@str USING latin1)) |
+-----+-----+
| North Carolina | north carolina           |
+-----+-----+
```

## 1.1.12.2.30 LPAD

### Syntax

```
LPAD(str, len [,padstr])
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the string

str  
, left-padded with the string  
padstr  
to a length of  
len  
characters. If  
str  
is longer than  
len  
, the return value is shortened to  
len  
characters. If  
padstr  
is omitted, the LPAD function pads spaces.

Prior to [MariaDB 10.3.1](#) , the

padstr  
parameter was mandatory.

Returns NULL if given a NULL argument. If the result is empty (zero length), returns either an empty string or, from [MariaDB 10.3.6](#) with [SQL\\_MODE=Oracle](#) , NULL.

The Oracle mode version of the function can be accessed outside of Oracle mode by using

LPAD\_ORACLE

as the function name.

## Examples

```
SELECT LPAD('hello',10,'.');
+-----+
| LPAD('hello',10,'.') |
+-----+
| .....hello         |
+-----+

SELECT LPAD('hello',2,'.');
+-----+
| LPAD('hello',2,'.') |
+-----+
| he                  |
+-----+
```

From [MariaDB 10.3.1](#), with the pad string defaulting to space.

```
SELECT LPAD('hello',10);
+-----+
| LPAD('hello',10) |
+-----+
|      hello      |
+-----+
```

Oracle mode version from [MariaDB 10.3.6](#):

```
SELECT LPAD('',0),LPAD_ORACLE('');
+-----+
| LPAD('',0) | LPAD_ORACLE('') |
+-----+
|          | NULL           |
+-----+
```

## See Also

- [RPAD](#) - Right-padding instead of left-padding.

## 1.1.12.2.31 LTRIM

### Syntax

```
LTRIM(str)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Returns the string

str

with leading space characters removed.

Returns NULL if given a NULL argument. If the result is empty, returns either an empty string, or, from [MariaDB 10.3.6](#) with `SQL_MODE=Oracle`, NULL.

The Oracle mode version of the function can be accessed outside of Oracle mode by using

`LTRIM_ORACLE`

as the function name.

## Examples

```
SELECT QUOTE(LTRIM(' MariaDB '));
+-----+
| QUOTE(LTRIM(' MariaDB ')) |
+-----+
| 'MariaDB' |
+-----+
```

Oracle mode version from MariaDB 10.3.6 :

```
SELECT LTRIM(''),LTRIM_ORACLE('');
+-----+
| LTRIM('') | LTRIM_ORACLE('') |
+-----+
|        | NULL |
+-----+
```

## See Also

- [RTRIM](#) - trailing spaces removed
- [TRIM](#) - removes all given prefixes or suffixes

## 1.1.12.2.32 MAKE\_SET

### Syntax

```
MAKE_SET(bits,str1,str2,...)
```

### Description

Returns a set value (a string containing substrings separated by "," characters) consisting of the strings that have the corresponding bit in bits set.

*str1*  
corresponds to bit 0,  
*str2*  
to bit 1, and so on. NULL values in  
*str1*  
,  
*str2*  
, ... are not appended to the result.

### Examples

```

SELECT MAKE_SET(1,'a','b','c');
+-----+
| MAKE_SET(1,'a','b','c') |
+-----+
| a |
+-----+

SELECT MAKE_SET(1 | 4,'hello','nice','world');
+-----+
| MAKE_SET(1 | 4,'hello','nice','world') |
+-----+
| hello,world |
+-----+

SELECT MAKE_SET(1 | 4,'hello','nice',NULL,'world');
+-----+
| MAKE_SET(1 | 4,'hello','nice',NULL,'world') |
+-----+
| hello |
+-----+

SELECT QUOTE(MAKE_SET(0,'a','b','c'));
+-----+
| QUOTE(MAKE_SET(0,'a','b','c')) |
+-----+
| '' |
+-----+

```

## 1.1.12.2.33 MATCH AGAINST

### Syntax

```
MATCH (col1,col2,...) AGAINST (expr [search_modifier])
```

### Description

A special construct used to perform a fulltext search on a fulltext index.

See [Fulltext Index Overview](#) for a full description, and [Full-text Indexes](#) for more articles on the topic.

### Examples

```

CREATE TABLE ft_myisam(copy TEXT,FULLTEXT(copy)) ENGINE=MyISAM;

INSERT INTO ft_myisam(copy) VALUES ('Once upon a time'), ('There was a wicked witch'),
('Who ate everybody up');

SELECT * FROM ft_myisam WHERE MATCH(copy) AGAINST('wicked');
+-----+
| copy |
+-----+
| There was a wicked witch |
+-----+

```

```

SELECT id, body, MATCH (title,body) AGAINST
('Security implications of running MySQL as root'
IN NATURAL LANGUAGE MODE) AS score
FROM articles WHERE MATCH (title,body) AGAINST
('Security implications of running MySQL as root'
IN NATURAL LANGUAGE MODE);
+-----+-----+
| id | body | score |
+-----+-----+
| 4 | 1. Never run mysqld as root. 2. ... | 1.5219271183014 |
| 6 | When configured properly, MySQL ... | 1.3114095926285 |
+-----+-----+

```

## 1.1.12.2.34 Full-Text Index Stopwords

## Contents

1. MyISAM Stopwords
2. InnoDB Stopwords

Stopwords are used to provide a list of commonly-used words that can be ignored for the purposes of [Full-text-indexes](#).

Full-text indexes built in [MyISAM](#) and [InnoDB](#) have different stopword lists by default.

## MyISAM Stopwords

For full-text indexes on MyISAM tables, by default, the list is built from the file

`storage/myisam/ft_static.c`

, and searched using the server's character set and collation. The [ft\\_stopword\\_file](#) system variable allows the default list to be overridden with words from another file, or for stopwords to be ignored altogether.

If the stopword list is changed, any existing full-text indexes need to be rebuilt

The following table shows the default list of stopwords, although you should always treat

`storage/myisam/ft_static.c`

as the definitive list. See the [Fulltext Index Overview](#) for more details, and [Full-text-indexes](#) for related articles.

a's	able	about	above
according	accordingly	across	actually
after	afterwards	again	against
ain't	all	allow	allows
almost	alone	along	already
also	although	always	am
among	amongst	an	and
another	any	anybody	anyhow
anyone	anything	anyway	anyways
anywhere	apart	appear	appreciate
appropriate	are	aren't	around
as	aside	ask	asking
associated	at	available	away
awfully	be	became	because
become	becomes	becoming	been
before	beforehand	behind	being
believe	below	beside	besides
best	better	between	beyond
both	brief	but	by
c'mon	c's	came	can
can't	cannot	cant	cause
causes	certain	certainly	changes
clearly	co	com	come
comes	concerning	consequently	consider
considering	contain	containing	contains
corresponding	could	couldn't	course
currently	definitely	described	despite
did	didn't	different	do
does	doesn't	doing	don't
done	down	downwards	during
each	edu	eg	eight
either	else	elsewhere	enough

entirely	especially	et	etc
even	ever	every	everybody
everyone	everything	everywhere	ex
exactly	example	except	far
few	fifth	first	five
followed	following	follows	for
former	formerly	forth	four
from	further	furthermore	get
gets	getting	given	gives
go	goes	going	gone
got	gotten	greetings	had
hadn't	happens	hardly	has
hasn't	have	haven't	having
he	he's	hello	help
hence	her	here	here's
hereafter	hereby	herein	hereupon
hers	herself	hi	him
himself	his	hither	hopefully
how	howbeit	however	i'd
i'll	i'm	i've	ie
if	ignored	immediate	in
inasmuch	inc	indeed	indicate
indicated	indicates	inner	insofar
instead	into	inward	is
isn't	it	it'd	it'll
it's	its	itself	just
keep	keeps	kept	know
knows	known	last	lately
later	latter	latterly	least
less	lest	let	let's
like	liked	likely	little
look	looking	looks	Itd
mainly	many	may	maybe
me	mean	meanwhile	merely
might	more	moreover	most
mostly	much	must	my
myself	name	namely	nd
near	nearly	necessary	need
needs	neither	never	nevertheless
new	next	nine	no
nobody	non	none	noone
nor	normally	not	nothing
novel	now	nowhere	obviously
of	off	often	oh
ok	okay	old	on

once	one	ones	only
onto	or	other	others
otherwise	ought	our	ours
ourselves	out	outside	over
overall	own	particular	particularly
per	perhaps	placed	please
plus	possible	presumably	probably
provides	que	quite	qv
rather	rd	re	really
reasonably	regarding	regardless	regards
relatively	respectively	right	said
same	saw	say	saying
says	second	secondly	see
seeing	seem	seemed	seeming
seems	seen	self	selves
sensible	sent	serious	seriously
seven	several	shall	she
should	shouldn't	since	six
so	some	somebody	somewhat
someone	something	sometime	sometimes
somewhat	somewhere	soon	sorry
specified	specify	specifying	still
sub	such	sup	sure
t's	take	taken	tell
tends	th	than	thank
thanks	thanx	that	that's
thats	the	their	theirs
them	themselves	then	thence
there	there's	thereafter	thereby
therefore	therein	theres	thereupon
these	they	they'd	they'll
they're	they've	think	third
this	thorough	thoroughly	those
though	three	through	throughout
thru	thus	to	together
too	took	toward	towards
tried	tries	truly	try
trying	twice	two	un
under	unfortunately	unless	unlikely
until	unto	up	upon
us	use	used	useful
uses	using	usually	value
various	very	via	viz
vs	want	wants	was
wasn't	way	we	we'd

we'll	we're	we've	welcome
well	went	were	weren't
what	what's	whatever	when
whence	whenever	where	where's
whereafter	whereas	whereby	wherein
whereupon	wherever	whether	which
while	whither	who	who's
whoever	whole	whom	whose
why	will	willing	wish
with	within	without	won't
wonder	would	wouldn't	yes
yet	you	you'd	you'll
you're	you've	your	yours
yourself	yourselves	zero	

## InnoDB Stopwords

Stopwords on full-text indexes are only enabled if the `innodb_ft_enable_stopword` system variable is set (by default it is) at the time the index was created.

The stopword list is determined as follows:

- If the `innodb_ft_user_stopword_table` system variable is set, that table is used as a stopword list.
- If
  - `innodb_ft_user_stopword_table` is not set, the table set by `innodb_ft_server_stopword_table` is used.
- If neither variable is set, the built-in list is used, which can be viewed by querying the `INNODB_FT_DEFAULT_STOPWORD` table in the `Information Schema`.

In the first two cases, the specified table must exist at the time the system variable is set and the full-text index created. It must be an InnoDB table with a single column, a `VARCHAR` named `VALUE`.

The default InnoDB stopword list differs from the default MyISAM list, being much shorter, and contains the following words:

a	about	an	are
as	at	be	by
com	de	en	for
from	how	i	in
is	it	la	of
on	or	that	the
this	to	was	what
when	where	who	will
with	und	the	www

## 1.1.12.2.35 MID

### Syntax

```
MID(str,pos,len)
```

### Description

`MID(str,pos,len)` is a synonym for `SUBSTRING(str,pos,len)`.

### Examples

```
SELECT MID('abcd',4,1);
+-----+
| MID('abcd',4,1) |
+-----+
| d               |
+-----+

SELECT MID('abcd',2,2);
+-----+
| MID('abcd',2,2) |
+-----+
| bc              |
+-----+
```

A negative starting position:

```
SELECT MID('abcd',-2,4);
+-----+
| MID('abcd',-2,4) |
+-----+
| cd              |
+-----+
```

## 1.1.12.2.36 NOT LIKE

### Syntax

```
expr NOT LIKE pat [ESCAPE 'escape_char']
```

### Description

This is the same as [NOT \(expr LIKE pat \[ESCAPE 'escape\\_char'\]\)](#).

## 1.1.12.2.37 NOT REGEXP

## 1.1.12.2.38 OCTET\_LENGTH

### Syntax

```
OCTET_LENGTH(str)
```

### Description

`OCTET_LENGTH()`

returns the length of the given string, in octets (bytes). This is a synonym for [LENGTHB\(\)](#), and, when [Oracle mode](#) from [MariaDB 10.3](#) is not set, a synonym for [LENGTH\(\)](#).

A multi-byte character counts as multiple bytes. This means that for a string containing five two-byte characters,

`OCTET_LENGTH()`

returns 10, whereas [CHAR\\_LENGTH\(\)](#) returns 5.

If

str

is not a string value, it is converted into a string. If

str

is

NULL

, the function returns

NULL

## Examples

When Oracle mode from MariaDB 10.3 is not set:

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+-----+-----+-----+
| CHAR_LENGTH('π') | LENGTH('π') | LENGTHB('π') | OCTET_LENGTH('π') |
+-----+-----+-----+
|          1 |         2 |         2 |         2 |
+-----+-----+
```

In Oracle mode from MariaDB 10.3 :

```
SELECT CHAR_LENGTH('π'), LENGTH('π'), LENGTHB('π'), OCTET_LENGTH('π');
+-----+-----+-----+
| CHAR_LENGTH('π') | LENGTH('π') | LENGTHB('π') | OCTET_LENGTH('π') |
+-----+-----+-----+
|          1 |         1 |         2 |         2 |
+-----+-----+
```

## See Also

- [CHAR\\_LENGTH\(\)](#)
- [LENGTH\(\)](#)
- [LENGTHB\(\)](#)
- Oracle mode from MariaDB 10.3

## 1.1.12.2.39 ORD

### Syntax

```
ORD(str)
```

### Description

If the leftmost character of the string

str

is a multi-byte character, returns the code for that character, calculated from the numeric values of its constituent bytes using this formula:

```
(1st byte code)
+ (2nd byte code x 256)
+ (3rd byte code x 256 x 256) ...
```

If the leftmost character is not a multi-byte character, ORD() returns the same value as the [ASCII\(\)](#) function.

### Examples

```
SELECT ORD('2');
+-----+
| ORD('2') |
+-----+
|      50 |
+-----+
```

## See Also

- [ASCII\(\)](#) - Return ASCII value of first character
- [CHAR\(\)](#) - Create a character from an integer value

## 1.1.12.2.40 POSITION

### Syntax

```
POSITION(substr IN str)
```

## Description

POSITION(substr IN str) is a synonym for LOCATE(substr,str).

It's part of ODBC 3.0.

### 1.1.12.2.41 QUOTE

#### Syntax

```
QUOTE(str)
```

#### Description

Quotes a string to produce a result that can be used as a properly escaped data value in an SQL statement. The string is returned enclosed by single quotes and with each instance of single quote ("

"), backslash ("\\"), ASCII NUL , and Control-Z preceded by a backslash. If the argument is NULL , the return value is the word "NULL" without enclosing single quotes.

#### Examples

```
SELECT QUOTE("Don't!");
+-----+
| QUOTE("Don't!") |
+-----+
| 'Don\''t!' |
+-----+
SELECT QUOTE(NULL);
+-----+
| QUOTE(NULL) |
+-----+
| NULL |
+-----+
```

### 1.1.12.2.42 QUOTE\_IDENTIFIER

#### Syntax

```
SYS.QUOTE_IDENTIFIER(str)
```

#### Description

Quotes a string to produce a result that can be used as an identifier in an SQL statement. The string is returned enclosed by backticks (`

`) and with each instance of backtick (`") doubled. If the argument is NULL , the return value is the word "NULL" without enclosing backticks.

#### Examples

```
SELECT SYS.QUOTE_IDENTIFIER("Identifier with spaces");
+-----+
| SYS.QUOTE_IDENTIFIER("Identifier with spaces") |
+-----+
| `Identifier with spaces` |
+-----+  
  
SELECT SYS.QUOTE_IDENTIFIER("Identifier` containing `backticks");
+-----+
| SYS.QUOTE_IDENTIFIER("Identifier` containing `backticks") |
+-----+
| `Identifier`` containing ``backticks` |
+-----+
```

## 1.1.12.2.43 REPEAT Function

### Syntax

```
REPEAT(str,count)
```

### Description

Returns a string consisting of the string

str  
repeated  
count  
times. If  
count  
is less than 1, returns an empty string. Returns NULL if  
str  
or  
count  
are NULL.

### Examples

```
SELECT QUOTE(REPEAT('MariaDB ',4));
+-----+
| QUOTE(REPEAT('MariaDB ',4)) |
+-----+
| 'MariaDB MariaDB MariaDB MariaDB ' |
+-----+
```

## 1.1.12.2.44 REPLACE Function

### Syntax

```
REPLACE(str,from_str,to_str)
```

### Description

Returns the string

str  
with all occurrences of the string  
from\_str  
replaced by the string  
to\_str  
. REPLACE() performs a case-sensitive match when searching for  
from\_str

### Examples

```
SELECT REPLACE('www.mariadb.org', 'w', 'Ww');
+-----+
| REPLACE('www.mariadb.org', 'w', 'Ww') |
+-----+
| WwWwWw.mariadb.org           |
+-----+
```

## 1.1.12.2.45 REVERSE

### Syntax

```
REVERSE(str)
```

### Description

Returns the string

*str*

with the order of the characters reversed.

### Examples

```
SELECT REVERSE('desserts');
+-----+
| REVERSE('desserts') |
+-----+
| stressed            |
+-----+
```

## 1.1.12.2.46 RIGHT

### Syntax

```
RIGHT(str,len)
```

### Description

Returns the rightmost

*Len*

characters from the string

*str*

, or NULL if any argument is NULL.

### Examples

```
SELECT RIGHT('MariaDB', 2);
+-----+
| RIGHT('MariaDB', 2) |
+-----+
| DB                 |
+-----+
```

## 1.1.12.2.47 RPAD

### Syntax

```
RPAD(str, len [, padstr])
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Returns the string

```
str
, right-padded with the string
padstr
to a length of
len
characters. If
str
is longer than
len
, the return value is shortened to
len
characters. If
padstr
is omitted, the RPAD function pads spaces.
```

Prior to [MariaDB 10.3.1](#), the

```
padstr
parameter was mandatory.
```

Returns NULL if given a NULL argument. If the result is empty (a length of zero), returns either an empty string, or, from [MariaDB 10.3.6](#) with [SQL\\_MODE=Oracle](#), NULL.

The Oracle mode version of the function can be accessed outside of Oracle mode by using

```
RPAD_ORACLE
as the function name.
```

## Examples

```
SELECT RPAD('hello',10,'.');
+-----+
| RPAD('hello',10,'.') |
+-----+
| hello..... |
+-----+

SELECT RPAD('hello',2,'.');
+-----+
| RPAD('hello',2,'.') |
+-----+
| he |
+-----+
```

From [MariaDB 10.3.1](#), with the pad string defaulting to space.

```
SELECT RPAD('hello',30);
+-----+
| RPAD('hello',30) |
+-----+
| hello |
+-----+
```

Oracle mode version from [MariaDB 10.3.6](#):

```
SELECT RPAD('',0),RPAD_ORACLE('',0);
+-----+
| RPAD('',0) | RPAD_ORACLE('',0) |
+-----+
|          | NULL           |
+-----+
```

## See Also

- [LPAD](#) - Left-padding instead of right-padding.

## 1.1.12.2.48 RTRIM

### Syntax

```
RTRIM(str)
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the string

```
str
with trailing space characters removed.
```

Returns NULL if given a NULL argument. If the result is empty, returns either an empty string, or, from [MariaDB 10.3.6](#) with `SQL_MODE=Oracle`, NULL.

The Oracle mode version of the function can be accessed outside of Oracle mode by using

```
RTRIM_ORACLE
as the function name.
```

### Examples

```
SELECT QUOTE(RTRIM('MariaDB    '));
+-----+
| QUOTE(RTRIM('MariaDB    ')) |
+-----+
| 'MariaDB'                   |
+-----+
```

Oracle mode version from [MariaDB 10.3.6](#):

```
SELECT RTRIM(''),RTRIM_ORACLE('');
+-----+
| RTRIM('') | RTRIM_ORACLE('') |
+-----+
|        | NULL           |
+-----+
```

### See Also

- [LTRIM](#) - leading spaces removed
- [TRIM](#) - removes all given prefixes or suffixes

## 1.1.12.2.49 SFORMAT

MariaDB starting with [10.7.0](#)

SFORMAT was added in [MariaDB 10.7.0](#).

### Description

The

```
SFORMAT
```

function takes an input string and a formatting specification and returns the string formatted using the rules the user passed in the specification.

It uses the [fmtlib library](#) for Python-like (as well as Rust, C++20, etc) string formatting.

Only fmtlib 7.0.0+ is supported.

There is no native support for temporal and decimal values:

- TIME\_RESULT is handled as STRING\_RESULT

- DECIMAL\_RESULT as REAL\_RESULT

## Examples

```

SELECT SFORMAT("The answer is {}.", 42);
+-----+
| SFORMAT("The answer is {}.", 42) |
+-----+
| The answer is 42.                |
+-----+

CREATE TABLE test_sformat(mdb_release char(6), mdev int, feature char(20));

INSERT INTO test_sformat VALUES('10.7.0', 25015, 'Python style sformat'),
 ('10.7.0', 4958, 'UUID');

SELECT * FROM test_sformat;
+-----+-----+
| mdb_release | mdev   | feature      |
+-----+-----+
| 10.7.0     | 25015  | Python style sformat |
| 10.7.0     | 4958   | UUID          |
+-----+-----+

SELECT SFORMAT('MariaDB Server {} has a preview for MDEV-{} which is about {}',
    mdb_release, mdev, feature) AS 'Preview Release Examples'
FROM test_sformat;
+-----+
| Preview Release Examples           |
+-----+
| MariaDB Server 10.7.0 has a preview for MDEV-25015 which is about Python style sformat |
| MariaDB Server 10.7.0 has a preview for MDEV-4958 which is about UUID                      |
+-----+

```

## See Also

- [10.7 preview feature: Python-like string formatting](#)

## 1.1.12.2.50 SOUNDEx

### Syntax

`SOUNDEx(str)`

### Description

Returns a soundex string from

`str`  
 . Two strings that sound almost the same should have identical soundex strings. A standard soundex string is four characters long, but the `SOUNDEx()` function returns an arbitrarily long string. You can use `SUBSTRING()` on the result to get a standard soundex string. All non-alphabetic characters in `str` are ignored. All international alphabetic characters outside the A-Z range are treated as vowels.

**Important:** When using `SOUNDEx()`, you should be aware of the following details:

- This function, as currently implemented, is intended to work well with strings that are in the English language only. Strings in other languages may not produce reasonable results.
- This function implements the original Soundex algorithm, not the more popular enhanced version (also described by D. Knuth). The difference is that original version discards vowels first and duplicates second, whereas the enhanced version discards duplicates first and vowels second.

## Examples

```
SOUNDEX('Hello');
+-----+
| SOUNDEX('Hello') |
+-----+
| H400           |
+-----+
```

```
SELECT SOUNDEX('MariaDB');
+-----+
| SOUNDEX('MariaDB') |
+-----+
| M631           |
+-----+
```

```
SELECT SOUNDEX('Knowledgebase');
+-----+
| SOUNDEX('Knowledgebase') |
+-----+
| K543212          |
+-----+
```

```
SELECT givenname, surname FROM users WHERE SOUNDEX(givenname) = SOUNDEX("robert");
+-----+-----+
| givenname | surname |
+-----+-----+
| Roberto   | Castro  |
+-----+-----+
```

## See Also

- [SOUNDS LIKE \(\)](#)

## 1.1.12.2.51 SOUNDS LIKE

### Syntax

```
expr1 SOUNDS LIKE expr2
```

### Description

This is the same as

```
SOUNDEX
```

```
(expr1) = SOUNDEX(expr2)
```

### Example

```
SELECT givenname, surname FROM users WHERE givenname SOUNDS LIKE "robert";
+-----+-----+
| givenname | surname |
+-----+-----+
| Roberto   | Castro  |
+-----+-----+
```

## 1.1.12.2.52 SPACE

### Syntax

```
SPACE(N)
```

## Description

Returns a string consisting of

$N$   
space characters. If  
 $N$   
is NULL, returns NULL.

## Examples

```
SELECT QUOTE(SPACE(6));  
+-----+  
| QUOTE(SPACE(6)) |  
+-----+  
| ' ' ' ' ' ' '|  
+-----+
```

## 1.1.12.2.53 STRCMP

### Syntax

```
STRCMP(expr1,expr2)
```

### Description

STRCMP()  
returns  
0  
if the strings are the same,  
-1  
if the first argument is smaller than the second according to the current sort order, and  
1  
if the strings are otherwise not the same. Returns  
NULL  
is either argument is  
NULL  
.

## Examples

```
SELECT STRCMP('text', 'text2');  
+-----+  
| STRCMP('text', 'text2') |  
+-----+  
| -1 |  
+-----+  
  
SELECT STRCMP('text2', 'text');  
+-----+  
| STRCMP('text2', 'text') |  
+-----+  
| 1 |  
+-----+  
  
SELECT STRCMP('text', 'text');  
+-----+  
| STRCMP('text', 'text') |  
+-----+  
| 0 |  
+-----+
```

## 1.1.12.2.54 SUBSTR

# Description

SUBSTR()  
is a synonym for

[SUBSTRING](#)

()

## 1.1.12.2.55 SUBSTRING

### Syntax

```
SUBSTRING(str,pos),  
SUBSTRING(str FROM pos),  
SUBSTRING(str,pos,len),  
SUBSTRING(str FROM pos FOR len)  
  
SUBSTR(str,pos),  
SUBSTR(str FROM pos),  
SUBSTR(str,pos,len),  
SUBSTR(str FROM pos FOR len)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

The forms without a

*Len*  
argument return a substring from string  
*str*  
starting at position  
*pos*

The forms with a

*Len*  
argument return a substring  
*Len*  
characters long from string  
*str*  
, starting at position  
*pos*

The forms that use

*FROM*  
are standard SQL syntax.

It is also possible to use a negative value for

*pos*  
. In this case, the beginning of the substring is  
*pos*  
characters from the end of the string, rather than the beginning. A negative value may be used for  
*pos*  
in any of the forms of this function.

By default, the position of the first character in the string from which the substring is to be extracted is reckoned as 1. For [Oracle-compatibility](#) , from [MariaDB 10.3.3](#) , when `sql_mode` is set to 'oracle', position zero is treated as position 1 (although the first character is still reckoned as 1).

If any argument is

NULL  
, returns  
NULL

## Examples

```
SELECT SUBSTRING('Knowledgebase',5);
+-----+
| SUBSTRING('Knowledgebase',5) |
+-----+
| ledgebase                   |
+-----+

SELECT SUBSTRING('MariaDB' FROM 6);
+-----+
| SUBSTRING('MariaDB' FROM 6) |
+-----+
| DB                           |
+-----+

SELECT SUBSTRING('Knowledgebase',3,7);
+-----+
| SUBSTRING('Knowledgebase',3,7) |
+-----+
| owledge                      |
+-----+

SELECT SUBSTRING('Knowledgebase', -4);
+-----+
| SUBSTRING('Knowledgebase', -4) |
+-----+
| base                         |
+-----+

SELECT SUBSTRING('Knowledgebase', -8, 4);
+-----+
| SUBSTRING('Knowledgebase', -8, 4) |
+-----+
| edge                          |
+-----+

SELECT SUBSTRING('Knowledgebase' FROM -8 FOR 4);
+-----+
| SUBSTRING('Knowledgebase' FROM -8 FOR 4) |
+-----+
| edge                         |
+-----+
```

Oracle mode from MariaDB 10.3.3 :

```

SELECT SUBSTR('abc',0,3);
+-----+
| SUBSTR('abc',0,3) |
+-----+
|      |
+-----+

SELECT SUBSTR('abc',1,2);
+-----+
| SUBSTR('abc',1,2) |
+-----+
| ab            |
+-----+

SET sql_mode='oracle';

SELECT SUBSTR('abc',0,3);
+-----+
| SUBSTR('abc',0,3) |
+-----+
| abc           |
+-----+

SELECT SUBSTR('abc',1,2);
+-----+
| SUBSTR('abc',1,2) |
+-----+
| ab            |
+-----+

```

## See Also

- [INSTR\(\)](#) - Returns the position of a string within a string
- [LOCATE\(\)](#) - Returns the position of a string within a string
- [SUBSTRING\\_INDEX\(\)](#) - Returns a string based on substring

## 1.1.12.2.56 SUBSTRING\_INDEX

### Syntax

```
SUBSTRING_INDEX(str,delim,count)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the substring from string

str  
before count occurrences of the delimiter  
delim  
. If  
count  
is positive, everything to the left of the final delimiter (counting from the left) is returned. If  
count  
is negative, everything to the right of the final delimiter (counting from the right) is returned.  
SUBSTRING\_INDEX()  
performs a case-sensitive match when searching for  
delim

If any argument is

NULL  
, returns  
NULL

For example

```
SUBSTRING_INDEX('www.mariadb.org', '.', 2)
```

means "Return all of the characters up to the 2nd occurrence of ."

## Examples

```
SELECT SUBSTRING_INDEX('www.mariadb.org', '.', 2);
+-----+
| SUBSTRING_INDEX('www.mariadb.org', '.', 2) |
+-----+
| www.mariadb                         |
+-----+

SELECT SUBSTRING_INDEX('www.mariadb.org', '.', -2);
+-----+
| SUBSTRING_INDEX('www.mariadb.org', '.', -2) |
+-----+
| mariadb.org                          |
+-----+
```

## See Also

- [INSTR\(\)](#) - Returns the position of a string within a string
- [LOCATE\(\)](#) - Returns the position of a string within a string
- [SUBSTRING\(\)](#) - Returns a string based on position

## 1.1.12.2.57 TO\_BASE64

### Syntax

```
TO_BASE64(str)
```

### Description

Converts the string argument

str

to its base-64 encoded form, returning the result as a character string in the connection character set and collation.

The argument

str

will be converted to string first if it is not a string. A NULL argument will return a NULL result.

The reverse function, [FROM\\_BASE64\(\)](#), decodes an encoded base-64 string.

There are a numerous different methods to base-64 encode a string. The following are used by MariaDB and MySQL:

- Alphabet value 64 is encoded as '+'.
- Alphabet value 63 is encoded as '/'.
- Encoding output is made up of groups of four printable characters, with each three bytes of data encoded using four characters. If the final group is not complete, it is padded with '=' characters to make up a length of four.
- To divide long output, a newline is added after every 76 characters.
- Decoding will recognize and ignore newlines, carriage returns, tabs, and spaces.

## Examples

```
SELECT TO_BASE64('Maria');
+-----+
| TO_BASE64('Maria') |
+-----+
| TWFyaWE=          |
+-----+
```

## 1.1.12.2.58 TO\_CHAR

MariaDB starting with 10.6.1

The TO\_CHAR function was introduced in MariaDB 10.6.1 to enhance Oracle compatibility.

## Syntax

```
TO_CHAR(expr[, fmt])
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

TO\_CHAR

function converts an *expr* of type `date`, `datetime`, `time` or `timestamp` to a string. The optional *fmt* argument supports `YYYY/YYYY/YY/RRRR/RR/MM/MON/MONTH/MI/DD/DY/HH/HH12/HH24/SS` and special characters. The default value is "YYYY-MM-DD HH24:MI:SS".

In Oracle, TO\_CHAR can also be used to convert numbers to strings, but this is not supported in MariaDB and will give an error.

## Examples

```
SELECT TO_CHAR('1980-01-11 04:50:39', 'YYYY-MM-DD');
+-----+
| TO_CHAR('1980-01-11 04:50:39', 'YYYY-MM-DD') |
+-----+
| 1980-01-11 |
+-----+  
  
SELECT TO_CHAR('1980-01-11 04:50:39', 'HH24-MI-SS');
+-----+
| TO_CHAR('1980-01-11 04:50:39', 'HH24-MI-SS') |
+-----+
| 04-50-39 |
+-----+  
  
SELECT TO_CHAR('00-01-01 00:00:00', 'YY-MM-DD HH24:MI:SS');
+-----+
| TO_CHAR('00-01-01 00:00:00', 'YY-MM-DD HH24:MI:SS') |
+-----+
| 00-01-01 00:00:00 |
+-----+  
  
SELECT TO_CHAR('99-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS');
+-----+
| TO_CHAR('99-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS') |
+-----+
| 99-12-31 23:59:59 |
+-----+  
  
SELECT TO_CHAR('9999-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS');
+-----+
| TO_CHAR('9999-12-31 23:59:59', 'YY-MM-DD HH24:MI:SS') |
+-----+
| 99-12-31 23:59:59 |
+-----+  
  
SELECT TO_CHAR('21-01-03 08:30:00', 'Y-MONTH-DY HH:MI:SS');
+-----+
| TO_CHAR('21-01-03 08:30:00', 'Y-MONTH-DY HH:MI:SS') |
+-----+
| 1-January -Sun 08:30:00 |
+-----+
```

## See Also

- [SQL\\_MODE=ORACLE](#)

## 1.1.12.2.59 TRIM

### Syntax

```
TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str), TRIM([remstr FROM] str)
```

From [MariaDB 10.3.6](#)

```
TRIM_ORACLE([{BOTH | LEADING | TRAILING} [remstr] FROM] str), TRIM([remstr FROM] str)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the string

str  
with all  
remstr  
prefixes or suffixes removed. If none of the specifiers  
BOTH  
,

LEADING  
, or  
TRAILING  
is given,  
BOTH  
is assumed.  
remstr  
is optional and, if not specified, spaces are removed.

Returns NULL if given a NULL argument. If the result is empty, returns either an empty string, or, from [MariaDB 10.3.6](#) with [SQL\\_MODE=Oracle](#), NULL.

[SQL\\_MODE=Oracle](#)  
is not set by default.

The Oracle mode version of the function can be accessed in any mode by using

TRIM\_ORACLE  
as the function name.

### Examples

```
SELECT TRIM(' bar ')\\G
***** 1. row *****
TRIM(' bar '): bar

SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx')\\G
***** 1. row *****
TRIM(LEADING 'x' FROM 'xxxbarxxx'): barxxx

SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx')\\G
***** 1. row *****
TRIM(BOTH 'x' FROM 'xxxbarxxx'): bar

SELECT TRIM(TRAILING 'xyz' FROM 'barxyz')\\G
***** 1. row *****
TRIM(TRAILING 'xyz' FROM 'barxyz'): barx
```

From [MariaDB 10.3.6](#), with [SQL\\_MODE=Oracle](#) not set:

```
SELECT TRIM(''),TRIM_ORACLE('');
+-----+-----+
| TRIM('') | TRIM_ORACLE('') |
+-----+-----+
|       | NULL           |
+-----+-----+
```

From MariaDB 10.3.6 , with SQL\_MODE=Oracle set:

```
SELECT TRIM(''),TRIM_ORACLE('');
+-----+-----+
| TRIM('') | TRIM_ORACLE('') |
+-----+-----+
| NULL     | NULL           |
+-----+-----+
```

## See Also

- [LTRIM](#) - leading spaces removed
- [RTRIM](#) - trailing spaces removed

## 1.1.12.2.60 TRIM\_ORACLE

MariaDB starting with 10.3.6

TRIM\_ORACLE  
is a synonym for the [Oracle mode](#) version of the [TRIM function](#) , and is available in all modes.

## 1.1.12.2.61 UCASE

### Syntax

```
UCASE(str)
```

### Description

UCASE()  
is a synonym for

UPPER

()

## 1.1.12.2.62 UNCOMPRESS

### Syntax

```
UNCOMPRESS(string_to_uncompress)
```

### Description

Uncompresses a string compressed by the

[COMPRESS\(\)](#)

function. If the argument is not a compressed value, the result is

NULL

. This function requires MariaDB to have been compiled with a compression library such as zlib. Otherwise, the return value is always

NULL

. The [have\\_compress](#) server system variable indicates whether a compression library is present.

## Examples

```
SELECT UNCOMPRESS(COMPRESS('a string'));
+-----+
| UNCOMPRESS(COMPRESS('a string')) |
+-----+
| a string                         |
+-----+  
  
SELECT UNCOMPRESS('a string');
+-----+
| UNCOMPRESS('a string') |
+-----+
| NULL                      |
+-----+
```

## 1.1.12.2.63 UNCOMPRESSED\_LENGTH

### Syntax

```
UNCOMPRESSED_LENGTH(compressed_string)
```

### Description

Returns the length that the compressed string had before being compressed with

[COMPRESS\(\)](#)

UNCOMPRESSED\_LENGTH()  
returns  
NULL  
or an incorrect result if the string is not compressed.

Until [MariaDB 10.3.1](#) , returns

MySQL\_Type\_LONGLONG  
, or [bigint\(10\)](#) , in all cases. From [MariaDB 10.3.1](#) , returns  
MySQL\_Type\_LONG  
, or [int\(10\)](#) , when the result would fit within 32-bits.

## Examples

```
SELECT UNCOMPRESSED_LENGTH(COMPRESS(REPEAT('a',30)));
+-----+
| UNCOMPRESSED_LENGTH(COMPRESS(REPEAT('a',30))) |
+-----+
|                                         30 |
+-----+
```

## 1.1.12.2.64 UNHEX

### Syntax

```
UNHEX(str)
```

### Description

Performs the inverse operation of [HEX](#) (str). That is, it interprets each pair of hexadecimal digits in the argument as a number and converts it to the character represented by the number. The resulting characters are returned as a binary string.

If  
str

```
is  
NULL  
,  
UNHEX()  
returns  
NULL
```

## Examples

```
SELECT HEX('MariaDB');  
+-----+  
| HEX('MariaDB') |  
+-----+  
| 4D617269614442 |  
+-----+  
  
SELECT UNHEX('4D617269614442');  
+-----+  
| UNHEX('4D617269614442') |  
+-----+  
| MariaDB |  
+-----+  
  
SELECT 0x4D617269614442;  
+-----+  
| 0x4D617269614442 |  
+-----+  
| MariaDB |  
+-----+  
  
SELECT UNHEX(HEX('string'));  
+-----+  
| UNHEX(HEX('string')) |  
+-----+  
| string |  
+-----+  
  
SELECT HEX(UNHEX('1267'));  
+-----+  
| HEX(UNHEX('1267')) |  
+-----+  
| 1267 |  
+-----+
```

## See Also

- [Hexadecimal literals](#)
- [HEX\(\)](#)
- [CONV\(\)](#)

## 1.1.12.2.65 UPDATEXML

### Syntax

```
UpdateXML(xml_target, xpath_expr, new_xml)
```

### Description

This function replaces a single portion of a given fragment of XML markup

xml\_target  
with a new XML fragment  
new\_xml  
, and then returns the changed XML. The portion of  
xml\_target  
that is replaced matches an XPath expression  
xpath\_expr  
supplied by the user. If no expression matching

`xpath_expr`  
is found, or if multiple matches are found, the function returns the original  
`xml_target`  
XML fragment. All three arguments should be strings.

## Examples

```
SELECT
    UpdateXML('<a><b>ccc</b><d></d></a>', '/a', '<e>fff</e>') AS val1,
    UpdateXML('<a><b>ccc</b><d></d></a>', '/b', '<e>fff</e>') AS val2,
    UpdateXML('<a><b>ccc</b><d></d></a>', '//b', '<e>fff</e>') AS val3,
    UpdateXML('<a><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val4,
    UpdateXML('<a><d></d><b>ccc</b><d></d></a>', '/a/d', '<e>fff</e>') AS val5
    G
*****
1. row ****
val1: <e>fff</e>
val2: <a><b>ccc</b><d></d></a>
val3: <a><e>fff</e><d></d></a>
val4: <a><b>ccc</b><e>fff</e></a>
val5: <a><d></d><b>ccc</b><d></d></a>
1 row in set (0.00 sec)
```

## 1.1.12.2.66 UPPER

### Syntax

```
UPPER(str)
```

### Description

Returns the string

`str`

with all characters changed to uppercase according to the current character set mapping. The default is latin1 (cp1252 West European).

```
SELECT UPPER(surname), givenname FROM users ORDER BY surname;
+-----+-----+
| UPPER(surname) | givenname |
+-----+-----+
| ABEL          | Jacinto    |
| CASTRO        | Robert     |
| COSTA         | Phestos   |
| MOSCHELLA    | Hippolytos|
+-----+-----+
```

`UPPER()`

is ineffective when applied to binary strings (

`BINARY`

,

`VARBINARY`

,

`BLOB`

). The description of

`LOWER`

( )

shows how to perform lettercase conversion of binary strings.

## 1.1.12.2.67 WEIGHT\_STRING

# Syntax

```
WEIGHT_STRING(str [AS {CHAR|BINARY}(N)] [LEVEL levels] [flags])
  levels: N [ASC|DESC|REVERSE] [, N [ASC|DESC|REVERSE]] ...
```

## Description

Returns a binary string representing the string's sorting and comparison value. A string with a lower result means that for sorting purposes the string appears before a string with a higher result.

`WEIGHT_STRING()` is particularly useful when adding new collations, for testing purposes.

If

str

is a non-binary string ( `CHAR` , `VARCHAR` or `TEXT` ), `WEIGHT_STRING` returns the string's collation weight. If str

is a binary string ( `BINARY` , `VARBINARY` or `BLOB` ), the return value is simply the input value, since the weight for each byte in a binary string is the byte value.

`WEIGHT_STRING()` returns NULL if given a NULL input.

The optional AS clause permits casting the input string to a binary or non-binary string, as well as to a particular length.

AS BINARY(N) measures the length in bytes rather than characters, and right pads with 0x00 bytes to the desired length.

AS CHAR(N) measures the length in characters, and right pads with spaces to the desired length.

N has a minimum value of 1, and if it is less than the length of the input string, the string is truncated without warning.

The optional LEVEL clause specifies that the return value should contain weights for specific collation levels. The

levels

specifier can either be a single integer, a comma-separated list of integers, or a range of integers separated by a dash (whitespace is ignored). Integers can range from 1 to a maximum of 6, dependent on the collation, and need to be listed in ascending order.

If the LEVEL clause is not provided, a default of 1 to the maximum for the collation is assumed.

If the LEVEL is specified without using a range, an optional modifier is permitted.

ASC

, the default, returns the weights without any modification.

DESC

returns bitwise-inverted weights.

REVERSE

returns the weights in reverse order.

## Examples

The examples below use the `HEX()` function to represent non-printable results in hexadecimal format.

```

SELECT HEX(WEIGHT_STRING('x'));
+-----+
| HEX(WEIGHT_STRING('x')) |
+-----+
| 0058 |
+-----+

SELECT HEX(WEIGHT_STRING('x' AS BINARY(4)));
+-----+
| HEX(WEIGHT_STRING('x' AS BINARY(4))) |
+-----+
| 78000000 |
+-----+

SELECT HEX(WEIGHT_STRING('x' AS CHAR(4)));
+-----+
| HEX(WEIGHT_STRING('x' AS CHAR(4))) |
+-----+
| 0058002000200020 |
+-----+

SELECT HEX(WEIGHT_STRING(0xaa22ee LEVEL 1));
+-----+
| HEX(WEIGHT_STRING(0xaa22ee LEVEL 1)) |
+-----+
| AA22EE |
+-----+

SELECT HEX(WEIGHT_STRING(0xaa22ee LEVEL 1 DESC));
+-----+
| HEX(WEIGHT_STRING(0xaa22ee LEVEL 1 DESC)) |
+-----+
| 55DD11 |
+-----+

SELECT HEX(WEIGHT_STRING(0xaa22ee LEVEL 1 REVERSE));
+-----+
| HEX(WEIGHT_STRING(0xaa22ee LEVEL 1 REVERSE)) |
+-----+
| EE22AA |
+-----+

```

## 1.1.12.2.68 Type Conversion

### Contents

- 1. Rules for Conversion on Comparison
  - 1. Comparison Examples
- 2. Rules for Conversion on Dyadic Arithmetic Operations
  - 1. Arithmetic Examples

Implicit type conversion takes place when MariaDB is using operands of different types, in order to make the operands compatible.

It is best practice not to rely upon implicit conversion; rather use [CAST](#) to explicitly convert types.

### Rules for Conversion on Comparison

- If either argument is NULL, the result of the comparison is NULL unless the NULL-safe `<=>` equality comparison operator is used.
- If both arguments are integers, they are compared as integers.
- If both arguments are strings, they are compared as strings.
- If one argument is decimal and the other argument is decimal or integer, they are compared as decimals.
- If one argument is decimal and the other argument is a floating point, they are compared as floating point values.
- If a hexadecimal argument is not compared to a number, it is treated as a binary string.
- If a constant is compared to a TIMESTAMP or DATETIME, the constant is converted to a timestamp, unless used as an argument to the IN function.
- In other cases, arguments are compared as floating point, or real, numbers.

Note that if a string column is being compared with a numeric value, MariaDB will not use the index on the column, as there are numerous alternatives that may evaluate as equal (see examples below).

### Comparison Examples

Converting a string to a number:

```
SELECT 15+'15';
+-----+
| 15+'15' |
+-----+
|      30 |
+-----+
```

Converting a number to a string:

```
SELECT CONCAT(15, '15');
+-----+
| CONCAT(15, '15') |
+-----+
| 1515             |
+-----+
```

Floating point number errors:

```
SELECT '9746718491924563214' = 9746718491924563213;
+-----+
| '9746718491924563214' = 9746718491924563213 |
+-----+
|                                         1 |
+-----+
```

Numeric equivalence with strings:

```
SELECT '5' = 5;
+-----+
| '5' = 5 |
+-----+
|      1 |
+-----+

SELECT ' 5' = 5;
+-----+
| ' 5' = 5 |
+-----+
|      1 |
+-----+

SELECT ' 5  ' = 5;
+-----+
| ' 5  ' = 5 |
+-----+
|      1 |
+-----+
1 row in set, 1 warning (0.000 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message           |
+-----+-----+
| Note  | 1292 | Truncated incorrect DOUBLE value: ' 5  ' |
+-----+-----+
```

As a result of the above, MariaDB cannot use the index when comparing a string with a numeric value in the example below:

```

CREATE TABLE t (a VARCHAR(10), b VARCHAR(10), INDEX idx_a (a));

INSERT INTO t VALUES ('1', '1'), ('2', '2'), ('3', '3'), ('4', '4'), ('5', '5'), ('1', '5');

EXPLAIN SELECT * FROM t WHERE a = '3' 
***** 1. row *****
    id: 1
  select_type: SIMPLE
      table: t
        type: ref
possible_keys: idx_a
      key: idx_a
     key_len: 13
       ref: const
      rows: 1
  Extra: Using index condition

EXPLAIN SELECT * FROM t WHERE a = 3 
***** 1. row *****
    id: 1
  select_type: SIMPLE
      table: t
        type: ALL
possible_keys: idx_a
      key: NULL
     key_len: NULL
       ref: NULL
      rows: 6
  Extra: Using where

```

## Rules for Conversion on Dyadic Arithmetic Operations

Implicit type conversion also takes place on dyadic arithmetic operations ( `+` , `-` , `*` , `/` ). MariaDB chooses the minimum data type that is guaranteed to fit the result and converts both arguments to the result data type.

For [addition \(+\)](#) , [subtraction \(-\)](#) and [multiplication \(\\*\)](#) , the result data type is chosen as follows:

- If either of the arguments is an approximate number (float, double), the result is double.
- If either of the arguments is a string (char, varchar, text), the result is double.
- If either of the arguments is a decimal number, the result is decimal.
- If either of the arguments is of a temporal type with a non-zero fractional second precision (time(N), datetime(N), timestamp(N)), the result is decimal.
- If either of the arguments is of a temporal type with a zero fractional second precision (time(0), date, datetime(0), timestamp(0)), the result may vary between int, int unsigned, bigint or bigint unsigned, depending on the exact data type combination.
- If both arguments are integer numbers (tinyint, smallint, mediumint, bigint), the result may vary between int, int unsigned, bigint or bigint unsigned, depending of the exact data types and their signs.

For [division \(/\)](#) , the result data type is chosen as follows:

- If either of the arguments is an approximate number (float, double), the result is double.
- If either of the arguments is a string (char, varchar, text), the result is double.
- Otherwise, the result is decimal.

## Arithmetic Examples

Note, the above rules mean that when an argument of a temporal data type appears in addition or subtraction, it's treated as a number by default.

```

SELECT TIME'10:20:30' + 1;
+-----+
| TIME'10:20:30' + 1 |
+-----+
|      102031 |
+-----+

```

In order to do temporal addition or subtraction instead, use the [DATE\\_ADD\(\)](#) or [DATE\\_SUB\(\)](#) functions, or an [INTERVAL](#) expression as the second argument:

```

SELECT TIME'10:20:30' + INTERVAL 1 SECOND;
+-----+
| TIME'10:20:30' + INTERVAL 1 SECOND |
+-----+
| 10:20:31 |
+-----+

```

```

SELECT "2.2" + 3;
+-----+
| "2.2" + 3 |
+-----+
| 5.2 |
+-----+

SELECT 2.2 + 3;
+-----+
| 2.2 + 3 |
+-----+
| 5.2 |
+-----+

SELECT 2.2 / 3;
+-----+
| 2.2 / 3 |
+-----+
| 0.73333 |
+-----+

SELECT "2.2" / 3;
+-----+
| "2.2" / 3 |
+-----+
| 0.733333333333334 |
+-----+

```

## 1.1.12.3 Date and Time Functions

### 1.1.12.3.1 Microseconds in MariaDB

#### Contents

1. Additional Information
2. MySQL 5.6 Microseconds
3. See Also

The `TIME`, `DATETIME`, and `TIMESTAMP` types, along with the temporal functions, `CAST` and `dynamic columns`, support microseconds. The datetime precision of a column can be specified when creating the table with `CREATE TABLE`, for example:

```

CREATE TABLE example(
    col_microsec DATETIME(6),
    col_millisec TIME(3)
);

```

Generally, the precision can be specified for any

```

TIME
,
DATETIME
, or
TIMESTAMP

```

column, in parentheses, after the type name. The datetime precision specifies number of digits after the decimal dot and can be any integer number from 0 to 6. If no precision is specified it is assumed to be 0, for backward compatibility reasons.

A datetime precision can be specified wherever a type name is used. For example:

- when declaring arguments of stored routines.
- when specifying a return type of a stored function.
- when declaring variables.
- in a

```

CAST
function:

```

```

create function example(x datetime(5)) returns time(4)
begin
    declare y timestamp(6);
    return cast(x as time(2));
end;

```

`%f`

is used as the formatting option for microseconds in the `STR_TO_DATE`, `DATE_FORMAT` and `FROM_UNIXTIME` functions, for example:

```

SELECT STR_TO_DATE('20200809 020917076','%Y%m%d %H%i%s%f');
+-----+
| STR_TO_DATE('20200809 020917076','%Y%m%d %H%i%s%f') |
+-----+
| 2020-08-09 02:09:17.076000 |
+-----+

```

## Additional Information

- when comparing anything to a temporal value (

DATETIME  
,  
TIME  
,  
  
DATE  
  
, or  
TIMESTAMP

), both values are compared as temporal values, not as strings.

- The [INFORMATION\\_SCHEMA.COLUMNS table](#) has a new column

DATETIME\_PRECISION

- [NOW\(\)](#) , [CURTIME\(\)](#) , [UTC\\_TIMESTAMP\(\)](#) , [UTC\\_TIME\(\)](#) , [CURRENT\\_TIME\(\)](#) , [CURRENT\\_TIMESTAMP\(\)](#) , [LOCALTIME\(\)](#) and [LOCALTIMESTAMP\(\)](#) now accept datetime precision as an optional argument. For example:

```

SELECT CURTIME(4);
--> 10:11:12.3456

```

- [TIME\\_TO\\_SEC\(\)](#) and [UNIX\\_TIMESTAMP\(\)](#) preserve microseconds of the argument. These functions will return a [decimal](#) number if the result non-zero datetime precision and an [integer](#) otherwise (for backward compatibility).

```

SELECT TIME_TO_SEC('10:10:10.12345');
--> 36610.12345

```

- Current versions of this patch fix a bug in the following optimization: in certain queries with

DISTINCT

MariaDB can ignore this clause if it can prove that all result rows are unique anyway, for example, when a primary key is compared with a constant. Sometimes this optimization was applied incorrectly, though — for example, when comparing a string with a date constant. This is now fixed.

- 

DATE\_ADD()  
and  
DATE\_SUB()  
functions can now take a  
TIME  
expression as an argument (not just  
DATETIME  
as before).

```

SELECT TIME('10:10:10') + INTERVAL 100 MICROSECOND;
--> 10:10:10.000100

```

- The

event\_time  
field in the [mysql.general\\_log](#) table and the  
start\_time  
,  
query\_time  
, and  
lock\_time  
fields in the [mysql.slow\\_log](#) table now store values with microsecond precision.

- This patch fixed a bug when comparing a temporal value using the

BETWEEN  
operator and one of the operands is  
NULL

- The old syntax

TIMESTAMP(N)  
, where  
N

is the display width, is no longer supported. It was deprecated in MySQL 4.1.0 (released on 2003-04-03).

- when a

DATETIME  
value is compared to a  
TIME  
value, the latter is treated as a full datetime with a zero date part, similar to comparing  
DATE  
to a  
DATETIME  
, or to comparing  
DECIMAL

numbers. Earlier versions of MariaDB used to compare only the time part of both operands in such a case.

- In MariaDB, an extra column

TIME\_MS

has been added to the

INFORMATION\_SCHEMA.PROCESSLIST

table, as well as to the output of

SHOW FULL PROCESSLIST

**Note:** When you convert a temporal value to a value with a smaller precision, it will be truncated, not rounded. This is done to guarantee that the date part is not changed. For example:

```
SELECT CAST('2009-12-31 23:59:59.998877' as DATETIME(3));
-> 2009-12-31 23:59:59.998
```

## MySQL 5.6 Microseconds

MySQL 5.6 introduced microseconds using a slightly different implementation to [MariaDB 5.3](#). Since [MariaDB 10.1](#), MariaDB has defaulted to the MySQL format, by means of the `--mysql56-temporal-format` variable. The MySQL version requires slightly [more storage](#) but has some advantages in permitting the eventual support of negative dates, and in replication.

### See Also

- [Data Type Storage Requirements](#)

### 1.1.12.3.2 Date and Time Units

The

INTERVAL

keyword can be used to add or subtract a time interval of time to a

DATETIME

,

DATE

or

TIME

value.

The syntax is:

```
INTERVAL time_quantity time_unit
```

For example, the

SECOND

unit is used below by the

## DATE\_ADD()

function:

```
SELECT '2008-12-31 23:59:59' + INTERVAL 1 SECOND;
+-----+
| '2008-12-31 23:59:59' + INTERVAL 1 SECOND |
+-----+
| 2009-01-01 00:00:00 |
+-----+
```

The following units are valid:

Unit	Description
MICROSECOND	Microseconds
SECOND	Seconds
MINUTE	Minutes
HOUR	Hours
DAY	Days
WEEK	Weeks
MONTH	Months
QUARTER	Quarters
YEAR	Years
SECOND_MICROSECOND	Seconds.Microseconds
MINUTE_MICROSECOND	Minutes.Seconds.Microseconds
MINUTE_SECOND	Minutes.Seconds
HOUR_MICROSECOND	Hours.Minutes.Seconds.Microseconds
HOUR_SECOND	Hours.Minutes.Seconds
HOUR_MINUTE	Hours.Minutes
DAY_MICROSECOND	Days Hours.Minutes.Seconds.Microseconds

DAY_SECOND	Days Hours.Minutes.Seconds
DAY_MINUTE	Days Hours.Minutes
DAY_HOUR	Days Hours
YEAR_MONTH	Years-Months

The time units containing an underscore are composite; that is, they consist of multiple base time units. For base time units,

`time_quantity`

is an integer number. For composite units, the quantity must be expressed as a string with multiple integer numbers separated by any punctuation character.

Example of composite units:

```
INTERVAL '2:2' YEAR_MONTH
INTERVAL '1:30:30' HOUR_SECOND
INTERVAL '1!30!30' HOUR_SECOND -- same as above
```

Time units can be used in the following contexts:

- after a

`+`

or a

`-`

operator;

- with the following

`DATE`

or

`TIME`

functions:

`ADDDATE()`

`SUBDATE()`

`DATE_ADD()`

`DATE_SUB()`

`TIMESTAMPADD()`

`TIMESTAMPDIFF()`

`EXTRACT()`

`;`

- in the

`ON SCHEDULE`  
clause of

[CREATE EVENT](#)

and

[ALTER EVENT](#)

- when defining a [partitioning](#)  
BY SYSTEM\_TIME

## See also

- [Date and time literals](#)

### 1.1.12.3.3 ADD\_MONTHS

MariaDB starting with [10.6.1](#)

The ADD\_MONTHS function was introduced in [MariaDB 10.6.1](#) to enhance Oracle compatibility. Similar functionality can be achieved with the [DATE\\_ADD](#) function.

## Syntax

```
ADD_MONTHS(date, months)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

`ADD_MONTHS`

adds an integer *months* to a given *date* ([DATE](#), [DATETIME](#) or [TIMESTAMP](#)), returning the resulting date.

*months* can be positive or negative.

The resulting day component will remain the same as that specified in *date* , unless the resulting month has fewer days than the day component of the given date, in which case the day will be the last day of the resulting month.

Returns NULL if given an invalid date, or a NULL argument.

## Examples

```

SELECT ADD_MONTHS('2012-01-31', 2);
+-----+
| ADD_MONTHS('2012-01-31', 2) |
+-----+
| 2012-03-31 |
+-----+

SELECT ADD_MONTHS('2012-01-31', -5);
+-----+
| ADD_MONTHS('2012-01-31', -5) |
+-----+
| 2011-08-31 |
+-----+

SELECT ADD_MONTHS('2011-01-31', 1);
+-----+
| ADD_MONTHS('2011-01-31', 1) |
+-----+
| 2011-02-28 |
+-----+

SELECT ADD_MONTHS('2012-01-31', 1);
+-----+
| ADD_MONTHS('2012-01-31', 1) |
+-----+
| 2012-02-29 |
+-----+

SELECT ADD_MONTHS('2012-01-31', 2);
+-----+
| ADD_MONTHS('2012-01-31', 2) |
+-----+
| 2012-03-31 |
+-----+

SELECT ADD_MONTHS('2012-01-31', 3);
+-----+
| ADD_MONTHS('2012-01-31', 3) |
+-----+
| 2012-04-30 |
+-----+

```

## See Also

- [SQL\\_MODE=ORACLE](#)

### 1.1.12.3.4 ADDDATE

## Syntax

```
ADDDATE(date,INTERVAL expr unit), ADDDATE(expr,days)
```

## Description

When invoked with the

INTERVAL  
form of the second argument,  
ADDDATE()  
is a synonym for

[DATE\\_ADD\(\)](#)

. The related function

[SUBDATE\(\)](#)

is a synonym for

[DATE\\_SUB\(\)](#)

. For information on the  
INTERVAL  
unit argument, see the discussion for

[DATE\\_ADD\(\)](#)

When invoked with the days form of the second argument, MariaDB treats it as an integer number of days to be added to *expr*.

## Examples

```
SELECT DATE_ADD('2008-01-02', INTERVAL 31 DAY);
+-----+
| DATE_ADD('2008-01-02', INTERVAL 31 DAY) |
+-----+
| 2008-02-02                                |
+-----+
```

```
SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY);
+-----+
| ADDDATE('2008-01-02', INTERVAL 31 DAY) |
+-----+
| 2008-02-02                                |
+-----+
```

```
SELECT ADDDATE('2008-01-02', 31);
+-----+
| ADDDATE('2008-01-02', 31) |
+-----+
| 2008-02-02                                |
+-----+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d, ADDDATE(d, 10) from t1;
+-----+
| d           | ADDDATE(d, 10)   |
+-----+
| 2007-01-30 21:31:07 | 2007-02-09 21:31:07 |
| 1983-10-15 06:42:51 | 1983-10-25 06:42:51 |
| 2011-04-21 12:34:56 | 2011-05-01 12:34:56 |
| 2011-10-30 06:31:41 | 2011-11-09 06:31:41 |
| 2011-01-30 14:03:25 | 2011-02-09 14:03:25 |
| 2004-10-07 11:19:34 | 2004-10-17 11:19:34 |
+-----+
```

```
SELECT d, ADDDATE(d, INTERVAL 10 HOUR) from t1;
+-----+
| d           | ADDDATE(d, INTERVAL 10 HOUR) |
+-----+
| 2007-01-30 21:31:07 | 2007-01-31 07:31:07 |
| 1983-10-15 06:42:51 | 1983-10-15 16:42:51 |
| 2011-04-21 12:34:56 | 2011-04-21 22:34:56 |
| 2011-10-30 06:31:41 | 2011-10-30 16:31:41 |
| 2011-01-30 14:03:25 | 2011-01-31 00:03:25 |
| 2004-10-07 11:19:34 | 2004-10-07 21:19:34 |
+-----+
```

## 1.1.12.3.5 ADDTIME

# Syntax

```
ADDTIME(expr1,expr2)
```

## Description

ADDTIME()

adds *expr2* to *expr1* and returns the result. *expr1* is a time or datetime expression, and *expr2* is a time expression.

## Examples

```
SELECT ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002');
+-----+
| ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002') |
+-----+
| 2008-01-02 01:01:01.000001 |
+-----+

SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');
+-----+
| ADDTIME('01:00:00.999999', '02:00:00.999998') |
+-----+
| 03:00:01.999997 |
+-----+
```

## 1.1.12.3.6 CONVERT\_TZ

### Syntax

```
CONVERT_TZ(dt,from_tz,to_tz)
```

## Description

CONVERT\_TZ() converts a datetime value *dt* from the [time zone](#) given by *from\_tz* to the time zone given by *to\_tz* and returns the resulting value.

In order to use named time zones, such as GMT, MET or Africa/Johannesburg, the `time_zone` tables must be loaded (see [mysql\\_tzinfo\\_to\\_sql](#) ).

No conversion will take place if the value falls outside of the supported TIMESTAMP range ('1970-01-01 00:00:01' to '2038-01-19 05:14:07' UTC) when converted from *from\_tz* to UTC.

This function returns NULL if the arguments are invalid (or named time zones have not been loaded).

See [time zones](#) for more information.

## Examples

```
SELECT CONVERT_TZ('2016-01-01 12:00:00','+00:00','+10:00');
+-----+
| CONVERT_TZ('2016-01-01 12:00:00','+00:00','+10:00') |
+-----+
| 2016-01-01 22:00:00 |
+-----+
```

Using named time zones (with the time zone tables loaded):

```
SELECT CONVERT_TZ('2016-01-01 12:00:00','GMT','Africa/Johannesburg');
+-----+
| CONVERT_TZ('2016-01-01 12:00:00','GMT','Africa/Johannesburg') |
+-----+
| 2016-01-01 14:00:00 |
+-----+
```

The value is out of the TIMESTAMP range, so no conversion takes place:

```
SELECT CONVERT_TZ('1969-12-31 22:00:00','+00:00','+10:00');
+-----+
| CONVERT_TZ('1969-12-31 22:00:00','+00:00','+10:00') |
+-----+
| 1969-12-31 22:00:00 |
+-----+
```

## 1.1.12.3.7 CURDATE

### Syntax

```
CURDATE()
CURRENT_DATE
CURRENT_DATE()
```

### Description

CURDATE  
returns the current date as a value in 'YYYY-MM-DD' or YYYYMMDD format, depending on whether the function is used in a string or numeric context.

CURRENT\_DATE  
and  
CURRENT\_DATE()  
are synonyms.

### Examples

```
SELECT CURDATE();
+-----+
| CURDATE() |
+-----+
| 2019-03-05 |
+-----+
```

In a numeric context (note this is not performing date calculations):

```
SELECT CURDATE() +0;
+-----+
| CURDATE() +0 |
+-----+
| 20190305 |
+-----+
```

Data calculation:

```
SELECT CURDATE() - INTERVAL 5 DAY;
+-----+
| CURDATE() - INTERVAL 5 DAY |
+-----+
| 2019-02-28 |
+-----+
```

## 1.1.12.3.8 CURRENT\_DATE

### Syntax

```
CURRENT_DATE, CURRENT_DATE()
```

### Description

`CURRENT_DATE`  
and  
`CURRENT_DATE()`  
are synonyms for [CURDATE\(\)](#).

## 1.1.12.3.9 CURRENT\_TIME

### Syntax

```
CURRENT_TIME  
CURRENT_TIME([precision])
```

### Description

`CURRENT_TIME`  
and  
`CURRENT_TIME()`  
are synonyms for  
  
[CURTIME\(\)](#)

### See Also

- [Microseconds in MariaDB](#)

## 1.1.12.3.10 CURRENT\_TIMESTAMP

### Syntax

```
CURRENT_TIMESTAMP  
CURRENT_TIMESTAMP([precision])
```

### Description

`CURRENT_TIMESTAMP`  
and  
`CURRENT_TIMESTAMP()`  
are synonyms for  
  
[NOW\(\)](#)

### See Also

- [Microseconds in MariaDB](#)
- The [TIMESTAMP](#) data type

## 1.1.12.3.11 CURTIME

### Syntax

```
CURTIME([precision])
```

### Description

Returns the current time as a value in 'HH:MM:SS' or HHMMSS.ududu format, depending on whether the function is used in a string or numeric context. The value is expressed in the current [time zone](#).

The optional *precision* determines the microsecond precision. See [Microseconds in MariaDB](#).

## Examples

```
SELECT CURTIME();
+-----+
| CURTIME() |
+-----+
| 12:45:39 |
+-----+

SELECT CURTIME() + 0;
+-----+
| CURTIME() + 0 |
+-----+
| 124545.00000 |
+-----+
```

With precision:

```
SELECT CURTIME(2);
+-----+
| CURTIME(2) |
+-----+
| 09:49:08.09 |
+-----+
```

## See Also

- [Microseconds in MariaDB](#)

### 1.1.12.3.12 DATE FUNCTION

#### Syntax

```
DATE(expr)
```

#### Description

Extracts the date part of the date or datetime expression expr.

## Examples

```
SELECT DATE('2013-07-18 12:21:32');
+-----+
| DATE('2013-07-18 12:21:32') |
+-----+
| 2013-07-18 |
+-----+
```

#### Error Handling

Until [MariaDB 5.5.32](#), some versions of MariaDB returned  
0000-00-00  
when passed an invalid date. From 5.5.32, NULL is returned.

### 1.1.12.3.13 DATEDIFF

#### Syntax

```
DATEDIFF(expr1,expr2)
```

## Description

DATEDIFF()

returns (*expr1 – expr2*) expressed as a value in days from one date to the other. *expr1* and *expr2* are date or date-and-time expressions. Only the date parts of the values are used in the calculation.

## Examples

```
SELECT DATEDIFF('2007-12-31 23:59:59','2007-12-30');
+-----+
| DATEDIFF('2007-12-31 23:59:59','2007-12-30') |
+-----+
| 1 |
+-----+  
  
SELECT DATEDIFF('2010-11-30 23:59:59','2010-12-31');
+-----+
| DATEDIFF('2010-11-30 23:59:59','2010-12-31') |
+-----+
| -31 |
+-----+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT NOW();
+-----+
| NOW() |
+-----+
| 2011-05-23 10:56:05 |
+-----+  
  
SELECT d, DATEDIFF(NOW(),d) FROM t1;
+-----+
| d | DATEDIFF(NOW(),d) |
+-----+
| 2007-01-30 21:31:07 | 1574 |
| 1983-10-15 06:42:51 | 10082 |
| 2011-04-21 12:34:56 | 32 |
| 2011-10-30 06:31:41 | -160 |
| 2011-01-30 14:03:25 | 113 |
| 2004-10-07 11:19:34 | 2419 |
+-----+
```

## 1.1.12.3.14 DATE\_ADD

### Syntax

```
DATE_ADD(date,INTERVAL expr unit)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

# Description

Performs date arithmetic. The *date* argument specifies the starting date or datetime value. *expr* is an expression specifying the interval value to be added or subtracted from the starting date. *expr* is a string; it may start with a "

" for negative intervals. *unit* is a keyword indicating the units in which the expression should be interpreted. See [Date and Time Units](#) for a complete list of permitted units.

## Examples

```
SELECT '2008-12-31 23:59:59' + INTERVAL 1 SECOND;
+-----+
| '2008-12-31 23:59:59' + INTERVAL 1 SECOND |
+-----+
| 2009-01-01 00:00:00 |
+-----+
```

```
SELECT INTERVAL 1 DAY + '2008-12-31';
+-----+
| INTERVAL 1 DAY + '2008-12-31' |
+-----+
| 2009-01-01 |
+-----+
```

```
SELECT '2005-01-01' - INTERVAL 1 SECOND;
+-----+
| '2005-01-01' - INTERVAL 1 SECOND |
+-----+
| 2004-12-31 23:59:59 |
+-----+
```

```
SELECT DATE_ADD('2000-12-31 23:59:59', INTERVAL 1 SECOND);
+-----+
| DATE_ADD('2000-12-31 23:59:59', INTERVAL 1 SECOND) |
+-----+
| 2001-01-01 00:00:00 |
+-----+
```

```
SELECT DATE_ADD('2010-12-31 23:59:59', INTERVAL 1 DAY);
+-----+
| DATE_ADD('2010-12-31 23:59:59', INTERVAL 1 DAY) |
+-----+
| 2011-01-01 23:59:59 |
+-----+
```

```
SELECT DATE_ADD('2100-12-31 23:59:59', INTERVAL '1:1' MINUTE_SECOND);
+-----+
| DATE_ADD('2100-12-31 23:59:59', INTERVAL '1:1' MINUTE_SECOND) |
+-----+
| 2101-01-01 00:01:00 |
+-----+
```

```
SELECT DATE_ADD('1900-01-01 00:00:00', INTERVAL '-1 10' DAY_HOUR);
+-----+
| DATE_ADD('1900-01-01 00:00:00', INTERVAL '-1 10' DAY_HOUR) |
+-----+
| 1899-12-30 14:00:00 |
+-----+
```

```
SELECT DATE_ADD('1992-12-31 23:59:59.000002', INTERVAL '1.999999' SECOND_MICROSECOND);
+-----+
| DATE_ADD('1992-12-31 23:59:59.000002', INTERVAL '1.999999' SECOND_MICROSECOND) |
+-----+
| 1993-01-01 00:00:01.000001 |
+-----+
```

## See Also

- [DATE\\_SUB](#)
- [ADD\\_MONTHS](#)

## 1.1.12.3.15 DATE\_FORMAT

### Syntax

```
DATE_FORMAT(date, format[, locale])
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Formats the date value according to the format string.

The language used for the names is controlled by the value of the [lc\\_time\\_names](#) system variable. See [server locale](#) for more on the supported locales.

The options that can be used by DATE\_FORMAT(), as well as its inverse [STR\\_TO\\_DATE\(\)](#) and the [FROM\\_UNIXTIME\(\)](#) function, are:

Option	Description
%a	Short weekday name in current locale (Variable <a href="#">lc_time_names</a> ).
%b	Short form month name in current locale. For locale en_US this is one of: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov or Dec.
%c	Month with 1 or 2 digits.
%d	Day with English suffix 'th', 'nd', 'st' or 'rd'. (1st, 2nd, 3rd...).
%d	Day with 2 digits.
%e	Day with 1 or 2 digits.
%f	<a href="#">Microseconds</a> 6 digits.
%H	Hour with 2 digits between 00-23.
%h	Hour with 2 digits between 01-12.
%I	Hour with 2 digits between 01-12.
%i	Minute with 2 digits.
%j	Day of the year (001-366)

%k	Hour with 1 digits between 0-23.
%l	Hour with 1 digits between 1-12.
%M	Full month name in current locale (Variable <a href="#">lc_time_names</a> ).
%m	Month with 2 digits.
%p	AM/PM according to current locale (Variable <a href="#">lc_time_names</a> ).
%r	Time in 12 hour format, followed by AM/PM. Short for '%l:%i:%S %p'.
%S	Seconds with 2 digits.
%s	Seconds with 2 digits.
%T	Time in 24 hour format. Short for '%H:%i:%S'.
%U	Week number (00-53), when first day of the week is Sunday.
%u	Week number (00-53), when first day of the week is Monday.
%v	Week number (01-53), when first day of the week is Sunday. Used with %X.
%v	Week number (01-53), when first day of the week is Monday. Used with %x.
%w	Full weekday name in current locale (Variable <a href="#">lc_time_names</a> ).
%w	Day of the week. 0 = Sunday, 6 = Saturday.
%x	Year with 4 digits when first day of the week is Sunday. Used with %V.
%x	Year with 4 digits when first day of the week is Monday. Used with %v.
%y	Year with 4 digits.
%y	Year with 2 digits.

	%# For <code>str_to_date</code> (), skip all numbers.
%.	For <code>str_to_date</code> (), skip all punctuation characters.
%@	For <code>str_to_date</code> (), skip all alpha characters.
%%	A literal % character.

To get a date in one of the standard formats,

`GET_FORMAT()`

can be used.

## Examples

```
SELECT DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y');
+-----+
| DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y') |
+-----+
| Sunday October 2009 |
+-----+


SELECT DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s');
+-----+
| DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s') |
+-----+
| 22:23:00 |
+-----+


SELECT DATE_FORMAT('1900-10-04 22:23:00', '%D %y %a %d %m %b %j');
+-----+
| DATE_FORMAT('1900-10-04 22:23:00', '%D %y %a %d %m %b %j') |
+-----+
| 4th 00 Thu 04 10 Oct 277 |
+-----+


SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w');
+-----+
| DATE_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w') |
+-----+
| 22 22 10 10:23:00 PM 22:23:00 00 6 |
+-----+


SELECT DATE_FORMAT('1999-01-01', '%X %V');
+-----+
| DATE_FORMAT('1999-01-01', '%X %V') |
+-----+
| 1998 52 |
+-----+


SELECT DATE_FORMAT('2006-06-00', '%d');
+-----+
| DATE_FORMAT('2006-06-00', '%d') |
+-----+
| 00 |
+-----+
```

MariaDB starting with 10.3.2

Optionally, the locale can be explicitly specified as the third `DATE_FORMAT()` argument. Doing so makes the function independent from the session settings, and the three argument version of `DATE_FORMAT()` can be used in virtual indexed and persistent [generated-columns](#) :

```
SELECT DATE_FORMAT('2006-01-01', '%W', 'el_GR');
+-----+
| DATE_FORMAT('2006-01-01', '%W', 'el_GR') |
+-----+
| Κυριακή |
+-----+
```

## See Also

- [STR\\_TO\\_DATE\(\)](#)
- [FROM\\_UNIXTIME\(\)](#)

## 1.1.12.3.16 DATE\_SUB

### Syntax

```
DATE_SUB(date, INTERVAL expr unit)
```

### Description

Performs date arithmetic. The *date* argument specifies the starting date or datetime value. *expr* is an expression specifying the interval value to be added or subtracted from the starting date. *expr* is a string; it may start with a "

" for negative intervals. *unit* is a keyword indicating the units in which the expression should be interpreted. See [Date and Time Units](#) for a complete list of permitted units.

See also

[DATE\\_ADD\(\)](#)

### Examples

```
SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
+-----+
| DATE_SUB('1998-01-02', INTERVAL 31 DAY) |
+-----+
| 1997-12-02 |
+-----+
```

```
SELECT DATE_SUB('2005-01-01 00:00:00', INTERVAL '1 1:1:1' DAY_SECOND);
+-----+
| DATE_SUB('2005-01-01 00:00:00', INTERVAL '1 1:1:1' DAY_SECOND) |
+-----+
| 2004-12-30 22:58:59 |
+-----+
```

## 1.1.12.3.17 DAY

### Syntax

```
DAY(date)
```

### Description

DAY()  
is a synonym for

[DAYOFMONTH\(\)](#)

## 1.1.12.3.18 DAYNAME

### Syntax

```
DAYNAME(date)
```

### Description

Returns the name of the weekday for date. The language used for the name is controlled by the value of the [lc\\_time\\_names](#) system variable. See [server locale](#) for more on the supported locales.

### Examples

```
SELECT DAYNAME('2007-02-03');
+-----+
| DAYNAME('2007-02-03') |
+-----+
| Saturday           |
+-----+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
  ("2007-01-30 21:31:07"),
  ("1983-10-15 06:42:51"),
  ("2011-04-21 12:34:56"),
  ("2011-10-30 06:31:41"),
  ("2011-01-30 14:03:25"),
  ("2004-10-07 11:19:34");
```

```
SELECT d, DAYNAME(d) FROM t1;
+-----+-----+
| d          | DAYNAME(d) |
+-----+-----+
| 2007-01-30 21:31:07 | Tuesday    |
| 1983-10-15 06:42:51 | Saturday   |
| 2011-04-21 12:34:56 | Thursday   |
| 2011-10-30 06:31:41 | Sunday     |
| 2011-01-30 14:03:25 | Sunday     |
| 2004-10-07 11:19:34 | Thursday   |
+-----+-----+
```

Changing the locale:

```
SET lc_time_names = 'fr_CA';

SELECT DAYNAME('2013-04-01');
+-----+
| DAYNAME('2013-04-01') |
+-----+
| lundi                 |
+-----+
```

## 1.1.12.3.19 DAYOFMONTH

### Syntax

```
DAYOFMONTH(date)
```

### Description

Returns the day of the month for date, in the range

```
1  
to  
31  
, or  
0  
for dates such as  
'0000-00-00'  
or  
'2008-00-00'  
which have a zero day part.
```

DAY() is a synonym.

## Examples

```
SELECT DAYOFMONTH('2007-02-03');  
+-----+  
| DAYOFMONTH('2007-02-03') |  
+-----+  
| 3 |  
+-----+
```

```
CREATE TABLE t1 (d DATETIME);  
INSERT INTO t1 VALUES  
    ("2007-01-30 21:31:07"),  
    ("1983-10-15 06:42:51"),  
    ("2011-04-21 12:34:56"),  
    ("2011-10-30 06:31:41"),  
    ("2011-01-30 14:03:25"),  
    ("2004-10-07 11:19:34");
```

```
SELECT d FROM t1 WHERE DAYOFMONTH(d) = 30;  
+-----+  
| d |  
+-----+  
| 2007-01-30 21:31:07 |  
| 2011-10-30 06:31:41 |  
| 2011-01-30 14:03:25 |  
+-----+
```

## 1.1.12.3.20 DAYOFWEEK

### Syntax

```
DAYOFWEEK(date)
```

### Description

Returns the day of the week index for the date (1 = Sunday, 2 = Monday, ..., 7 = Saturday). These index values correspond to the ODBC standard.

This contrasts with

[WEEKDAY\(\)](#)

which follows a different index numbering (

```
0  
= Monday,  
1  
= Tuesday, ...  
6  
= Sunday).
```

## Examples

```
SELECT DAYOFWEEK('2007-02-03');
+-----+
| DAYOFWEEK('2007-02-03') |
+-----+
| 7 |
+-----+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d, DAYNAME(d), DAYOFWEEK(d), WEEKDAY(d) from t1;
+-----+-----+-----+
| d | DAYNAME(d) | DAYOFWEEK(d) | WEEKDAY(d) |
+-----+-----+-----+
| 2007-01-30 21:31:07 | Tuesday | 3 | 1 |
| 1983-10-15 06:42:51 | Saturday | 7 | 5 |
| 2011-04-21 12:34:56 | Thursday | 5 | 3 |
| 2011-10-30 06:31:41 | Sunday | 1 | 6 |
| 2011-01-30 14:03:25 | Sunday | 1 | 6 |
| 2004-10-07 11:19:34 | Thursday | 5 | 3 |
+-----+-----+-----+
```

## 1.1.12.3.21 DAYOFYEAR

### Syntax

```
DAYOFYEAR(date)
```

### Description

Returns the day of the year for date, in the range 1 to 366.

### Examples

```
SELECT DAYOFYEAR('2018-02-16');
+-----+
| DAYOFYEAR('2018-02-16') |
+-----+
| 47 |
+-----+
```

## 1.1.12.3.22 EXTRACT

### Syntax

```
EXTRACT(unit FROM date)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

The EXTRACT() function extracts the required unit from the date. See [Date and Time Units](#) for a complete list of permitted units.

In MariaDB 10.0.7 and MariaDB 5.5.35 ,

EXTRACT (HOUR FROM ...)

was changed to return a value from 0 to 23, adhering to the SQL standard. Until MariaDB 10.0.6 and MariaDB 5.5.34 , and in all versions of MySQL at least as of MySQL 5.7, it could return a value > 23. HOUR() is not a standard function, so continues to adhere to the old behaviour inherited from MySQL.

## Examples

```
SELECT EXTRACT(YEAR FROM '2009-07-02');
+-----+
| EXTRACT(YEAR FROM '2009-07-02') |
+-----+
|          2009 |
+-----+  
  
SELECT EXTRACT(YEAR_MONTH FROM '2009-07-02 01:02:03');
+-----+
| EXTRACT(YEAR_MONTH FROM '2009-07-02 01:02:03') |
+-----+
|          200907 |
+-----+  
  
SELECT EXTRACT(DAY_MINUTE FROM '2009-07-02 01:02:03');
+-----+
| EXTRACT(DAY_MINUTE FROM '2009-07-02 01:02:03') |
+-----+
|          20102 |
+-----+  
  
SELECT EXTRACT(MICROSECOND FROM '2003-01-02 10:30:00.000123');
+-----+
| EXTRACT(MICROSECOND FROM '2003-01-02 10:30:00.000123') |
+-----+
|          123 |
+-----+
```

From MariaDB 10.0.7 and MariaDB 5.5.35 ,

EXTRACT (HOUR FROM...)

returns a value from 0 to 23, as per the SQL standard.

HOUR

is not a standard function, so continues to adhere to the old behaviour inherited from MySQL.

```
SELECT EXTRACT(HOUR FROM '26:30:00'), HOUR('26:30:00');
+-----+-----+
| EXTRACT(HOUR FROM '26:30:00') | HOUR('26:30:00') |
+-----+-----+
|          2 |          26 |
+-----+-----+
```

## See Also

- [Date and Time Units](#)
- [Date and Time Literals](#)
- [HOUR\(\)](#)

## 1.1.12.3.23 FROM\_DAYS

### Syntax

```
FROM_DAYS(N)
```

### Description

Given a day number N, returns a DATE value. The day count is based on the number of days from the start of the standard calendar (0000-00-00).

The function is not designed for use with dates before the advent of the Gregorian calendar in October 1582. Results will not be reliable since it doesn't account for the lost days when the calendar changed from the Julian calendar.

This is the converse of the [TO\\_DAYS\(\)](#) function.

## Examples

```
SELECT FROM_DAYS(730669);
+-----+
| FROM_DAYS(730669) |
+-----+
| 2000-07-03 |
+-----+
```

## 1.1.12.3.24 FROM\_UNIXTIME

### Syntax

```
FROM_UNIXTIME(unix_timestamp), FROM_UNIXTIME(unix_timestamp,format)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Performance Considerations](#)
4. [Examples](#)
5. [See Also](#)

### Description

Returns a representation of the unix\_timestamp argument as a value in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uduuuu format, depending on whether the function is used in a string or numeric context. The value is expressed in the current [time zone](#). unix\_timestamp is an internal timestamp value such as is produced by the [UNIX\\_TIMESTAMP\(\)](#) function.

If format is given, the result is formatted according to the format string, which is used the same way as listed in the entry for the [DATE\\_FORMAT\(\)](#) function.

Timestamps in MariaDB have a maximum value of 2147483647, equivalent to 2038-01-19 05:14:07. This is due to the underlying 32-bit limitation. Using the function on a timestamp beyond this will result in NULL being returned. Use [DATETIME](#) as a storage type if you require dates beyond this.

The options that can be used by `FROM_UNIXTIME()`, as well as [DATE\\_FORMAT\(\)](#) and [STR\\_TO\\_DATE\(\)](#) , are:

Option	Description
%a	Short weekday name in current locale (Variable <a href="#">lc_time_names</a> ).
%b	Short form month name in current locale. For locale en_US this is one of: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov or Dec.
%c	Month with 1 or 2 digits.
%d	Day with English suffix 'th', 'nd', 'st' or 'rd'. (1st, 2nd, 3rd...).
%d	Day with 2 digits.
%e	Day with 1 or 2 digits.
%f	<a href="#">Microseconds</a> 6 digits.
%H	Hour with 2 digits between 00-23.

%h	Hour with 2 digits between 01-12.
%I	Hour with 2 digits between 01-12.
%i	Minute with 2 digits.
%j	Day of the year (001-366)
%k	Hour with 1 digits between 0-23.
%l	Hour with 1 digits between 1-12.
%M	Full month name in current locale (Variable <a href="#">lc_time_names</a> ).
%m	Month with 2 digits.
%p	AM/PM according to current locale (Variable <a href="#">lc_time_names</a> ).
%r	Time in 12 hour format, followed by AM/PM. Short for "%l:%i:%S %p".
%S	Seconds with 2 digits.
%s	Seconds with 2 digits.
%T	Time in 24 hour format. Short for "%H:%i:%S".
%U	Week number (00-53), when first day of the week is Sunday.
%u	Week number (00-53), when first day of the week is Monday.
%v	Week number (01-53), when first day of the week is Sunday. Used with %X.
%v	Week number (01-53), when first day of the week is Monday. Used with %x.
%w	Full weekday name in current locale (Variable <a href="#">lc_time_names</a> ).
%w	Day of the week. 0 = Sunday, 6 = Saturday.
%x	Year with 4 digits when first day of the week is Sunday. Used with %V.

%x	Year with 4 digits when first day of the week is Sunday. Used with %v.
%Y	Year with 4 digits.
%y	Year with 2 digits.
%#	For <code>str_to_date</code> (), skip all numbers.
%.	For <code>str_to_date</code> (), skip all punctuation characters.
%@	For <code>str_to_date</code> (), skip all alpha characters.
%%	A literal % character.

## Performance Considerations

If your session time zone is set to

```
SYSTEM
(the default),
FROM_UNIXTIME()
will call the OS function to convert the data using the system time zone. At least on Linux, the corresponding function (
localtime_r
) uses a global mutex inside glibc that can cause contention under high concurrent load.
```

Set your time zone to a named time zone to avoid this issue. See [mysql time zone tables](#) for details on how to do this.

## Examples

```
SELECT FROM_UNIXTIME(1196440219);
+-----+
| FROM_UNIXTIME(1196440219) |
+-----+
| 2007-11-30 11:30:19 |
+-----+

SELECT FROM_UNIXTIME(1196440219) + 0;
+-----+
| FROM_UNIXTIME(1196440219) + 0 |
+-----+
| 20071130113019.000000 |
+-----+

SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(), '%Y %D %M %h:%i:%s %x');
+-----+
| FROM_UNIXTIME(UNIX_TIMESTAMP(), '%Y %D %M %h:%i:%s %x') |
+-----+
| 2010 27th March 01:03:47 |
+-----+
```

## See Also

- [UNIX\\_TIMESTAMP\(\)](#)
- [DATE\\_FORMAT\(\)](#)
- [STR\\_TO\\_DATE\(\)](#)

## 1.1.12.3.25 GET\_FORMAT

# Syntax

```
GET_FORMAT({DATE|DATETIME|TIME}, {'EUR' | 'USA' | 'JIS' | 'ISO' | 'INTERNAL'})
```

## Description

Returns a format string. This function is useful in combination with the [DATE\\_FORMAT\(\)](#) and the [STR\\_TO\\_DATE\(\)](#) functions.

Possible result formats are:

Function Call	Result Format
GET_FORMAT(DATE,'EUR')	'%d.%m.%Y'
GET_FORMAT(DATE,'USA')	'%m.%d.%Y'
GET_FORMAT(DATE,'JIS')	'%Y-%m-%d'
GET_FORMAT(DATE,'ISO')	'%Y-%m-%d'
GET_FORMAT(DATE,'INTERNAL')	'%Y%m%d'
GET_FORMAT(DATETIME,'EUR')	'%Y-%m-%d %H.%i.%s'
GET_FORMAT(DATETIME,'USA')	'%Y-%m-%d %H.%i.%s'
GET_FORMAT(DATETIME,'JIS')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(DATETIME,'ISO')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(DATETIME,'INTERNAL')	'%Y%m%d%H%i%s'
GET_FORMAT(TIME,'EUR')	'%H.%i.%s'
GET_FORMAT(TIME,'USA')	'%h:%i:%s %p'
GET_FORMAT(TIME,'JIS')	'%H:%i:%s'
GET_FORMAT(TIME,'ISO')	'%H:%i:%s'
GET_FORMAT(TIME,'INTERNAL')	'%H%i%s'

## Examples

Obtaining the string matching to the standard European date format:

```
SELECT GET_FORMAT(DATE, 'EUR');
+-----+
| GET_FORMAT(DATE, 'EUR') |
+-----+
| %d.%m.%Y |
+-----+
```

Using the same string to format a date:

```
SELECT DATE_FORMAT('2003-10-03',GET_FORMAT(DATE, 'EUR'));
+-----+
| DATE_FORMAT('2003-10-03',GET_FORMAT(DATE, 'EUR')) |
+-----+
| 03.10.2003 |
+-----+
SELECT STR_TO_DATE('10.31.2003',GET_FORMAT(DATE, 'USA'));
+-----+
| STR_TO_DATE('10.31.2003',GET_FORMAT(DATE, 'USA')) |
+-----+
| 2003-10-31 |
+-----+
```

## 1.1.12.3.26 HOUR

### Syntax

```
HOUR(time)
```

## Description

Returns the hour for time. The range of the return value is 0 to 23 for time-of-day values. However, the range of TIME values actually is much larger, so HOUR can return values greater than 23.

The return value is always positive, even if a negative TIME value is provided.

## Examples

```
SELECT HOUR('10:05:03');
+-----+
| HOUR('10:05:03') |
+-----+
|          10 |
+-----+
SELECT HOUR('272:59:59');
+-----+
| HOUR('272:59:59') |
+-----+
|         272 |
+-----+
```

Difference between

[EXTRACT \(HOUR FROM ...\)](#)

( $\geq$  MariaDB 10.0.7 and MariaDB 5.5.35) and  
HOUR  
:

```
SELECT EXTRACT(HOUR FROM '26:30:00'), HOUR('26:30:00');
+-----+-----+
| EXTRACT(HOUR FROM '26:30:00') | HOUR('26:30:00') |
+-----+-----+
|          2 |           26 |
+-----+-----+
```

## See Also

- [Date and Time Units](#)
- [Date and Time Literals](#)
- [EXTRACT\(\)](#)

## 1.1.12.3.27 LAST\_DAY

### Syntax

```
LAST_DAY(date)
```

## Description

Takes a date or datetime value and returns the corresponding value for the last day of the month. Returns NULL if the argument is invalid.

## Examples

```
SELECT LAST_DAY('2003-02-05');
+-----+
| LAST_DAY('2003-02-05') |
+-----+
| 2003-02-28 |
+-----+

SELECT LAST_DAY('2004-02-05');
+-----+
| LAST_DAY('2004-02-05') |
+-----+
| 2004-02-29 |
+-----+

SELECT LAST_DAY('2004-01-01 01:01:01');
+-----+
| LAST_DAY('2004-01-01 01:01:01') |
+-----+
| 2004-01-31 |
+-----+

SELECT LAST_DAY('2003-03-32');
+-----+
| LAST_DAY('2003-03-32') |
+-----+
| NULL |
+-----+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1292): Incorrect datetime value: '2003-03-32'
```

## 1.1.12.3.28 LOCALTIME

### Syntax

```
LOCALTIME
LOCALTIME([precision])
```

### Description

LOCALTIME  
and  
LOCALTIME()  
are synonyms for

NOW()

### See Also

- [Microseconds in MariaDB](#)

## 1.1.12.3.29 LOCALTIMESTAMP

### Syntax

```
LOCALTIMESTAMP
LOCALTIMESTAMP([precision])
```

### Description

LOCALTIMESTAMP  
and

`LOCALTIMESTAMP()`  
are synonyms for

`NOW()`

## See Also

- [Microseconds in MariaDB](#)

## 1.1.12.3.30 MAKEDATE

### Syntax

```
MAKEDATE(year,dayofyear)
```

### Description

Returns a date, given

year  
and  
day-of-year values  
  
dayofyear  
must be greater than 0 or the result is NULL.

### Examples

```
SELECT MAKEDATE(2011,31), MAKEDATE(2011,32);
+-----+-----+
| MAKEDATE(2011,31) | MAKEDATE(2011,32) |
+-----+-----+
| 2011-01-31       | 2011-02-01       |
+-----+-----+

SELECT MAKEDATE(2011,365), MAKEDATE(2014,365);
+-----+-----+
| MAKEDATE(2011,365) | MAKEDATE(2014,365) |
+-----+-----+
| 2011-12-31       | 2014-12-31       |
+-----+-----+

SELECT MAKEDATE(2011,0);
+-----+
| MAKEDATE(2011,0) |
+-----+
| NULL            |
+-----+
```

## 1.1.12.3.31 MAKETIME

### Syntax

```
MAKETIME(hour,minute,second)
```

### Description

Returns a time value calculated from the

hour  
,  
minute  
, and  
second

arguments.

If  
minute  
or  
second  
are out of the range 0 to 60, NULL is returned. The  
hour  
can be in the range -838 to 838, outside of which the value is truncated with a warning.

## Examples

```
SELECT MAKETIME(13,57,33);
+-----+
| MAKETIME(13,57,33) |
+-----+
| 13:57:33           |
+-----+

SELECT MAKETIME(-13,57,33);
+-----+
| MAKETIME(-13,57,33) |
+-----+
| -13:57:33          |
+-----+

SELECT MAKETIME(13,67,33);
+-----+
| MAKETIME(13,67,33) |
+-----+
| NULL               |
+-----+

SELECT MAKETIME(-1000,57,33);
+-----+
| MAKETIME(-1000,57,33) |
+-----+
| -838:59:59          |
+-----+
1 row in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+
| Level   | Code | Message           |
+-----+
| Warning | 1292 | Truncated incorrect time value: '-1000:57:33' |
+-----+
```

## 1.1.12.3.32 MICROSECOND

### Syntax

```
MICROSECOND(expr)
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the microseconds from the time or datetime expression *expr* as a number in the range from 0 to 999999.

If *expr* is a time with no microseconds, zero is returned, while if *expr* is a date with no time, zero with a warning is returned.

## Examples

```

SELECT MICROSECOND('12:00:00.123456');
+-----+
| MICROSECOND('12:00:00.123456') |
+-----+
|          123456 |
+-----+

SELECT MICROSECOND('2009-12-31 23:59:59.000010');
+-----+
| MICROSECOND('2009-12-31 23:59:59.000010') |
+-----+
|                  10 |
+-----+

SELECT MICROSECOND('2013-08-07 12:13:14');
+-----+
| MICROSECOND('2013-08-07 12:13:14') |
+-----+
|                  0 |
+-----+

SELECT MICROSECOND('2013-08-07');
+-----+
| MICROSECOND('2013-08-07') |
+-----+
|                  0 |
+-----+
1 row in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+
| Level | Code | Message           |
+-----+
| Warning | 1292 | Truncated incorrect time value: '2013-08-07' |
+-----+

```

## See Also

- [Microseconds in MariaDB](#)

## 1.1.12.3.33 MINUTE

### Syntax

`MINUTE(time)`

### Description

Returns the minute for `time`, in the range 0 to 59.

### Examples

```

SELECT MINUTE('2013-08-03 11:04:03');
+-----+
| MINUTE('2013-08-03 11:04:03') |
+-----+
|                  4 |
+-----+

SELECT MINUTE ('23:12:50');
+-----+
| MINUTE ('23:12:50') |
+-----+
|                 12 |
+-----+

```

## 1.1.12.3.34 MONTH

## Syntax

```
MONTH(date)
```

## Description

Returns the month for

date

in the range 1 to 12 for January to December, or 0 for dates such as '0000-00-00' or '2008-00-00' that have a zero month part.

## Examples

```
SELECT MONTH('2019-01-03');
+-----+
| MONTH('2019-01-03') |
+-----+
|          1 |
+-----+

SELECT MONTH('2019-00-03');
+-----+
| MONTH('2019-00-03') |
+-----+
|          0 |
+-----+
```

## 1.1.12.3.35 MONTHNAME

## Syntax

```
MONTHNAME(date)
```

## Description

Returns the full name of the month for date. The language used for the name is controlled by the value of the `lc_time_names` system variable. See [server locale](#) for more on the supported locales.

## Examples

```
SELECT MONTHNAME('2019-02-03');
+-----+
| MONTHNAME('2019-02-03') |
+-----+
| February           |
+-----+
```

Changing the locale:

```
SET lc_time_names = 'fr_CA';

SELECT MONTHNAME('2019-05-21');
+-----+
| MONTHNAME('2019-05-21') |
+-----+
| mai                 |
+-----+
```

## 1.1.12.3.36 NOW

## Syntax

```
NOW([precision])
CURRENT_TIMESTAMP
CURRENT_TIMESTAMP([precision])
LOCALTIME, LOCALTIME([precision])
LOCALTIMESTAMP
LOCALTIMESTAMP([precision])
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Returns the current date and time as a value in

```
'YYYY-MM-DD HH:MM:SS'
```

or

```
YYYYMMDDHHMMSS.uduuuu
```

format, depending on whether the function is used in a string or numeric context. The value is expressed in the current [time zone](#).

The optional *precision* determines the microsecond precision. See [Microseconds in MariaDB](#).

```
NOW()
```

(or its synonyms) can be used as the default value for [TIMESTAMP](#) columns as well as, since [MariaDB 10.0.1](#), [DATETIME](#) columns. Before [MariaDB 10.0.1](#), it was only possible for a single [TIMESTAMP](#) column per table to contain the [CURRENT\\_TIMESTAMP](#) as its default.

When displayed in the [INFORMATION\\_SCHEMA.COLUMNS](#) table, a default [CURRENT\\_TIMESTAMP](#) is displayed as

```
CURRENT_TIMESTAMP
```

up until [MariaDB 10.2.2](#), and as

```
current_timestamp()
```

from [MariaDB 10.2.3](#), due to [MariaDB 10.2](#) accepting expressions in the [DEFAULT](#) clause.

## Examples

```
SELECT NOW();
+-----+
| NOW()           |
+-----+
| 2010-03-27 13:13:25 |
+-----+

SELECT NOW() + 0;
+-----+
| NOW() + 0          |
+-----+
| 20100327131329.000000 |
+-----+
```

With precision:

```
SELECT CURRENT_TIMESTAMP(2);
+-----+
| CURRENT_TIMESTAMP(2)   |
+-----+
| 2018-07-10 09:47:26.24 |
+-----+
```

Used as a default [TIMESTAMP](#):

```
CREATE TABLE t (createdTS TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP);
```

From [MariaDB 10.2.2](#):

```
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA='test'  
AND COLUMN_NAME LIKE '%ts%'  
***** 1. row *****  
TABLE_CATALOG: def  
TABLE_SCHEMA: test  
TABLE_NAME: t  
COLUMN_NAME: ts  
ORDINAL_POSITION: 1  
COLUMN_DEFAULT: current_timestamp()  
...
```

<= MariaDB 10.2.1

```
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA='test'  
AND COLUMN_NAME LIKE '%ts%'  
***** 1. row *****  
TABLE_CATALOG: def  
TABLE_SCHEMA: test  
TABLE_NAME: t  
COLUMN_NAME: createdTS  
ORDINAL_POSITION: 1  
COLUMN_DEFAULT: CURRENT_TIMESTAMP  
...
```

## See Also

- [Microseconds in MariaDB](#)
- [timestamp server system variable](#)

## 1.1.12.3.37 PERIOD\_ADD

### Syntax

```
PERIOD_ADD(P,N)
```

### Description

Adds

N

months to period

P

.

P

is in the format YYMM or YYYYMM, and is not a date value. If

P

contains a two-digit year, values from 00 to 69 are converted to from 2000 to 2069, while values from 70 are converted to 1970 upwards.

Returns a value in the format YYYYMM.

### Examples

```
SELECT PERIOD_ADD(200801,2);
```

```
+-----+
| PERIOD_ADD(200801,2) |
+-----+
|      200803 |
+-----+
```

```
SELECT PERIOD_ADD(6910,2);
```

```
+-----+
| PERIOD_ADD(6910,2) |
+-----+
|      206912 |
+-----+
```

```
SELECT PERIOD_ADD(7010,2);
```

```
+-----+
| PERIOD_ADD(7010,2) |
+-----+
|      197012 |
+-----+
```

## 1.1.12.3.38 PERIOD\_DIFF

### Syntax

```
PERIOD_DIFF(P1,P2)
```

### Description

Returns the number of months between periods P1 and P2. P1 and P2 can be in the format

YYMM

or

YYYYMM

, and are not date values.

If P1 or P2 contains a two-digit year, values from 00 to 69 are converted to from 2000 to 2069, while values from 70 are converted to 1970 upwards.

### Examples

```
SELECT PERIOD_DIFF(200802,200703);
```

```
+-----+
| PERIOD_DIFF(200802,200703) |
+-----+
|      11 |
+-----+
```

```
SELECT PERIOD_DIFF(6902,6803);
```

```
+-----+
| PERIOD_DIFF(6902,6803) |
+-----+
|      11 |
+-----+
```

```
SELECT PERIOD_DIFF(7002,6803);
```

```
+-----+
| PERIOD_DIFF(7002,6803) |
+-----+
|     -1177 |
+-----+
```

## 1.1.12.3.39 QUARTER

### Syntax

```
QUARTER(date)
```

## Description

Returns the quarter of the year for

date  
, in the range 1 to 4. Returns 0 if month contains a zero value, or  
NULL  
if the given value is not otherwise a valid date (zero values are accepted).

## Examples

```
SELECT QUARTER('2008-04-01');
+-----+
| QUARTER('2008-04-01') |
+-----+
|          2          |
+-----+
SELECT QUARTER('2019-00-01');
+-----+
| QUARTER('2019-00-01') |
+-----+
|          0          |
+-----+
```

## 1.1.12.3.40 SECOND

### Syntax

```
SECOND(time)
```

## Description

Returns the second for a given

time  
(which can include microseconds ), in the range 0 to 59, or  
NULL  
if not given a valid time value.

## Examples

```
SELECT SECOND('10:05:03');
+-----+
| SECOND('10:05:03') |
+-----+
|          3          |
+-----+
SELECT SECOND('10:05:01.999999');
+-----+
| SECOND('10:05:01.999999') |
+-----+
|          1          |
+-----+
```

## 1.1.12.3.41 SEC\_TO\_TIME

### Syntax

```
SEC_TO_TIME(seconds)
```

## Description

Returns the seconds argument, converted to hours, minutes, and seconds, as a TIME value. The range of the result is constrained to that of the TIME

[data type](#). A warning occurs if the argument corresponds to a value outside that range.

The time will be returned in the format

hh:mm:ss  
, or  
hhmmss  
if used in a numeric calculation.

## Examples

```
SELECT SEC_TO_TIME(12414);
+-----+
| SEC_TO_TIME(12414) |
+-----+
| 03:26:54           |
+-----+

SELECT SEC_TO_TIME(12414)+0;
+-----+
| SEC_TO_TIME(12414)+0 |
+-----+
|            32654   |
+-----+

SELECT SEC_TO_TIME(9999999);
+-----+
| SEC_TO_TIME(9999999) |
+-----+
| 838:59:59           |
+-----+
1 row in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+
| Level | Code | Message          |
+-----+
| Warning | 1292 | Truncated incorrect time value: '9999999' |
+-----+
```

## 1.1.12.3.42 STR\_TO\_DATE

### Syntax

```
STR_TO_DATE(str,format)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

This is the inverse of the [DATE\\_FORMAT \(\)](#) function. It takes a string

str  
and a format string  
format  
  
STR\_TO\_DATE()  
returns a  
DATETIME  
value if the format string contains both date and time parts, or a  
DATE  
or  
TIME  
value if the string contains only date or time parts.

The date, time, or datetime values contained in

str

```

should be given in the format indicated by format. If str contains an illegal date, time, or datetime value,
STR_TO_DATE()
returns
NULL
. An illegal value also produces a warning.

```

The options that can be used by STR\_TO\_DATE(), as well as its inverse [DATE\\_FORMAT\(\)](#) and the [FROM\\_UNIXTIME\(\)](#) function, are:

Option	Description
%a	Short weekday name in current locale (Variable <a href="#">lc_time_names</a> ).
%b	Short form month name in current locale. For locale en_US this is one of: Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov or Dec.
%c	Month with 1 or 2 digits.
%d	Day with English suffix 'th', 'nd', 'st' or 'rd'. (1st, 2nd, 3rd...).
%d	Day with 2 digits.
%e	Day with 1 or 2 digits.
%f	<a href="#">Microseconds</a> 6 digits.
%H	Hour with 2 digits between 00-23.
%h	Hour with 2 digits between 01-12.
%I	Hour with 2 digits between 01-12.
%i	Minute with 2 digits.
%j	Day of the year (001-366)
%k	Hour with 1 digits between 0-23.
%l	Hour with 1 digits between 1-12.
%M	Full month name in current locale (Variable <a href="#">lc_time_names</a> ).
%m	Month with 2 digits.
%p	AM/PM according to current locale (Variable <a href="#">lc_time_names</a> ).

%r	Time in 12 hour format, followed by AM/PM. Short for '%l:%i:%S %p'.
%s	Seconds with 2 digits.
%s	Seconds with 2 digits.
%T	Time in 24 hour format. Short for '%H:%i:%S'.
%U	Week number (00-53), when first day of the week is Sunday.
%u	Week number (00-53), when first day of the week is Monday.
%v	Week number (01-53), when first day of the week is Sunday. Used with %X.
%v	Week number (01-53), when first day of the week is Monday. Used with %x.
%w	Full weekday name in current locale (Variable <a href="#">lc_time_names</a> ).
%w	Day of the week. 0 = Sunday, 6 = Saturday.
%x	Year with 4 digits when first day of the week is Sunday. Used with %V.
%x	Year with 4 digits when first day of the week is Monday. Used with %v.
%Y	Year with 4 digits.
%y	Year with 2 digits.
%#	For <a href="#">str_to_date</a> (), skip all numbers.
%.	For <a href="#">str_to_date</a> (), skip all punctuation characters.
%@	For <a href="#">str_to_date</a> (), skip all alpha characters.
%%	A literal % character.

## Examples

```

SELECT STR_TO_DATE('Wednesday, June 2, 2014', '%W, %M %e, %Y');
+-----+
| STR_TO_DATE('Wednesday, June 2, 2014', '%W, %M %e, %Y') |
+-----+
| 2014-06-02 |
+-----+


SELECT STR_TO_DATE('Wednesday23423, June 2, 2014', '%W, %M %e, %Y');
+-----+
| STR_TO_DATE('Wednesday23423, June 2, 2014', '%W, %M %e, %Y') |
+-----+
| NULL |
+-----+
1 row in set, 1 warning (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1411 | Incorrect datetime value: 'Wednesday23423, June 2, 2014' for function str_to_date |
+-----+-----+


SELECT STR_TO_DATE('Wednesday23423, June 2, 2014', '%W%, %M %e, %Y');
+-----+
| STR_TO_DATE('Wednesday23423, June 2, 2014', '%W%, %M %e, %Y') |
+-----+
| 2014-06-02 |
+-----+

```

## See Also

- [DATE\\_FORMAT\(\)](#)
- [FROM\\_UNIXTIME\(\)](#)

## 1.1.12.3.43 SUBDATE

### Syntax

```
SUBDATE(date,INTERVAL expr unit), SUBDATE(expr,days)
```

### Description

When invoked with the

INTERVAL  
form of the second argument,  
SUBDATE()  
is a synonym for

[DATE\\_SUB\(\)](#)

. See [Date and Time Units](#) for a complete list of permitted units.

The second form allows the use of an integer value for days. In such cases, it is interpreted as the number of days to be subtracted from the date or datetime expression expr.

### Examples

```

SELECT DATE_SUB('2008-01-02', INTERVAL 31 DAY);
+-----+
| DATE_SUB('2008-01-02', INTERVAL 31 DAY) |
+-----+
| 2007-12-02                                |
+-----+


SELECT SUBDATE('2008-01-02', INTERVAL 31 DAY);
+-----+
| SUBDATE('2008-01-02', INTERVAL 31 DAY) |
+-----+
| 2007-12-02                                |
+-----+

```

```

SELECT SUBDATE('2008-01-02 12:00:00', 31);
+-----+
| SUBDATE('2008-01-02 12:00:00', 31) |
+-----+
| 2007-12-02 12:00:00                   |
+-----+

```

```

CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");

```

```

SELECT d, SUBDATE(d, 10) from t1;
+-----+
| d           | SUBDATE(d, 10)   |
+-----+
| 2007-01-30 21:31:07 | 2007-01-20 21:31:07 |
| 1983-10-15 06:42:51 | 1983-10-05 06:42:51 |
| 2011-04-21 12:34:56 | 2011-04-11 12:34:56 |
| 2011-10-30 06:31:41 | 2011-10-20 06:31:41 |
| 2011-01-30 14:03:25 | 2011-01-20 14:03:25 |
| 2004-10-07 11:19:34 | 2004-09-27 11:19:34 |
+-----+


SELECT d, SUBDATE(d, INTERVAL 10 MINUTE) from t1;
+-----+
| d           | SUBDATE(d, INTERVAL 10 MINUTE) |
+-----+
| 2007-01-30 21:31:07 | 2007-01-30 21:21:07 |
| 1983-10-15 06:42:51 | 1983-10-15 06:32:51 |
| 2011-04-21 12:34:56 | 2011-04-21 12:24:56 |
| 2011-10-30 06:31:41 | 2011-10-30 06:21:41 |
| 2011-01-30 14:03:25 | 2011-01-30 13:53:25 |
| 2004-10-07 11:19:34 | 2004-10-07 11:09:34 |
+-----+

```

## 1.1.12.3.44 SUBTIME

### Syntax

```
SUBTIME(expr1,expr2)
```

### Description

SUBTIME() returns

```

expr1
-
expr2
expressed as a value in the same format as
expr1

```

expr1  
is a time or datetime expression, and expr2 is a time expression.

## Examples

```
SELECT SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002');
+-----+
| SUBTIME('2007-12-31 23:59:59.999999','1 1:1:1.000002') |
+-----+
| 2007-12-30 22:58:58.999997 |
+-----+

SELECT SUBTIME('01:00:00.999999', '02:00:00.999998');
+-----+
| SUBTIME('01:00:00.999999', '02:00:00.999998') |
+-----+
| -00:59:59.999999 |
+-----+
```

## 1.1.12.3.45 SYSDATE

### Syntax

```
SYSDATE([precision])
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the current date and time as a value in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uduuuu format, depending on whether the function is used in a string or numeric context.

The optional *precision* determines the microsecond precision. See [Microseconds in MariaDB](#).

SYSDATE() returns the time at which it executes. This differs from the behavior for [NOW\(\)](#), which returns a constant time that indicates the time at which the statement began to execute. (Within a stored routine or trigger, NOW() returns the time at which the routine or triggering statement began to execute.)

In addition, changing the [timestamp system variable](#) with a [SET](#)

timestamp

statement affects the value returned by NOW() but not by SYSDATE(). This means that timestamp settings in the [binary log](#) have no effect on invocations of SYSDATE().

Because SYSDATE() can return different values even within the same statement, and is not affected by SET TIMESTAMP, it is non-deterministic and therefore unsafe for replication if statement-based binary logging is used. If that is a problem, you can use row-based logging, or start the server with the mysqld option [--sysdate-is-now](#) to cause SYSDATE() to be an alias for NOW(). The non-deterministic nature of SYSDATE() also means that indexes cannot be used for evaluating expressions that refer to it, and that statements using the SYSDATE() function are [unsafe for statement-based replication](#).

## Examples

Difference between NOW() and SYSDATE():

```
SELECT NOW(), SLEEP(2), NOW();
+-----+-----+-----+
| NOW() | SLEEP(2) | NOW() |
+-----+-----+-----+
| 2010-03-27 13:23:40 | 0 | 2010-03-27 13:23:40 |
+-----+-----+-----+

SELECT SYSDATE(), SLEEP(2), SYSDATE();
+-----+-----+-----+
| SYSDATE() | SLEEP(2) | SYSDATE() |
+-----+-----+-----+
| 2010-03-27 13:23:52 | 0 | 2010-03-27 13:23:54 |
+-----+-----+-----+
```

With precision:

```
SELECT SYSDATE(4);
+-----+
| SYSDATE(4)           |
+-----+
| 2018-07-10 10:17:13.1689 |
+-----+
```

## See Also

- [Microseconds in MariaDB](#)
- [timestamp server system variable](#)

## 1.1.12.3.46 TIME Function

### Syntax

```
TIME(expr)
```

### Description

Extracts the time part of the time or datetime expression

expr  
and returns it as a string.

### Examples

```
SELECT TIME('2003-12-31 01:02:03');
+-----+
| TIME('2003-12-31 01:02:03') |
+-----+
| 01:02:03           |
+-----+

SELECT TIME('2003-12-31 01:02:03.000123');
+-----+
| TIME('2003-12-31 01:02:03.000123') |
+-----+
| 01:02:03.000123           |
+-----+
```

## 1.1.12.3.47 TIMEDIFF

### Syntax

```
TIMEDIFF(expr1,expr2)
```

### Description

TIMEDIFF() returns

expr1  
-  
expr2  
expressed as a time value.  
expr1  
and  
expr2  
are time or date-and-time expressions, but both must be of the same type.

### Examples

```

SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001');
+-----+
| TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001') |
+-----+
| -00:00:00.000001 |
+-----+

SELECT TIMEDIFF('2008-12-31 23:59:59.000001', '2008-12-30 01:01:01.000002');
+-----+
| TIMEDIFF('2008-12-31 23:59:59.000001', '2008-12-30 01:01:01.000002') |
+-----+
| 46:58:57.999999 |
+-----+

```

## 1.1.12.3.48 TIMESTAMP FUNCTION

### Syntax

```
TIMESTAMP(expr), TIMESTAMP(expr1,expr2)
```

### Description

With a single argument, this function returns the date or datetime expression

expr

as a datetime value. With two arguments, it adds the time expression

expr2

to the date or datetime expression

expr1

and returns the result as a datetime value.

### Examples

```

SELECT TIMESTAMP('2003-12-31');
+-----+
| TIMESTAMP('2003-12-31') |
+-----+
| 2003-12-31 00:00:00 |
+-----+

SELECT TIMESTAMP('2003-12-31 12:00:00','6:30:00');
+-----+
| TIMESTAMP('2003-12-31 12:00:00','6:30:00') |
+-----+
| 2003-12-31 18:30:00 |
+-----+

```

## 1.1.12.3.49 TIMESTAMPADD

### Syntax

```
TIMESTAMPADD(unit,interval,datetime_expr)
```

### Description

Adds the integer expression interval to the date or datetime expression datetime\_expr. The unit for interval is given by the unit argument, which should be one of the following values: MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, or YEAR.

The unit value may be specified using one of keywords as shown, or with a prefix of SQL\_TSI\_. For example, DAY and SQL\_TSI\_DAY both are legal.

Before MariaDB 5.5 , FRAC\_SECOND was permitted as a synonym for MICROSECOND.

### Examples

```

SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');
+-----+
| TIMESTAMPADD(MINUTE,1,'2003-01-02') |
+-----+
| 2003-01-02 00:01:00 |
+-----+

SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');
+-----+
| TIMESTAMPADD(WEEK,1,'2003-01-02') |
+-----+
| 2003-01-09 |
+-----+

```

## 1.1.12.3.50 TIMESTAMPDIFF

### Syntax

```
TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2)
```

### Description

Returns

```

datetime_expr2
-
datetime_expr1
, where
datetime_expr1
and
datetime_expr2

```

are date or datetime expressions. One expression may be a date and the other a datetime; a date value is treated as a datetime having the time part '00:00:00' where necessary. The unit for the result (an integer) is given by the unit argument. The legal values for unit are the same as those listed in the description of the [TIMESTAMPADD\(\)](#) function, i.e MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, or YEAR.

`TIMESTAMPDIFF`

can also be used to calculate age.

### Examples

```

SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');
+-----+
| TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01') |
+-----+
| 3 |
+-----+

SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');
+-----+
| TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01') |
+-----+
| -1 |
+-----+

SELECT TIMESTAMPDIFF(MINUTE,'2003-02-01','2003-05-01 12:05:55');
+-----+
| TIMESTAMPDIFF(MINUTE,'2003-02-01','2003-05-01 12:05:55') |
+-----+
| 128885 |
+-----+

```

Calculating age:

```

SELECT CURDATE();
+-----+
| CURDATE() |
+-----+
| 2019-05-27 |
+-----+

SELECT TIMESTAMPDIFF(YEAR, '1971-06-06', CURDATE()) AS age;
+-----+
| age |
+-----+
| 47 |
+-----+

SELECT TIMESTAMPDIFF(YEAR, '1971-05-06', CURDATE()) AS age;
+-----+
| age |
+-----+
| 48 |
+-----+

```

Age as of 2014-08-02:

```

SELECT name, date_of_birth, TIMESTAMPDIFF(YEAR,date_of_birth,'2014-08-02') AS age
  FROM student_details;
+-----+-----+-----+
| name | date_of_birth | age |
+-----+-----+-----+
| Chun | 1993-12-31 | 20 |
| Esben | 1946-01-01 | 68 |
| Kaolin | 1996-07-16 | 18 |
| Tatiana | 1988-04-13 | 26 |
+-----+-----+-----+

```

## 1.1.12.3.51 TIME\_FORMAT

### Syntax

```
TIME_FORMAT(time,format)
```

### Description

This is used like the [DATE\\_FORMAT\(\)](#) function, but the format string may contain format specifiers only for hours, minutes, and seconds. Other specifiers produce a NULL value or 0.

### Examples

```

SELECT TIME_FORMAT('100:00:00', '%H %k %h %I %l');
+-----+
| TIME_FORMAT('100:00:00', '%H %k %h %I %l') |
+-----+
| 100 100 04 04 4 |
+-----+

```

## 1.1.12.3.52 TIME\_TO\_SEC

### Syntax

```
TIME_TO_SEC(time)
```

### Description

Returns the time argument, converted to seconds.

The value returned by

TIME\_TO\_SEC  
is of type

DOUBLE

. Before MariaDB 5.3 (and MySQL 5.6), the type was

INT

. The returned value preserves microseconds of the argument. See also [Microseconds in MariaDB](#).

## Examples

```
SELECT TIME_TO_SEC('22:23:00');
+-----+
| TIME_TO_SEC('22:23:00') |
+-----+
|          80580 |
+-----+
```

```
SELECT TIME_TO_SEC('00:39:38');
+-----+
| TIME_TO_SEC('00:39:38') |
+-----+
|          2378 |
+-----+
```

```
SELECT TIME_TO_SEC('09:12:55.2355');
+-----+
| TIME_TO_SEC('09:12:55.2355') |
+-----+
|          33175.2355 |
+-----+
1 row in set (0.000 sec)
```

## 1.1.12.3.53 TO\_DAYS

### Syntax

```
TO_DAYS(date)
```

### Description

Given a date

date  
, returns the number of days since the start of the current calendar (0000-00-00).

The function is not designed for use with dates before the advent of the Gregorian calendar in October 1582. Results will not be reliable since it doesn't account for the lost days when the calendar changed from the Julian calendar.

This is the converse of the [FROM\\_DAYS\(\)](#) function.

## Examples

```
SELECT TO_DAYS('2007-10-07');
+-----+
| TO_DAYS('2007-10-07') |
+-----+
|      733321 |
+-----+

SELECT TO_DAYS('0000-01-01');
+-----+
| TO_DAYS('0000-01-01') |
+-----+
|          1 |
+-----+

SELECT TO_DAYS(950501);
+-----+
| TO_DAYS(950501) |
+-----+
|      728779 |
+-----+
```

## 1.1.12.3.54 TO\_SECONDS

### Syntax

```
TO_SECONDS(expr)
```

### Description

Returns the number of seconds from year 0 till

expr  
, or NULL if  
expr  
is not a valid date or [datetime](#).

### Examples

```

SELECT TO_SECONDS('2013-06-13');
+-----+
| TO_SECONDS('2013-06-13') |
+-----+
|          63538300800 |
+-----+

SELECT TO_SECONDS('2013-06-13 21:45:13');
+-----+
| TO_SECONDS('2013-06-13 21:45:13') |
+-----+
|          63538379113 |
+-----+

SELECT TO_SECONDS(NOW());
+-----+
| TO_SECONDS(NOW()) |
+-----+
|      63543530875 |
+-----+

SELECT TO_SECONDS(20130513);
+-----+
| TO_SECONDS(20130513) |
+-----+
|          63535622400 |
+-----+
1 row in set (0.00 sec)

SELECT TO_SECONDS(130513);
+-----+
| TO_SECONDS(130513) |
+-----+
|          63535622400 |
+-----+

```

## 1.1.12.3.55 UNIX\_TIMESTAMP

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [Error Handling](#)
  2. [Compatibility](#)
3. [Examples](#)
4. [See Also](#)

### Syntax

```

UNIX_TIMESTAMP()
UNIX_TIMESTAMP(date)

```

### Description

If called with no argument, returns a Unix timestamp (seconds since '1970-01-01 00:00:00' UTC ) as an unsigned integer. If

`UNIX_TIMESTAMP()`

is called with a date argument, it returns the value of the argument as seconds since '1970-01-01 00:00:00' UTC. date may be a

`DATE`

`string, a`

`DATETIME`

`string, a`

`TIMESTAMP`

, or a number in the format YYMMDD or YYYYMMDD. The server interprets date as a value in the current [time zone](#) and converts it to an internal value in [UTC](#) . Clients can set their time zone as described in [time zones](#) .

The inverse function of

UNIX\_TIMESTAMP()  
is

FROM\_UNIXTIME()

UNIX\_TIMESTAMP()  
supports microseconds .

Timestamps in MariaDB have a maximum value of 2147483647, equivalent to 2038-01-19 05:14:07. This is due to the underlying 32-bit limitation. Using the function on a date beyond this will result in NULL being returned. Use DATETIME as a storage type if you require dates beyond this.

## Error Handling

Returns NULL for wrong arguments to

UNIX\_TIMESTAMP()  
. In MySQL and MariaDB before 5.3 wrong arguments to  
UNIX\_TIMESTAMP()  
returned 0.

## Compatibility

As you can see in the examples above, UNIX\_TIMESTAMP(constant-date-string) returns a timestamp with 6 decimals while MariaDB 5.2 and before returns it without decimals. This can cause a problem if you are using UNIX\_TIMESTAMP() as a partitioning function. You can fix this by using FLOOR(UNIX\_TIMESTAMP(..)) or changing the date string to a date number, like 20080101000000.

## Examples

```
SELECT UNIX_TIMESTAMP();
+-----+
| UNIX_TIMESTAMP() |
+-----+
|      1269711082 |
+-----+

SELECT UNIX_TIMESTAMP('2007-11-30 10:30:19');
+-----+
| UNIX_TIMESTAMP('2007-11-30 10:30:19') |
+-----+
|          1196436619.000000 |
+-----+

SELECT UNIX_TIMESTAMP("2007-11-30 10:30:19.123456");
+-----+
| unix_timestamp("2007-11-30 10:30:19.123456") |
+-----+
|          1196411419.123456 |
+-----+

SELECT FROM_UNIXTIME(UNIX_TIMESTAMP('2007-11-30 10:30:19'));
+-----+
| FROM_UNIXTIME(UNIX_TIMESTAMP('2007-11-30 10:30:19')) |
+-----+
| 2007-11-30 10:30:19.000000 |
+-----+

SELECT FROM_UNIXTIME(FLOOR(UNIX_TIMESTAMP('2007-11-30 10:30:19')));
+-----+
| FROM_UNIXTIME(FLOOR(UNIX_TIMESTAMP('2007-11-30 10:30:19'))) |
+-----+
| 2007-11-30 10:30:19 |
+-----+
```

## See Also

- [FROM\\_UNIXTIME\(\)](#)

## 1.1.12.3.56 UTC\_DATE

### Syntax

```
UTC_DATE, UTC_DATE()
```

### Description

Returns the current [UTC date](#) as a value in 'YYYY-MM-DD' or YYYYMMDD format, depending on whether the function is used in a string or numeric context.

### Examples

```
SELECT UTC_DATE(), UTC_DATE() + 0;
+-----+
| UTC_DATE() | UTC_DATE() + 0 |
+-----+
| 2010-03-27 |      20100327 |
+-----+
```

## 1.1.12.3.57 UTC\_TIME

### Syntax

```
UTC_TIME
UTC_TIME([precision])
```

### Description

Returns the current [UTC time](#) as a value in 'HH:MM:SS' or HHMMSS.uduuuu format, depending on whether the function is used in a string or numeric context.

The optional *precision* determines the microsecond precision. See [Microseconds in MariaDB](#).

### Examples

```
SELECT UTC_TIME(), UTC_TIME() + 0;
+-----+
| UTC_TIME() | UTC_TIME() + 0 |
+-----+
| 17:32:34 | 173234.000000 |
+-----+
```

With precision:

```
SELECT UTC_TIME(5);
+-----+
| UTC_TIME(5) |
+-----+
| 07:52:50.78369 |
+-----+
```

### See Also

- [Microseconds in MariaDB](#)

## 1.1.12.3.58 UTC\_TIMESTAMP

### Syntax

```
UTC_TIMESTAMP  
UTC_TIMESTAMP([precision])
```

## Description

Returns the current [UTC](#) date and time as a value in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uduuuu format, depending on whether the function is used in a string or numeric context.

The optional *precision* determines the microsecond precision. See [Microseconds in MariaDB](#).

## Examples

```
SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() + 0;  
+-----+  
| UTC_TIMESTAMP() | UTC_TIMESTAMP() + 0 |  
+-----+  
| 2010-03-27 17:33:16 | 20100327173316.000000 |  
+-----+
```

With precision:

```
SELECT UTC_TIMESTAMP(4);  
+-----+  
| UTC_TIMESTAMP(4) |  
+-----+  
| 2018-07-10 07:51:09.1019 |  
+-----+
```

## See Also

- [Time Zones](#)
- [Microseconds in MariaDB](#)

## 1.1.12.3.59 WEEK

### Syntax

```
WEEK(date[,mode])
```

## Description

This function returns the week number for

date

. The two-argument form of

WEEK()

allows you to specify whether the week starts on Sunday or Monday and whether the return value should be in the range from 0 to 53 or from 1 to 53. If the

mode

argument is omitted, the value of the [default\\_week\\_format](#) system variable is used.

### Modes

Mode	1st day of week	Range	Week 1 is the 1st week with
0	Sunday	0-53	a Sunday in this year
1	Monday	0-53	more than 3 days this year
2	Sunday	1-53	a Sunday in this year
3	Monday	1-53	more than 3 days this year
4	Sunday	0-53	more than 3 days this year
5	Monday	0-53	a Monday in this year

6	Sunday	1-53	more than 3 days this year
7	Monday	1-53	a Monday in this year

With the mode value of 3, which means “more than 3 days this year”, weeks are numbered according to ISO 8601:1988.

## Examples

```
SELECT WEEK('2008-02-20');
```

```
+-----+
| WEEK('2008-02-20') |
+-----+
| 7 |
+-----+
```

```
SELECT WEEK('2008-02-20',0);
```

```
+-----+
| WEEK('2008-02-20',0) |
+-----+
| 7 |
+-----+
```

```
SELECT WEEK('2008-02-20',1);
```

```
+-----+
| WEEK('2008-02-20',1) |
+-----+
| 8 |
+-----+
```

```
SELECT WEEK('2008-12-31');
```

```
+-----+
| WEEK('2008-12-31') |
+-----+
| 52 |
+-----+
```

```
SELECT WEEK('2008-12-31',1);
```

```
+-----+
| WEEK('2008-12-31',1) |
+-----+
| 53 |
+-----+
```

```
SELECT WEEK('2019-12-30',3);
```

```
+-----+
| WEEK('2019-12-30',3) |
+-----+
| 1 |
+-----+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT d, WEEK(d,0), WEEK(d,1) from t1;
```

```
+-----+-----+-----+
| d | WEEK(d,0) | WEEK(d,1) |
+-----+-----+-----+
| 2007-01-30 21:31:07 | 4 | 5 |
| 1983-10-15 06:42:51 | 41 | 41 |
| 2011-04-21 12:34:56 | 16 | 16 |
| 2011-10-30 06:31:41 | 44 | 43 |
| 2011-01-30 14:03:25 | 5 | 4 |
| 2004-10-07 11:19:34 | 40 | 41 |
+-----+-----+-----+
```

## 1.1.12.3.60 WEEKDAY

### Syntax

```
WEEKDAY(date)
```

### Description

Returns the weekday index for

```
date
(
0
= Monday,
1
= Tuesday, ...
6
= Sunday).
```

This contrasts with

```
DAYOFWEEK()
```

which follows the ODBC standard (

```
1
= Sunday,
2
= Monday, ...,
7
= Saturday).
```

### Examples

```
SELECT WEEKDAY('2008-02-03 22:23:00');
+-----+
| WEEKDAY('2008-02-03 22:23:00') |
+-----+
| 6 |
+-----+

SELECT WEEKDAY('2007-11-06');
+-----+
| WEEKDAY('2007-11-06') |
+-----+
| 1 |
+-----+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
("2007-01-30 21:31:07"),
("1983-10-15 06:42:51"),
("2011-04-21 12:34:56"),
("2011-10-30 06:31:41"),
("2011-01-30 14:03:25"),
("2004-10-07 11:19:34");
```

```
SELECT d FROM t1 WHERE WEEKDAY(d) = 6;
+-----+
| d |
+-----+
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
+-----+
```

## 1.1.12.3.61 WEEKOFYEAR

# Syntax

```
WEEKOFYEAR(date)
```

## Description

Returns the calendar week of the date as a number in the range from 1 to 53.

`WEEKOFYEAR()`

is a compatibility function that is equivalent to

```
WEEK(date,3)
```

## Examples

```
SELECT WEEKOFYEAR('2008-02-20');
+-----+
| WEEKOFYEAR('2008-02-20') |
+-----+
|          8 |
+-----+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
select * from t1;
+-----+
| d           |
+-----+
| 2007-01-30 21:31:07 |
| 1983-10-15 06:42:51 |
| 2011-04-21 12:34:56 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
| 2004-10-07 11:19:34 |
+-----+
```

```
SELECT d, WEEKOFYEAR(d), WEEK(d,3) from t1;
+-----+-----+-----+
| d           | WEEKOFYEAR(d) | WEEK(d,3) |
+-----+-----+-----+
| 2007-01-30 21:31:07 |      5 |      5 |
| 1983-10-15 06:42:51 |     41 |     41 |
| 2011-04-21 12:34:56 |     16 |     16 |
| 2011-10-30 06:31:41 |     43 |     43 |
| 2011-01-30 14:03:25 |      4 |      4 |
| 2004-10-07 11:19:34 |     41 |     41 |
+-----+-----+-----+
```

## 1.1.12.3.62 YEAR

### Syntax

```
YEAR(date)
```

## Description

Returns the year for the given date, in the range 1000 to 9999, or 0 for the "zero" date.

## Examples

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT * FROM t1;
+-----+
| d
+-----+
| 2007-01-30 21:31:07 |
| 1983-10-15 06:42:51 |
| 2011-04-21 12:34:56 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
| 2004-10-07 11:19:34 |
+-----+

SELECT * FROM t1 WHERE YEAR(d) = 2011;
+-----+
| d
+-----+
| 2011-04-21 12:34:56 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
+-----+
```

```
SELECT YEAR('1987-01-01');
+-----+
| YEAR('1987-01-01') |
+-----+
| 1987 |
+-----+
```

## See Also

- [YEAR data type](#)

## 1.1.12.3.63 YEARWEEK

### Syntax

```
YEARWEEK(date), YEARWEEK(date,mode)
```

### Description

Returns year and week for a date. The mode argument works exactly like the mode argument to [WEEK\(\)](#). The year in the result may be different from the year in the date argument for the first and the last week of the year.

## Examples

```
SELECT YEARWEEK('1987-01-01');
+-----+
| YEARWEEK('1987-01-01') |
+-----+
| 198652 |
+-----+
```

```
CREATE TABLE t1 (d DATETIME);
INSERT INTO t1 VALUES
    ("2007-01-30 21:31:07"),
    ("1983-10-15 06:42:51"),
    ("2011-04-21 12:34:56"),
    ("2011-10-30 06:31:41"),
    ("2011-01-30 14:03:25"),
    ("2004-10-07 11:19:34");
```

```
SELECT * FROM t1;
+-----+
| d      |
+-----+
| 2007-01-30 21:31:07 |
| 1983-10-15 06:42:51 |
| 2011-04-21 12:34:56 |
| 2011-10-30 06:31:41 |
| 2011-01-30 14:03:25 |
| 2004-10-07 11:19:34 |
+-----+
6 rows in set (0.02 sec)
```

```
SELECT YEARWEEK(d) FROM t1 WHERE YEAR(d) = 2011;
+-----+
| YEARWEEK(d) |
+-----+
| 201116      |
| 201144      |
| 201105      |
+-----+
3 rows in set (0.03 sec)
```

## 1.1.12.4 Aggregate Functions

### 1.1.12.4.1 Stored Aggregate Functions

MariaDB starting with [10.3.3](#)

The ability to create stored aggregate functions was added in [MariaDB 10.3.3](#).

#### Contents

- 1. Standard Syntax
  - 1. Using SQL/PL
- 2. Examples
  - 1. SQL/PL Example
- 3. See Also

[Aggregate functions](#) are functions that are computed over a sequence of rows and return one result for the sequence of rows.

Creating a custom aggregate function is done using the [CREATE FUNCTION](#) statement with two main differences:

- The addition of the AGGREGATE keyword, so

```
CREATE AGGREGATE FUNCTION
```

- The

```
FETCH GROUP NEXT ROW
```

instruction inside the loop

- Oracle PL/SQL compatibility using SQL/PL is provided

## Standard Syntax

```

CREATE AGGREGATE FUNCTION function_name (parameters) RETURNS return_type
BEGIN
    All types of declarations
    DECLARE CONTINUE HANDLER FOR NOT FOUND RETURN return_val;
    LOOP
        FETCH GROUP NEXT ROW; // fetches next row from table
        other instructions
    END LOOP;
END

```

Stored aggregate functions were a [2016 Google Summer of Code](#) project by Varun Gupta.

## Using SQL/PL

```

SET sql_mode=Oracle;
DELIMITER //

CREATE AGGREGATE FUNCTION function_name (parameters) RETURN return_type
    declarations
BEGIN
    LOOP
        FETCH GROUP NEXT ROW; -- fetches next row from table
        -- other instructions

    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN return_val;
END //

DELIMITER ;

```

## Examples

First a simplified example:

```

CREATE TABLE marks(stud_id INT, grade_count INT);

INSERT INTO marks VALUES (1,6), (2,4), (3,7), (4,5), (5,8);

SELECT * FROM marks;
+-----+-----+
| stud_id | grade_count |
+-----+-----+
|      1 |         6 |
|      2 |         4 |
|      3 |         7 |
|      4 |         5 |
|      5 |         8 |
+-----+-----+

DELIMITER //
CREATE AGGREGATE FUNCTION IF NOT EXISTS aggregate_count(x INT) RETURNS INT
BEGIN
    DECLARE count_students INT DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR NOT FOUND
    RETURN count_students;
    LOOP
        FETCH GROUP NEXT ROW;
        IF x THEN
            SET count_students = count_students+1;
        END IF;
    END LOOP;
END //
DELIMITER ;

```

A non-trivial example that cannot easily be rewritten using existing functions:

```

DELIMITER //
CREATE AGGREGATE FUNCTION medi_int(x INT) RETURNS DOUBLE
BEGIN
    DECLARE CONTINUE HANDLER FOR NOT FOUND
    BEGIN
        DECLARE res DOUBLE;
        DECLARE cnt INT DEFAULT (SELECT COUNT(*) FROM tt);
        DECLARE lim INT DEFAULT (cnt-1) DIV 2;
        IF cnt % 2 = 0 THEN
            SET res = (SELECT AVG(a) FROM (SELECT a FROM tt ORDER BY a LIMIT lim,2) ttt);
        ELSE
            SET res = (SELECT a FROM tt ORDER BY a LIMIT lim,1);
        END IF;
        DROP TEMPORARY TABLE tt;
        RETURN res;
    END;
    CREATE TEMPORARY TABLE tt (a INT);
    LOOP
        FETCH GROUP NEXT ROW;
        INSERT INTO tt VALUES (x);
    END LOOP;
END //
DELIMITER ;

```

## SQL/PL Example

This uses the same marks table as created above.

```

SET sql_mode=Oracle;
DELIMITER //

CREATE AGGREGATE FUNCTION aggregate_count(x INT) RETURN INT AS count_students INT DEFAULT 0;
BEGIN
    LOOP
        FETCH GROUP NEXT ROW;
        IF x THEN
            SET count_students := count_students+1;
        END IF;
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN count_students;
END aggregate_count //
DELIMITER ;

SELECT aggregate_count(stud_id) FROM marks;

```

## See Also

- [Stored Function Overview](#)
- [CREATE FUNCTION](#)
- [SHOW CREATE FUNCTION](#)
- [DROP FUNCTION](#)
- [Stored Routine Privileges](#)
- [SHOW FUNCTION STATUS](#)
- [Information Schema ROUTINES Table](#)

## 1.1.12.4.2 AVG

### Syntax

```
AVG([DISTINCT] expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

# Description

Returns the average value of expr. The DISTINCT option can be used to return the average of the distinct values of expr. NULL values are ignored. It is an [aggregate function](#), and so can be used with the [GROUP BY](#) clause.

AVG() returns NULL if there were no matching rows.

From [MariaDB 10.2.0](#), AVG() can be used as a [window function](#).

## Examples

```
CREATE TABLE sales (sales_value INT);

INSERT INTO sales VALUES(10),(20),(20),(40);

SELECT AVG(sales_value) FROM sales;
+-----+
| AVG(sales_value) |
+-----+
|      22.5000 |
+-----+

SELECT AVG(DISTINCT(sales_value)) FROM sales;
+-----+
| AVG(DISTINCT(sales_value)) |
+-----+
|          23.3333 |
+-----+
```

Commonly, AVG() is used with a [GROUP BY](#) clause:

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT name, AVG(score) FROM student GROUP BY name;
+-----+
| name    | AVG(score) |
+-----+
| Chun    | 74.0000 |
| Esben   | 37.0000 |
| Kaolin  | 72.0000 |
| Tatiana | 85.0000 |
+-----+
```

Be careful to avoid this common mistake, not grouping correctly and returning mismatched data:

```
SELECT name,test,AVG(score) FROM student;
+-----+
| name | test | MIN(score) |
+-----+
| Chun | SQL  |      31 |
+-----+
```

As a [window function](#):

```

CREATE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT name, test, score, AVG(score) OVER (PARTITION BY test)
    AS average_by_test FROM student_test;
+-----+-----+-----+-----+
| name   | test   | score | average_by_test |
+-----+-----+-----+-----+
| Chun   | SQL    | 75   | 65.2500 |
| Chun   | Tuning | 73   | 68.7500 |
| Esben  | SQL    | 43   | 65.2500 |
| Esben  | Tuning | 31   | 68.7500 |
| Kaolin | SQL    | 56   | 65.2500 |
| Kaolin | Tuning | 88   | 68.7500 |
| Tatiana| SQL    | 87   | 65.2500 |
| Tatiana| Tuning | 83   | 68.7500 |
+-----+-----+-----+-----+

```

## See Also

- [MAX](#) (maximum)
- [MIN](#) (minimum)
- [SUM](#) (sum total)

## 1.1.12.4.3 BIT\_AND

### Syntax

```
BIT_AND(expr) [over_clause]
```

### Description

Returns the bitwise AND of all bits in *expr*. The calculation is performed with 64-bit (`BIGINT`) precision. It is an [aggregate function](#), and so can be used with the `GROUP BY` clause.

If no rows match,

`BIT_AND`

will return a value with all bits set to 1. NULL values have no effect on the result unless all results are NULL, which is treated as no match.

From [MariaDB 10.2.0](#),

`BIT_AND`

can be used as a [window function](#) with the addition of the *over\_clause*.

### Examples

```

CREATE TABLE vals (x INT);

INSERT INTO vals VALUES(111),(110),(100);

SELECT BIT_AND(x), BIT_OR(x), BIT_XOR(x) FROM vals;
+-----+-----+-----+
| BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+-----+-----+-----+
|      100   |       111  |        101 |
+-----+-----+-----+

```

As an [aggregate function](#):

```

CREATE TABLE vals2 (category VARCHAR(1), x INT);

INSERT INTO vals2 VALUES
('a',111),('a',110),('a',100),
('b','000'),('b',001),('b',011);

SELECT category, BIT_AND(x), BIT_OR(x), BIT_XOR(x)
  FROM vals2 GROUP BY category;
+-----+-----+-----+
| category | BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+-----+-----+-----+
| a        |      100 |      111 |      101 |
| b        |       0 |       11 |       10 |
+-----+-----+-----+

```

No match:

```

SELECT BIT_AND(NULL);
+-----+
| BIT_AND(NULL) |
+-----+
| 18446744073709551615 |
+-----+

```

## See Also

- [BIT\\_OR](#)
- [BIT\\_XOR](#)

## 1.1.12.4.4 BIT\_OR

### Syntax

```
BIT_OR(expr) [over_clause]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the bitwise OR of all bits in

`expr`

. The calculation is performed with 64-bit ( `BIGINT` ) precision. It is an [aggregate function](#) , and so can be used with the `GROUP BY` clause.

If no rows match,

`BIT_OR`

will return a value with all bits set to

`0`

. `NULL` values have no effect on the result unless all results are `NULL`, which is treated as no match.

From [MariaDB 10.2.0](#) ,

`BIT_OR`

can be used as a [window function](#) with the addition of the `over_clause` .

### Examples

```

CREATE TABLE vals (x INT);

INSERT INTO vals VALUES(111),(110),(100);

SELECT BIT_AND(x), BIT_OR(x), BIT_XOR(x) FROM vals;
+-----+-----+-----+
| BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+-----+-----+-----+
|      100   |      111   |      101   |
+-----+-----+-----+

```

As an [aggregate function](#) :

```

CREATE TABLE vals2 (category VARCHAR(1), x INT);

INSERT INTO vals2 VALUES
('a',111),('a',110),('a',100),
('b','000'),('b',001),('b',011);

SELECT category, BIT_AND(x), BIT_OR(x), BIT_XOR(x)
  FROM vals2 GROUP BY category;
+-----+-----+-----+
| category | BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+-----+-----+-----+
| a        |      100   |      111   |      101   |
| b        |          0  |      11    |      10    |
+-----+-----+-----+

```

No match:

```

SELECT BIT_OR(NULL);
+-----+
| BIT_OR(NULL) |
+-----+
|          0  |
+-----+

```

## See Also

- [BIT\\_AND](#)
- [BIT\\_XOR](#)

## 1.1.12.4.5 BIT\_XOR

### Syntax

```
BIT_XOR(expr) [over_clause]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the bitwise XOR of all bits in

expr

. The calculation is performed with 64-bit ( [BIGINT](#) ) precision. It is an [aggregate function](#) , and so can be used with the [GROUP BY](#) clause.

If no rows match,

BIT\_XOR

will return a value with all bits set to

0

. NULL values have no effect on the result unless all results are NULL, which is treated as no match.

From [MariaDB 10.2.0](#) ,

BIT\_XOR

can be used as a [window function](#) with the addition of the [over\\_clause](#) .

## Examples

```
CREATE TABLE vals (x INT);

INSERT INTO vals VALUES(111),(110),(100);

SELECT BIT_AND(x), BIT_OR(x), BIT_XOR(x) FROM vals;
+-----+-----+-----+
| BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+-----+-----+-----+
|      100   |      111   |      101   |
+-----+-----+-----+
```

As an aggregate function :

```
CREATE TABLE vals2 (category VARCHAR(1), x INT);

INSERT INTO vals2 VALUES
    ('a',111),('a',110),('a',100),
    ('b','000'),('b',001),('b',011);

SELECT category, BIT_AND(x), BIT_OR(x), BIT_XOR(x)
    FROM vals2 GROUP BY category;
+-----+-----+-----+
| category | BIT_AND(x) | BIT_OR(x) | BIT_XOR(x) |
+-----+-----+-----+
| a        |      100   |      111   |      101   |
| b        |          0  |       11    |       10    |
+-----+-----+-----+
```

No match:

```
SELECT BIT_XOR(NULL);
+-----+
| BIT_XOR(NULL) |
+-----+
|          0   |
+-----+
```

## See Also

- [BIT\\_AND](#)
- [BIT\\_OR](#)

## 1.1.12.4.6 COUNT

### Syntax

```
COUNT(expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns a count of the number of non-NULL values of expr in the rows retrieved by a `SELECT` statement. The result is a `BIGINT` value. It is an [aggregate function](#), and so can be used with the `GROUP BY` clause.

`COUNT(*)` counts the total number of rows in a table.

`COUNT()` returns 0 if there were no matching rows.

From [MariaDB 10.2.0](#), `COUNT()` can be used as a [window function](#).

## Examples

```

CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT COUNT(*) FROM student;
+-----+
| COUNT(*) |
+-----+
|      8   |
+-----+

```

COUNT(DISTINCT) example:

```

SELECT COUNT(DISTINCT name) FROM student;
+-----+
| COUNT(DISTINCT name) |
+-----+
|          4           |
+-----+

```

As a window function

```

CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
('Tatiana', 'SQL', 87);

SELECT name, test, score, COUNT(score) OVER (PARTITION BY name)
  AS tests_written FROM student_test;
+-----+-----+-----+-----+
| name    | test   | score | tests_written |
+-----+-----+-----+-----+
| Chun    | SQL    | 75   |        2       |
| Chun    | Tuning | 73   |        2       |
| Esben   | SQL    | 43   |        2       |
| Esben   | Tuning | 31   |        2       |
| Kaolin  | SQL    | 56   |        2       |
| Kaolin  | Tuning | 88   |        2       |
| Tatiana | SQL    | 87   |        1       |
+-----+-----+-----+-----+

```

## See Also

- [SELECT](#)
- [COUNT DISTINCT](#)
- [Window Functions](#)

## 1.1.12.4.7 COUNT DISTINCT

### Syntax

```
COUNT(DISTINCT expr,[expr...])
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns a count of the number of different non-NULL values.

COUNT(DISTINCT) returns 0 if there were no matching rows.

Although, from MariaDB 10.2.0 , COUNT can be used as a window function , COUNT DISTINCT cannot be.

## Examples

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT COUNT(*) FROM student;
+-----+
| COUNT(*) |
+-----+
|      8   |
+-----+

SELECT COUNT(DISTINCT (name)) FROM student;
+-----+
| COUNT(DISTINCT (name)) |
+-----+
|             4          |
+-----+
```

## See Also

- [SELECT](#)
- [COUNT](#)

## 1.1.12.4.8 GROUP\_CONCAT

### Syntax

```
GROUP_CONCAT(expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
  - 1. [LIMIT](#)
3. [Examples](#)
4. [See Also](#)

### Description

This function returns a string result with the concatenated non-NULL values from a group. It returns NULL if there are no non-NULL values.

The maximum returned length in bytes is determined by the `group_concat_max_len` server system variable, which defaults to 1M (>= MariaDB 10.2.4 ) or 1K (<= MariaDB 10.2.3 ).

If `group_concat_max_len <= 512`, the return type is `VARBINARY` or `VARCHAR` ; otherwise, the return type is `BLOB` or `TEXT` . The choice between binary or non-binary types depends from the input.

The full syntax is as follows:

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
            [ORDER BY {unsigned_integer | col_name | expr}
             [ASC | DESC] [,col_name ...]]
            [SEPARATOR str_val]
            [LIMIT {[offset,] row_count | row_count OFFSET offset}])
```

`DISTINCT`

eliminates duplicate values from the output string.

`ORDER BY` determines the order of returned values.

SEPARATOR  
specifies a separator between the values. The default separator is a comma ( , ). It is possible to avoid using a separator by specifying an empty string.

## LIMIT

MariaDB starting with 10.3.3

Until MariaDB 10.3.2 , it was not possible to use the LIMIT clause with GROUP\_CONCAT . This restriction was lifted in MariaDB 10.3.3 .

## Examples

```
SELECT student_name,  
       GROUP_CONCAT(test_score)  
  FROM student  
 GROUP BY student_name;
```

Get a readable list of MariaDB users from the mysql.user table:

```
SELECT GROUP_CONCAT(DISTINCT User ORDER BY User SEPARATOR '\n')  
      FROM mysql.user;
```

In the former example,

DISTINCT  
is used because the same user may occur more than once. The new line ( \n ) used as a  
SEPARATOR  
makes the results easier to read.

Get a readable list of hosts from which each user can connect:

```
SELECT User, GROUP_CONCAT(Host ORDER BY Host SEPARATOR ', ')  
      FROM mysql.user GROUP BY User ORDER BY User;
```

The former example shows the difference between the

GROUP\_CONCAT  
's ORDER BY (which sorts the concatenated hosts), and the  
SELECT  
's ORDER BY (which sorts the rows).

From MariaDB 10.3.3 , LIMIT can be used with

GROUP\_CONCAT  
, so, for example, given the following table:

```
CREATE TABLE d (dd DATE, cc INT);  
  
INSERT INTO d VALUES ('2017-01-01',1);  
INSERT INTO d VALUES ('2017-01-02',2);  
INSERT INTO d VALUES ('2017-01-04',3);
```

the following query:

```
SELECT SUBSTRING_INDEX(GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC),",",1) FROM d;  
+-----+  
| SUBSTRING_INDEX(GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC),",",1) |  
+-----+  
| 2017-01-04:3 |  
+-----+
```

can be more simply rewritten as:

```

SELECT GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC LIMIT 1) FROM d;
+-----+
| GROUP_CONCAT(CONCAT_WS(":",dd,cc) ORDER BY cc DESC LIMIT 1) |
+-----+
| 2017-01-04:3 |
+-----+

```

## See Also

- [CONCAT\(\)](#)
- [CONCAT\\_WS\(\)](#)
- [SELECT](#)
- [ORDER BY](#)

## 1.1.12.4.9 JSON\_ARRAYAGG

MariaDB starting with [10.5.0](#)

`JSON_ARRAYAGG` was added in [MariaDB 10.5.0](#).

## Syntax

```
JSON_ARRAYAGG(column_or_expression)
```

## Description

`JSON_ARRAYAGG`

returns a JSON array containing an element for each value in a given set of JSON or SQL values. It acts on a column or an expression that evaluates to a single value.

Returns NULL in the case of an error, or if the result contains no rows.

`JSON_ARRAYAGG`

cannot currently be used as a [window function](#).

The full syntax is as follows:

```

JSON_ARRAYAGG([DISTINCT] expr [,expr ...]
              [ORDER BY {unsigned_integer | col_name | expr}
               [ASC | DESC] [,col_name ...]]
              [LIMIT {[offset,] row_count | row_count OFFSET offset}])

```

## Examples

```

CREATE TABLE t1 (a INT, b INT);

INSERT INTO t1 VALUES (1, 1),(2, 1), (1, 1),(2, 1), (3, 2),(2, 2),(2, 2);

SELECT JSON_ARRAYAGG(a), JSON_ARRAYAGG(b) FROM t1;
+-----+-----+
| JSON_ARRAYAGG(a) | JSON_ARRAYAGG(b) |
+-----+-----+
| [1,2,1,2,3,2,2] | [1,1,1,2,2,2] |
+-----+-----+

SELECT JSON_ARRAYAGG(a), JSON_ARRAYAGG(b) FROM t1 GROUP BY b;
+-----+-----+
| JSON_ARRAYAGG(a) | JSON_ARRAYAGG(b) |
+-----+-----+
| [1,2,1,2]        | [1,1,1,1]          |
| [3,2,2,2]        | [2,2,2,2]          |
+-----+-----+

```

## 1.1.12.4.10 JSON\_OBJECTAGG

MariaDB starting with 10.5.0

JSON\_OBJECTAGG was added in MariaDB 10.5.0 .

## Syntax

```
JSON_OBJECTAGG(key, value)
```

## Description

JSON\_OBJECTAGG

returns a JSON object containing key-value pairs. It takes two expressions that evaluate to a single value, or two column names, as arguments, the first used as a key, and the second as a value.

Returns NULL in the case of an error, or if the result contains no rows.

JSON\_OBJECTAGG

cannot currently be used as a [window function](#) .

## Examples

```
select * from t1;
+-----+
| a      | b      |
+-----+
| 1      | Hello  |
| 1      | World  |
| 2      | This   |
+-----+

SELECT JSON_OBJECTAGG(a, b) FROM t1;
+-----+
| JSON_OBJECTAGG(a, b)           |
+-----+
| {"1":"Hello", "1":"World", "2":"This"} |
+-----+
```

## 1.1.12.4.11 MAX

### Syntax

```
MAX([DISTINCT] expr)
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the largest, or maximum, value of

*expr*

MAX()

can also take a string argument in which case it returns the maximum string value. The

DISTINCT

keyword can be used to find the maximum of the distinct values of

*expr*

, however, this produces the same result as omitting

DISTINCT

Note that **SET** and **ENUM** fields are currently compared by their string value rather than their relative position in the set, so **MAX()** may produce a different highest result than **ORDER BY DESC**.

It is an [aggregate function](#), and so can be used with the [GROUP BY](#) clause.

From [MariaDB 10.2.2](#), **MAX()** can be used as a [window function](#).

```
MAX()  
returns  
NULL  
if there were no matching rows.
```

## Examples

```
CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);  
  
INSERT INTO student VALUES  
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),  
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),  
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),  
('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);  
  
SELECT name, MAX(score) FROM student GROUP BY name;  
+-----+  
| name | MAX(score) |  
+-----+  
| Chun | 75 |  
| Esben | 43 |  
| Kaolin | 88 |  
| Tatiana | 87 |  
+-----+
```

MAX string:

```
SELECT MAX(name) FROM student;  
+-----+  
| MAX(name) |  
+-----+  
| Tatiana |  
+-----+
```

Be careful to avoid this common mistake, not grouping correctly and returning mismatched data:

```
SELECT name,test,MAX(SCORE) FROM student;  
+-----+-----+  
| name | test | MAX(SCORE) |  
+-----+-----+  
| Chun | SQL | 88 |  
+-----+-----+
```

Difference between **ORDER BY DESC** and **MAX()**:

```
CREATE TABLE student2(name CHAR(10),grade ENUM('b','c','a'));  
  
INSERT INTO student2 VALUES('Chun','b'),('Esben','c'),('Kaolin','a');  
  
SELECT MAX(grade) FROM student2;  
+-----+  
| MAX(grade) |  
+-----+  
| c |  
+-----+  
  
SELECT grade FROM student2 ORDER BY grade DESC LIMIT 1;  
+-----+  
| grade |  
+-----+  
| a |  
+-----+
```

As a [window function](#):

```

CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);
INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, MAX(score)
    OVER (PARTITION BY name) AS highest_score FROM student_test;
+-----+-----+-----+-----+
| name | test | score | highest_score |
+-----+-----+-----+-----+
| Chun | SQL | 75 | 75 |
| Chun | Tuning | 73 | 75 |
| Esben | SQL | 43 | 43 |
| Esben | Tuning | 31 | 43 |
| Kaolin | SQL | 56 | 88 |
| Kaolin | Tuning | 88 | 88 |
| Tatiana | SQL | 87 | 87 |
+-----+-----+-----+-----+

```

## See Also

- [AVG](#) (average)
- [MIN](#) (minimum)
- [SUM](#) (sum total)
- [MIN/MAX optimization](#) used by the optimizer
- [GREATEST\(\)](#) returns the largest value from a list

## 1.1.12.4.12 MIN

### Syntax

```
MIN([DISTINCT] expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the minimum value of

*expr*

`MIN()`

may take a string argument, in which case it returns the minimum string value. The

`DISTINCT`

keyword can be used to find the minimum of the distinct values of

*expr*

, however, this produces the same result as omitting

`DISTINCT`

Note that `SET` and `ENUM` fields are currently compared by their string value rather than their relative position in the set, so `MIN()` may produce a different lowest result than `ORDER BY ASC`.

It is an [aggregate function](#) , and so can be used with the [GROUP BY](#) clause.

From [MariaDB 10.2.2](#) , `MIN()` can be used as a [window function](#) .

```

MIN()
returns
NULL
if there were no matching rows.

```

### Examples

```

CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

SELECT name, MIN(score) FROM student GROUP BY name;
+-----+-----+
| name | MIN(score) |
+-----+-----+
| Chun |      73 |
| Esben |     31 |
| Kaolin |    56 |
| Tatiana |   83 |
+-----+-----+

```

MIN() with a string:

```

SELECT MIN(name) FROM student;
+-----+
| MIN(name) |
+-----+
| Chun      |
+-----+

```

Be careful to avoid this common mistake, not grouping correctly and returning mismatched data:

```

SELECT name,test,MIN(score) FROM student;
+-----+-----+
| name | test | MIN(score) |
+-----+-----+
| Chun | SQL |      31 |
+-----+-----+

```

Difference between ORDER BY ASC and MIN():

```

CREATE TABLE student2(name CHAR(10),grade ENUM('b','c','a'));

INSERT INTO student2 VALUES('Chun','b'),('Esben','c'),('Kaolin','a');

SELECT MIN(grade) FROM student2;
+-----+
| MIN(grade) |
+-----+
| a          |
+-----+

SELECT grade FROM student2 ORDER BY grade ASC LIMIT 1;
+-----+
| grade |
+-----+
| b      |
+-----+

```

As a [window function](#):

```

CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);
INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, MIN(score)
    OVER (PARTITION BY name) AS lowest_score FROM student_test;
+-----+-----+-----+-----+
| name | test | score | lowest_score |
+-----+-----+-----+-----+
| Chun | SQL | 75 | 73 |
| Chun | Tuning | 73 | 73 |
| Esben | SQL | 43 | 31 |
| Esben | Tuning | 31 | 31 |
| Kaolin | SQL | 56 | 56 |
| Kaolin | Tuning | 88 | 56 |
| Tatiana | SQL | 87 | 87 |
+-----+-----+-----+-----+

```

## See Also

- [AVG](#) (average)
- [MAX](#) (maximum)
- [SUM](#) (sum total)
- [MIN/MAX optimization](#) used by the optimizer
- [LEAST\(\)](#) returns the smallest value from a list.

## 1.1.12.4.13 STD

### Syntax

`STD(expr)`

### Description

Returns the population standard deviation of

*expr*  
. This is an extension to standard SQL. The standard SQL function

[STDDEV\\_POP\(\)](#)

can be used instead.

It is an [aggregate function](#), and so can be used with the [GROUP BY](#) clause.

From [MariaDB 10.2.2](#), `STD()` can be used as a [window function](#).

This function returns

`NULL`  
if there were no matching rows.

## Examples

As an [aggregate function](#):

```

CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
('a',1),('a',2),('a',3),
('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+-----+-----+-----+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+-----+-----+-----+
| a        |      0.8165 |      1.0000 |     0.6667 |
| b        |    18.0400 |    20.1693 |  325.4400 |
+-----+-----+-----+

```

As a window function :

```

CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
('Tatiana', 'SQL', 87);

SELECT name, test, score, STDDEV_POP(score)
  OVER (PARTITION BY test) AS stddev_results FROM student_test;
+-----+-----+-----+-----+
| name   | test   | score | stddev_results |
+-----+-----+-----+-----+
| Chun   | SQL    | 75   |      16.9466 |
| Chun   | Tuning | 73   |      24.1247 |
| Esben  | SQL    | 43   |      16.9466 |
| Esben  | Tuning | 31   |      24.1247 |
| Kaolin | SQL    | 56   |      16.9466 |
| Kaolin | Tuning | 88   |      24.1247 |
| Tatiana| SQL    | 87   |      16.9466 |
+-----+-----+-----+-----+

```

## See Also

- [STDDEV\\_POP](#) (equivalent, standard SQL)
- [STDDEV](#) (equivalent, Oracle-compatible non-standard SQL)
- [VAR\\_POP](#) (variance)
- [STDDEV\\_SAMP](#) (sample standard deviation)

## 1.1.12.4.14 STDDEV

### Syntax

`STDDEV(expr)`

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the population standard deviation of

*expr*

. This function is provided for compatibility with Oracle. The standard SQL function

[STDDEV\\_POP\(\)](#)

can be used instead.

It is an [aggregate function](#), and so can be used with the [GROUP BY](#) clause.

From MariaDB 10.2.2 , STDDEV() can be used as a [window function](#) .

This function returns

NULL

if there were no matching rows.

## Examples

As an [aggregate function](#) :

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
('a',1),('a',2),('a',3),
('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+-----+-----+-----+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+-----+-----+-----+
| a        |      0.8165 |       1.0000 |     0.6667 |
| b        |    18.0400 |    20.1693 | 325.4400 |
+-----+-----+-----+
```

As a [window function](#) :

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
('Tatiana', 'SQL', 87);

SELECT name, test, score, STDDEV_POP(score)
  OVER (PARTITION BY test) AS stddev_results FROM student_test;
+-----+-----+-----+-----+
| name   | test   | score | stddev_results |
+-----+-----+-----+-----+
| Chun   | SQL    | 75   | 16.9466 |
| Chun   | Tuning | 73   | 24.1247 |
| Esben  | SQL    | 43   | 16.9466 |
| Esben  | Tuning | 31   | 24.1247 |
| Kaolin | SQL    | 56   | 16.9466 |
| Kaolin | Tuning | 88   | 24.1247 |
| Tatiana| SQL    | 87   | 16.9466 |
+-----+-----+-----+-----+
```

## See Also

- [STDDEV\\_POP](#) (equivalent, standard SQL)
- [STD](#) (equivalent, non-standard SQL)
- [VAR\\_POP](#) (variance)
- [STDDEV\\_SAMP](#) (sample standard deviation)

## 1.1.12.4.15 STDDEV\_POP

### Syntax

```
STDDEV_POP(expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the population standard deviation of

*expr*  
(the square root of

`VAR_POP()`

). You can also use

`STD()`

or

`STDDEV()`

, which are equivalent but not standard SQL.

It is an [aggregate function](#) , and so can be used with the `GROUP BY` clause.

From [MariaDB 10.2.2](#) , `STDDEV_POP()` can be used as a [window function](#) .

`STDDEV_POP()` returns

`NULL`

if there were no matching rows.

## Examples

As an [aggregate function](#) :

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
  ('a',1),('a',2),('a',3),
  ('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+-----+-----+-----+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+-----+-----+-----+
| a        |     0.8165 |      1.0000 |    0.6667 |
| b        |   18.0400 |    20.1693 | 325.4400 |
+-----+-----+-----+
```

As a [window function](#) :

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
  ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
  ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
  ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
  ('Tatiana', 'SQL', 87);

SELECT name, test, score, STDDEV_POP(score)
  OVER (PARTITION BY test) AS stddev_results FROM student_test;
+-----+-----+-----+
| name   | test   | score | stddev_results |
+-----+-----+-----+
| Chun   | SQL    | 75   |    16.9466 |
| Chun   | Tuning | 73   |    24.1247 |
| Esben  | SQL    | 43   |    16.9466 |
| Esben  | Tuning | 31   |    24.1247 |
| Kaolin | SQL    | 56   |    16.9466 |
| Kaolin | Tuning | 88   |    24.1247 |
| Tatiana| SQL    | 87   |    16.9466 |
+-----+-----+-----+
```

## See Also

- [STD](#) (equivalent, non-standard SQL)
- [STDDEV](#) (equivalent, Oracle-compatible non-standard SQL)
- [VAR\\_POP](#) (variance)
- [STDDEV\\_SAMP](#) (sample standard deviation)

## 1.1.12.4.16 STDDEV\_SAMP

### Syntax

```
STDDEV_SAMP(expr)
```

### Description

Returns the sample standard deviation of

expr  
(the square root of [VAR\\_SAMP\(\)](#) ).

It is an [aggregate function](#) , and so can be used with the [GROUP BY](#) clause.

From [MariaDB 10.2.2](#) , STDDEV\_SAMP() can be used as a [window function](#) .

STDDEV\_SAMP() returns

NULL  
if there were no matching rows.

## 1.1.12.4.17 SUM

### Syntax

```
SUM([DISTINCT] expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the sum of

expr  
. If the return set has no rows,  
SUM()  
returns  
NULL  
. The  
DISTINCT  
keyword can be used to sum only the distinct values of  
expr  
.

From [MariaDB 10.2.0](#) , SUM() can be used as a [window function](#) , although not with the DISTINCT specifier.

### Examples

```
CREATE TABLE sales (sales_value INT);
INSERT INTO sales VALUES(10),(20),(20),(40);

SELECT SUM(sales_value) FROM sales;
+-----+
| SUM(sales_value) |
+-----+
|          90      |
+-----+

SELECT SUM(DISTINCT(sales_value)) FROM sales;
+-----+
| SUM(DISTINCT(sales_value)) |
+-----+
|          70      |
+-----+
```

Commonly, SUM is used with a [GROUP BY](#) clause:

```
CREATE TABLE sales (name CHAR(10), month CHAR(10), units INT);

INSERT INTO sales VALUES
    ('Chun', 'Jan', 75), ('Chun', 'Feb', 73),
    ('Esben', 'Jan', 43), ('Esben', 'Feb', 31),
    ('Kaolin', 'Jan', 56), ('Kaolin', 'Feb', 88),
    ('Tatiana', 'Jan', 87), ('Tatiana', 'Feb', 83);

SELECT name, SUM(units) FROM sales GROUP BY name;
+-----+
| name | SUM(units) |
+-----+
| Chun |      148 |
| Esben |       74 |
| Kaolin |     144 |
| Tatiana |    170 |
+-----+
```

The [GROUP BY](#) clause is required when using an aggregate function along with regular column data, otherwise the result will be a mismatch, as in the following common type of mistake:

```
SELECT name, SUM(units) FROM sales
+-----+
| name | SUM(units) |
+-----+
| Chun |      536 |
+-----+
```

As a [window function](#):

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);
INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, SUM(score) OVER (PARTITION BY name) AS total_score FROM student_test;
+-----+
| name | test | score | total_score |
+-----+
| Chun | SQL | 75 | 148 |
| Chun | Tuning | 73 | 148 |
| Esben | SQL | 43 | 74 |
| Esben | Tuning | 31 | 74 |
| Kaolin | SQL | 56 | 144 |
| Kaolin | Tuning | 88 | 144 |
| Tatiana | SQL | 87 | 87 |
+-----+
```

## See Also

- [AVG](#) (average)
- [MAX](#) (maximum)
- [MIN](#) (minimum)

## 1.1.12.4.18 VARIANCE

### Syntax

```
VARIANCE(expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

# Description

Returns the population standard variance of

expr

. This is an extension to standard SQL. The standard SQL function `VAR_POP()` can be used instead.

Variance is calculated by

- working out the mean for the set
- for each number, subtracting the mean and squaring the result
- calculate the average of the resulting differences

It is an [aggregate function](#), and so can be used with the `GROUP BY` clause.

From [MariaDB 10.2.2](#), `VARIANCE()` can be used as a [window function](#).

`VARIANCE()` returns

`NULL`

if there were no matching rows.

## Examples

```
CREATE TABLE v(i tinyint);

INSERT INTO v VALUES(101),(99);

SELECT VARIANCE(i) FROM v;
+-----+
| VARIANCE(i) |
+-----+
|      1.0000 |
+-----+

INSERT INTO v VALUES(120),(80);

SELECT VARIANCE(i) FROM v;
+-----+
| VARIANCE(i) |
+-----+
|    200.5000 |
+-----+
```

As an [aggregate function](#):

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
('a',1),('a',2),('a',3),
('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+-----+-----+-----+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+-----+-----+-----+
| a        |      0.8165 |      1.0000 |     0.6667 |
| b        |     18.0400 |     20.1693 |  325.4400 |
+-----+-----+-----+
```

As a [window function](#):

```

CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, VAR_POP(score)
    OVER (PARTITION BY test) AS variance_results FROM student_test;
+-----+-----+-----+-----+
| name   | test   | score | variance_results |
+-----+-----+-----+-----+
| Chun   | SQL    | 75   | 287.1875 |
| Chun   | Tuning | 73   | 582.0000 |
| Esben  | SQL    | 43   | 287.1875 |
| Esben  | Tuning | 31   | 582.0000 |
| Kaolin | SQL    | 56   | 287.1875 |
| Kaolin | Tuning | 88   | 582.0000 |
| Tatiana| SQL    | 87   | 287.1875 |
+-----+-----+-----+-----+

```

## See Also

- [VAR\\_POP](#) (equivalent, standard SQL)
- [STDDEV\\_POP](#) (population standard deviation)
- [STDDEV\\_SAMP](#) (sample standard deviation)

## 1.1.12.4.19 VAR\_POP

### Syntax

`VAR_POP(expr)`

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the population standard variance of

`expr`

. It considers rows as the whole population, not as a sample, so it has the number of rows as the denominator. You can also use [VARIANCE\(\)](#), which is equivalent but is not standard SQL.

Variance is calculated by

- working out the mean for the set
- for each number, subtracting the mean and squaring the result
- calculate the average of the resulting differences

It is an [aggregate function](#), and so can be used with the [GROUP BY](#) clause.

From [MariaDB 10.2.2](#), `VAR_POP()` can be used as a [window function](#).

`VAR_POP()` returns

`NULL`

if there were no matching rows.

### Examples

```

CREATE TABLE v(i tinyint);

INSERT INTO v VALUES(101),(99);

SELECT VAR_POP(i) FROM v;
+-----+
| VAR_POP(i) |
+-----+
| 1.0000 |
+-----+

INSERT INTO v VALUES(120),(80);

SELECT VAR_POP(i) FROM v;
+-----+
| VAR_POP(i) |
+-----+
| 200.5000 |
+-----+

```

As an [aggregate function](#) :

```

CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
('a',1),('a',2),('a',3),
('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
  FROM stats GROUP BY category;
+-----+-----+-----+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+-----+-----+-----+
| a        |      0.8165 |       1.0000 |     0.6667 |
| b        |    18.0400 |    20.1693 | 325.4400 |
+-----+-----+-----+

```

As a [window function](#) :

```

CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
('Tatiana', 'SQL', 87);

SELECT name, test, score, VAR_POP(score)
  OVER (PARTITION BY test) AS variance_results FROM student_test;
+-----+-----+-----+-----+
| name   | test   | score | variance_results |
+-----+-----+-----+-----+
| Chun   | SQL    | 75   | 287.1875 |
| Esben  | SQL    | 43   | 287.1875 |
| Kaolin | SQL    | 56   | 287.1875 |
| Tatiana| SQL    | 87   | 287.1875 |
| Chun   | Tuning | 73   | 582.0000 |
| Esben  | Tuning | 31   | 582.0000 |
| Kaolin | Tuning | 88   | 582.0000 |
+-----+-----+-----+-----+

```

## See Also

- [VARIANCE](#) (equivalent, non-standard SQL)
- [STDDEV\\_POP](#) (population standard deviation)
- [STDDEV\\_SAMP](#) (sample standard deviation)

## 1.1.12.4.20 VAR\_SAMP

### Syntax

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Returns the sample variance of

*expr*

. That is, the denominator is the number of rows minus one.

It is an [aggregate function](#), and so can be used with the [GROUP BY](#) clause.

From [MariaDB 10.2.2](#), VAR\_SAMP() can be used as a [window function](#).

VAR\_SAMP() returns

NULL

if there were no matching rows.

## Examples

As an [aggregate function](#):

```
CREATE OR REPLACE TABLE stats (category VARCHAR(2), x INT);

INSERT INTO stats VALUES
    ('a',1),('a',2),('a',3),
    ('b',11),('b',12),('b',20),('b',30),('b',60);

SELECT category, STDDEV_POP(x), STDDEV_SAMP(x), VAR_POP(x)
    FROM stats GROUP BY category;
+-----+-----+-----+
| category | STDDEV_POP(x) | STDDEV_SAMP(x) | VAR_POP(x) |
+-----+-----+-----+
| a        |      0.8165 |      1.0000 |     0.6667 |
| b        |     18.0400 |     20.1693 |  325.4400 |
+-----+-----+-----+
```

As a [window function](#):

```
CREATE OR REPLACE TABLE student_test (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student_test VALUES
    ('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
    ('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
    ('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
    ('Tatiana', 'SQL', 87);

SELECT name, test, score, VAR_SAMP(score)
    OVER (PARTITION BY test) AS variance_results FROM student_test;
+-----+-----+-----+-----+
| name   | test   | score | variance_results |
+-----+-----+-----+-----+
| Chun   | SQL    | 75   |      382.9167 |
| Chun   | Tuning | 73   |      873.0000 |
| Esben  | SQL    | 43   |      382.9167 |
| Esben  | Tuning | 31   |      873.0000 |
| Kaolin | SQL    | 56   |      382.9167 |
| Kaolin | Tuning | 88   |      873.0000 |
| Tatiana| SQL    | 87   |      382.9167 |
+-----+-----+-----+-----+
```

## See Also

- [VAR\\_POP](#) (variance)
- [STDDEV\\_POP](#) (population standard deviation)

## 1.1.12.5 Numeric Functions

## 1.1.12.5.1 Addition Operator (+)

### Syntax

```
+
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Addition.

If both operands are integers, the result is calculated with [BIGINT](#) precision. If either integer is unsigned, the result is also an unsigned integer.

For real or string operands, the operand with the highest precision determines the result precision.

### Examples

```
SELECT 3+5;
+-----+
| 3+5 |
+-----+
|   8 |
+-----+
```

### See Also

- [Type Conversion](#)
- [Subtraction Operator \(-\)](#)
- [Multiplication Operator \(\\*\)](#)
- [Division Operator \(/\)](#)

## 1.1.12.5.2 Subtraction Operator (-)

### Syntax

```
-
```

### Description

Subtraction. The operator is also used as the unary minus for changing sign.

If both operands are integers, the result is calculated with [BIGINT](#) precision. If either integer is unsigned, the result is also an unsigned integer, unless the [NO\\_UNSIGNED\\_SUBTRACTION SQL MODE](#) is enabled, in which case the result is always signed.

For real or string operands, the operand with the highest precision determines the result precision.

### Examples

```

SELECT 96-9;
+-----+
| 96-9 |
+-----+
|   87 |
+-----+

SELECT 15-17;
+-----+
| 15-17 |
+-----+
|    -2 |
+-----+

SELECT 3.66 + 1.333;
+-----+
| 3.66 + 1.333 |
+-----+
|      4.993 |
+-----+

```

Unary minus:

```

SELECT - (3+5);
+-----+
| - (3+5) |
+-----+
|    -8 |
+-----+

```

## See Also

- [Type Conversion](#)
- [Addition Operator \(+\)](#)
- [Multiplication Operator \(\\*\)](#)
- [Division Operator \(/\)](#)

## 1.1.12.5.3 Division Operator (/)

### Syntax

/

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Division operator. Dividing by zero will return NULL. By default, returns four digits after the decimal. This is determined by the server system variable `div_precision_increment` which by default is four. It can be set from 0 to 30.

Dividing by zero returns

```

NULL
. If the
ERROR_ON_DIVISION_BY_ZERO
SQL_MODE is used (the default since MariaDB 10.2.4 ), a division by zero also produces a warning.

```

### Examples

```

SELECT 4/5;
+-----+
| 4/5   |
+-----+
| 0.8000 |
+-----+

SELECT 300/(2-2);
+-----+
| 300/(2-2) |
+-----+
|      NULL  |
+-----+

SELECT 300/7;
+-----+
| 300/7   |
+-----+
| 42.8571  |
+-----+

```

Changing `div_precision_increment` for the session from the default of four to six:

```

SET div_precision_increment = 6;

SELECT 300/7;
+-----+
| 300/7   |
+-----+
| 42.857143 |
+-----+

SELECT 300/7;
+-----+
| 300/7   |
+-----+
| 42.857143 |
+-----+

```

## See Also

- [Type Conversion](#)
- [Addition Operator \(+\)](#)
- [Subtraction Operator \(-\)](#)
- [Multiplication Operator \(\\*\)](#)

## 1.1.12.5.4 Multiplication Operator (\*)

### Syntax

\*

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Multiplication operator.

### Examples

```

SELECT 7*6;
+-----+
| 7*6 |
+-----+
|   42 |
+-----+

SELECT 1234567890*9876543210;
+-----+
| 1234567890*9876543210 |
+-----+
| -6253480962446024716 |
+-----+

SELECT 18014398509481984*18014398509481984.0;
+-----+
| 18014398509481984*18014398509481984.0 |
+-----+
| 324518553658426726783156020576256.0 |
+-----+

SELECT 18014398509481984*18014398509481984;
+-----+
| 18014398509481984*18014398509481984 |
+-----+
|          0 |
+-----+

```

## See Also

- [Type Conversion](#)
- [Addition Operator \(+\)](#)
- [Subtraction Operator \(-\)](#)
- [Division Operator \(/\)](#)

## 1.1.12.5.5 Modulo Operator (%)

### Syntax

N % M

### Description

Modulo operator. Returns the remainder of

N  
divided by  
M  
. See also [MOD](#) .

### Examples

```

SELECT 1042 % 50;
+-----+
| 1042 % 50 |
+-----+
|      42 |
+-----+

```

## 1.1.12.5.6 DIV

### Syntax

DIV

## Description

Integer division. Similar to [FLOOR\(\)](#), but is safe with **BIGINT** values. Incorrect results may occur for non-integer operands that exceed BIGINT range.

If the

`ERROR_ON_DIVISION_BY_ZERO`

`SQL_MODE` is used, a division by zero produces an error. Otherwise, it returns NULL.

The remainder of a division can be obtained using the [MOD](#) operator.

## Examples

```
SELECT 300 DIV 7;
+-----+
| 300 DIV 7 |
+-----+
|        42 |
+-----+  
  
SELECT 300 DIV 0;
+-----+
| 300 DIV 0 |
+-----+
|      NULL |
+-----+
```

## 1.1.12.5.7 ABS

### Syntax

`ABS(X)`

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the absolute (non-negative) value of

`X`  
. If  
`X`

is not a number, it is converted to a numeric type.

## Examples

```
SELECT ABS(42);
+-----+
| ABS(42) |
+-----+
|      42 |
+-----+

SELECT ABS(-42);
+-----+
| ABS(-42) |
+-----+
|      42 |
+-----+

SELECT ABS(DATE '1994-01-01');
+-----+
| ABS(DATE '1994-01-01') |
+-----+
|          19940101 |
+-----+
```

## See Also

- [SIGN\(\)](#)

## 1.1.12.5.8 ACOS

### Syntax

ACOS(X)

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

### Description

Returns the arc cosine of

X  
, that is, the value whose cosine is  
X  
. Returns  
NULL  
if  
X  
is not in the range  
-1  
to  
1  
.

### Examples

```

SELECT ACOS(1);
+-----+
| ACOS(1) |
+-----+
|      0   |
+-----+

SELECT ACOS(1.0001);
+-----+
| ACOS(1.0001) |
+-----+
|      NULL    |
+-----+

SELECT ACOS(0);
+-----+
| ACOS(0)      |
+-----+
| 1.5707963267949 |
+-----+

SELECT ACOS(0.234);
+-----+
| ACOS(0.234)    |
+-----+
| 1.33460644244679 |
+-----+

```

## 1.1.12.5.9 ASIN

### Syntax

ASIN(X)

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

### Description

Returns the arc sine of X, that is, the value whose sine is X. Returns NULL if X is not in the range -1 to 1.

### Examples

```

SELECT ASIN(0.2);
+-----+
| ASIN(0.2)      |
+-----+
| 0.2013579207903308 |
+-----+

SELECT ASIN('foo');
+-----+
| ASIN('foo')    |
+-----+
|      0         |
+-----+

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message           |
+-----+-----+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'foo' |
+-----+-----+

```

## 1.1.12.5.10 ATAN

# Syntax

ATAN(X)

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

## Description

Returns the arc tangent of X, that is, the value whose tangent is X.

## Examples

```
SELECT ATAN(2);
+-----+
| ATAN(2)           |
+-----+
| 1.1071487177940904 |
+-----+

SELECT ATAN(-2);
+-----+
| ATAN(-2)           |
+-----+
| -1.1071487177940904 |
+-----+
```

## 1.1.12.5.11 ATAN2

# Syntax

ATAN(Y,X), ATAN2(Y,X)

## Description

Returns the arc tangent of the two variables X and Y. It is similar to calculating the arc tangent of Y / X, except that the signs of both arguments are used to determine the quadrant of the result.

## Examples

```
SELECT ATAN(-2,2);
+-----+
| ATAN(-2,2)           |
+-----+
| -0.7853981633974483 |
+-----+

SELECT ATAN2(PI(),0);
+-----+
| ATAN2(PI(),0)           |
+-----+
| 1.5707963267948966 |
+-----+
```

## 1.1.12.5.12 CEIL

# Syntax

CEIL(X)

## Description

CEIL() is a synonym for [CEILING\(\)](#).

### 1.1.12.5.13 CEILING

#### Syntax

```
CEILING(X)
```

#### Description

Returns the smallest integer value not less than X.

#### Examples

```
SELECT CEILING(1.23);
+-----+
| CEILING(1.23) |
+-----+
|          2   |
+-----+  
  
SELECT CEILING(-1.23);
+-----+
| CEILING(-1.23) |
+-----+
|         -1   |
+-----+
```

### 1.1.12.5.14 CONV

#### Syntax

```
CONV(N,from_base,to_base)
```

#### Description

Converts numbers between different number bases. Returns a string representation of the number

*N*

, converted from base

*from\_base*

to base

*to\_base*

.

Returns

NULL

if any argument is

NULL

, or if the second or third argument are not in the allowed range.

The argument

*N*

is interpreted as an integer, but may be specified as an integer or a string. The minimum base is 2 and the maximum base is 36. If

*to\_base*

is a negative number,

*N*

is regarded as a signed number. Otherwise,

*N*

is treated as unsigned.

[CONV\(\)](#)

works with 64-bit precision.

Some shortcuts for this function are also available:

```
BIN()  
  
,  
  
OCT()  
  
,  
  
HEX()  
  
,  
  
UNHEX()
```

. Also, MariaDB allows [binary](#) literal values and [hexadecimal](#) literal values.

## Examples

```
SELECT CONV('a',16,2);  
+-----+  
| CONV('a',16,2) |  
+-----+  
| 1010 |  
+-----+  
  
SELECT CONV('6E',18,8);  
+-----+  
| CONV('6E',18,8) |  
+-----+  
| 172 |  
+-----+  
  
SELECT CONV(-17,10,-18);  
+-----+  
| CONV(-17,10,-18) |  
+-----+  
| -H |  
+-----+  
  
SELECT CONV(12+'10'+'10'+0xa,10,10);  
+-----+  
| CONV(12+'10'+'10'+0xa,10,10) |  
+-----+  
| 42 |  
+-----+
```

## 1.1.12.5.15 COS

### Syntax

```
COS(X)
```

### Description

Returns the cosine of X, where X is given in radians.

## Examples

```
SELECT COS(PI());  
+-----+  
| COS(PI()) |  
+-----+  
| -1 |  
+-----+
```

## 1.1.12.5.16 COT

### Syntax

```
COT(X)
```

### Description

Returns the cotangent of X.

### Examples

```
SELECT COT(42);
+-----+
| COT(42)      |
+-----+
| 0.4364167060752729 |
+-----+

SELECT COT(12);
+-----+
| COT(12)      |
+-----+
| -1.5726734063976893 |
+-----+

SELECT COT(0);
ERROR 1690 (22003): DOUBLE value is out of range in 'cot(0)'
```

## 1.1.12.5.17 CRC32

### Syntax

<= MariaDB 10.7

```
CRC32(expr)
```

From MariaDB 10.8

```
CRC32([par,]expr)
```

### Description

Computes a cyclic redundancy check (CRC) value and returns a 32-bit unsigned value. The result is NULL if the argument is NULL. The argument is expected to be a string and (if possible) is treated as one if it is not.

Uses the ISO 3309 polynomial that used by zlib and many others. MariaDB 10.8 introduced the [CRC32C\(\)](#) function, which uses the alternate Castagnoli polynomia.

MariaDB starting with 10.8

Often, CRC is computed in pieces. To facilitate this, MariaDB 10.8.0 introduced an optional parameter:  
CRC32('MariaDB')=CRC32(CRC32('Maria'), 'DB').

### Examples

```
SELECT CRC32('MariaDB');
```

```
+-----+
| CRC32('MariaDB') |
+-----+
| 4227209140 |
+-----+
```

```
SELECT CRC32('mariadb');
```

```
+-----+
| CRC32('mariadb') |
+-----+
| 2594253378 |
+-----+
```

From MariaDB 10.8.0

```
SELECT CRC32(CRC32('Maria'), 'DB');
```

```
+-----+
| CRC32(CRC32('Maria'), 'DB') |
+-----+
| 4227209140 |
+-----+
```

## See Also

- [CRC32C\(\)](#)

### 1.1.12.5.18 CRC32C

MariaDB starting with 10.8

Introduced in MariaDB 10.8.0 to compute a cyclic redundancy check (CRC) value using the Castagnoli polynomial.

## Syntax

```
CRC32C([par,]expr)
```

## Description

MariaDB has always included a native unary function [CRC32\(\)](#) that computes the CRC-32 of a string using the ISO 3309 polynomial that used by zlib and many others.

InnoDB and MyRocks use a different polynomial, which was implemented in SSE4.2 instructions that were introduced in the Intel Nehalem microarchitecture. This is commonly called CRC-32C (Castagnoli).

The CRC32C function uses the Castagnoli polynomial.

This allows

```
SELECT...INTO DUMPFILE  
to be used for the creation of files with valid checksums, such as a logically empty InnoDB redo log file  
ib_logfile0  
corresponding to a particular log sequence number.
```

The optional parameter allows the checksum to be computed in pieces:  $\text{CRC32C('MariaDB')} = \text{CRC32C}(\text{CRC32C('Maria')}, 'DB')$ .

## Examples

```

SELECT CRC32C('MariaDB');
+-----+
| CRC32C('MariaDB') |
+-----+
|      809606978 |
+-----+

SELECT CRC32C(CRC32C('Maria'), 'DB');
+-----+
| CRC32C(CRC32C('Maria'), 'DB') |
+-----+
|          809606978 |
+-----+

```

## 1.1.12.5.19 DEGREES

### Syntax

`DEGREES(X)`

### Description

Returns the argument

$x$ , converted from radians to degrees.

This is the converse of the [RADIAN\(\)](#) function.

### Examples

```

SELECT DEGREES(PI());
+-----+
| DEGREES(PI()) |
+-----+
|      180 |
+-----+

SELECT DEGREES(PI() / 2);
+-----+
| DEGREES(PI() / 2) |
+-----+
|        90 |
+-----+

SELECT DEGREES(45);
+-----+
| DEGREES(45) |
+-----+
| 2578.3100780887 |
+-----+

```

## 1.1.12.5.20 EXP

### Syntax

`EXP(X)`

### Description

Returns the value of e (the base of natural logarithms) raised to the power of X. The inverse of this function is [LOG\(\)](#) (using a single argument only) or [LN\(\)](#).

If

$x$   
is

NULL  
, this function returns  
NULL

## Examples

```
SELECT EXP(2);
+-----+
| EXP(2)      |
+-----+
| 7.38905609893065 |
+-----+

SELECT EXP(-2);
+-----+
| EXP(-2)      |
+-----+
| 0.1353352832366127 |
+-----+

SELECT EXP(0);
+-----+
| EXP(0)      |
+-----+
| 1           |
+-----+

SELECT EXP(NULL);
+-----+
| EXP(NULL)   |
+-----+
| NULL        |
+-----+
```

## 1.1.12.5.21 FLOOR

### Syntax

```
FLOOR(X)
```

### Description

Returns the largest integer value not greater than X.

### Examples

```
SELECT FLOOR(1.23);
+-----+
| FLOOR(1.23) |
+-----+
| 1           |
+-----+

SELECT FLOOR(-1.23);
+-----+
| FLOOR(-1.23) |
+-----+
| -2          |
+-----+
```

## 1.1.12.5.22 GREATEST

### Syntax

```
GREATEST(value1,value2,...)
```

## Description

With two or more arguments, returns the largest (maximum-valued) argument. The arguments are compared using the same rules as for [LEAST\(\)](#).

## Examples

```
SELECT GREATEST(2,0);
+-----+
| GREATEST(2,0) |
+-----+
|      2      |
+-----+
```

```
SELECT GREATEST(34.0,3.0,5.0,767.0);
+-----+
| GREATEST(34.0,3.0,5.0,767.0) |
+-----+
|          767.0      |
+-----+
```

```
SELECT GREATEST('B','A','C');
+-----+
| GREATEST('B','A','C') |
+-----+
|  C                  |
+-----+
```

## 1.1.12.5.23 LEAST

### Syntax

```
LEAST(value1,value2,...)
```

## Description

With two or more arguments, returns the smallest (minimum-valued) argument. The arguments are compared using the following rules:

- If the return value is used in an INTEGER context or all arguments are integer-valued, they are compared as integers.
- If the return value is used in a REAL context or all arguments are real-valued, they are compared as reals.
- If any argument is a case-sensitive string, the arguments are compared as case-sensitive strings.
- In all other cases, the arguments are compared as case-insensitive strings.

LEAST() returns NULL if any argument is NULL.

## Examples

```
SELECT LEAST(2,0);
+-----+
| LEAST(2,0) |
+-----+
|      0      |
+-----+
```

```
SELECT LEAST(34.0,3.0,5.0,767.0);
+-----+
| LEAST(34.0,3.0,5.0,767.0) |
+-----+
|          3.0      |
+-----+
```

```
SELECT LEAST('B','A','C');
+-----+
| LEAST('B','A','C') |
+-----+
| A |
+-----+
```

## 1.1.12.5.24 LN

### Syntax

```
LN(X)
```

### Description

Returns the natural logarithm of X; that is, the base-e logarithm of X. If X is less than or equal to 0, or

NULL

, then NULL is returned.

The inverse of this function is

[EXP\(\)](#)

### Examples

```
SELECT LN(2);
+-----+
| LN(2) |
+-----+
| 0.693147180559945 |
+-----+
```

```
SELECT LN(-2);
+-----+
| LN(-2) |
+-----+
| NULL |
+-----+
```

## 1.1.12.5.25 LOG

### Syntax

```
LOG(X), LOG(B,X)
```

### Description

If called with one parameter, this function returns the natural logarithm of X. If X is less than or equal to 0, then NULL is returned.

If called with two parameters, it returns the logarithm of X to the base B. If B is <= 1 or X <= 0, the function returns NULL.

If any argument is

NULL

, the function returns

NULL

The inverse of this function (when called with a single argument) is the [EXP\(\)](#) function.

### Examples

LOG(X):

```
SELECT LOG(2);
+-----+
| LOG(2) |
+-----+
| 0.693147180559945 |
+-----+

SELECT LOG(-2);
+-----+
| LOG(-2) |
+-----+
| NULL |
+-----+
```

LOG(B,X)

```
SELECT LOG(2,16);
+-----+
| LOG(2,16) |
+-----+
| 4 |
+-----+

SELECT LOG(3,27);
+-----+
| LOG(3,27) |
+-----+
| 3 |
+-----+

SELECT LOG(3,1);
+-----+
| LOG(3,1) |
+-----+
| 0 |
+-----+

SELECT LOG(3,0);
+-----+
| LOG(3,0) |
+-----+
| NULL |
+-----+
```

## 1.1.12.5.26 LOG10

### Syntax

```
LOG10(X)
```

### Description

Returns the base-10 logarithm of X.

### Examples

```
SELECT LOG10(2);
+-----+
| LOG10(2) |
+-----+
| 0.301029995663981 |
+-----+

SELECT LOG10(100);
+-----+
| LOG10(100) |
+-----+
|      2 |
+-----+

SELECT LOG10(-100);
+-----+
| LOG10(-100) |
+-----+
|      NULL |
+-----+
```

## 1.1.12.5.27 LOG2

### Syntax

```
LOG2(X)
```

### Description

Returns the base-2 logarithm of X.

### Examples

```
SELECT LOG2(4398046511104);
+-----+
| LOG2(4398046511104) |
+-----+
|      42 |
+-----+

SELECT LOG2(65536);
+-----+
| LOG2(65536) |
+-----+
|      16 |
+-----+

SELECT LOG2(-100);
+-----+
| LOG2(-100) |
+-----+
|      NULL |
+-----+
```

## 1.1.12.5.28 MOD

### Syntax

```
MOD(N,M), N % M, N MOD M
```

### Description

Modulo operation. Returns the remainder of N divided by M. See also [Modulo Operator](#).

If the

ERROR\_ON\_DIVISION\_BY\_ZERO

**SQL\_MODE** is used, any number modulus zero produces an error. Otherwise, it returns NULL.

The integer part of a division can be obtained using **DIV**.

## Examples

```
SELECT 1042 % 50;
+-----+
| 1042 % 50 |
+-----+
|      42 |
+-----+

SELECT MOD(234, 10);
+-----+
| MOD(234, 10) |
+-----+
|        4 |
+-----+

SELECT 253 % 7;
+-----+
| 253 % 7 |
+-----+
|       1 |
+-----+

SELECT MOD(29,9);
+-----+
| MOD(29,9) |
+-----+
|       2 |
+-----+

SELECT 29 MOD 9;
+-----+
| 29 MOD 9 |
+-----+
|       2 |
+-----+
```

## 1.1.12.5.29 OCT

### Syntax

OCT(N)

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns a string representation of the octal value of N, where N is a longlong ( **BIGINT** ) number. This is equivalent to **CONV(N,10,8)** . Returns NULL if N is NULL.

## Examples

```
SELECT OCT(34);
```

```
+-----+
| OCT(34) |
+-----+
| 42      |
+-----+
```

```
SELECT OCT(12);
```

```
+-----+
| OCT(12) |
+-----+
| 14      |
+-----+
```

## See Also

- [CONV\(\)](#)
- [BIN\(\)](#)
- [HEX\(\)](#)

## 1.1.12.5.30 PI

### Syntax

```
PI()
```

### Description

Returns the value of  $\pi$  (pi). The default number of decimal places displayed is six, but MariaDB uses the full double-precision value internally.

### Examples

```
SELECT PI();
```

```
+-----+
| PI()      |
+-----+
| 3.141593 |
+-----+
```

```
SELECT PI()+0.00000000000000000000;
```

```
+-----+
| PI()+0.00000000000000000000 |
+-----+
| 3.1415926535897931159980 |
+-----+
```

## 1.1.12.5.31 POW

### Syntax

```
POW(X,Y)
```

### Description

Returns the value of X raised to the power of Y.

POWER() is a synonym.

### Examples

```
SELECT POW(2,3);
+-----+
| POW(2,3) |
+-----+
|      8   |
+-----+

SELECT POW(2,-2);
+-----+
| POW(2,-2) |
+-----+
|     0.25  |
+-----+
```

## 1.1.12.5.32 POWER

### Syntax

```
POWER(X,Y)
```

### Description

This is a synonym for [POW\(\)](#) , which returns the value of X raised to the power of Y.

## 1.1.12.5.33 RADIANS

### Syntax

```
RADIANS(X)
```

### Description

Returns the argument

X

, converted from degrees to radians. Note that  $\pi$  radians equals 180 degrees.

This is the converse of the [DEGREES\(\)](#) function.

### Examples

```

SELECT RADIANS(45);
+-----+
| RADIANS(45) |
+-----+
| 0.785398163397448 |
+-----+

SELECT RADIANS(90);
+-----+
| RADIANS(90) |
+-----+
| 1.5707963267949 |
+-----+

SELECT RADIANS(PI());
+-----+
| RADIANS(PI()) |
+-----+
| 0.0548311355616075 |
+-----+

SELECT RADIANS(180);
+-----+
| RADIANS(180) |
+-----+
| 3.14159265358979 |
+-----+

```

## 1.1.12.5.34 RAND

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Practical uses](#)
4. [Examples](#)
5. [See Also](#)

## Syntax

RAND(), RAND(N)

## Description

Returns a random

`DOUBLE`

precision floating point value v in the range  $0 \leq v < 1.0$ . If a constant integer argument N is specified, it is used as the seed value, which produces a repeatable sequence of column values. In the example below, note that the sequences of values produced by RAND(3) is the same both places where it occurs.

In a WHERE clause, RAND() is evaluated each time the WHERE is executed.

Statements using the RAND() function are not [safe for statement-based replication](#).

## Practical uses

The expression to get a random integer from a given range is the following:

```
FLOOR(min_value + RAND() * (max_value - min_value +1))
```

RAND() is often used to read random rows from a table, as follows:

```
SELECT * FROM my_table ORDER BY RAND() LIMIT 10;
```

Note, however, that this technique should never be used on a large table as it will be extremely slow. MariaDB will read all rows in the table, generate a random value for each of them, order them, and finally will apply the LIMIT clause.

# Examples

```
CREATE TABLE t (i INT);

INSERT INTO t VALUES(1),(2),(3);

SELECT i, RAND() FROM t;
+-----+
| i    | RAND()          |
+-----+
| 1   | 0.255651095188829 |
| 2   | 0.833920199269355 |
| 3   | 0.40264774151393 |
+-----+

SELECT i, RAND(3) FROM t;
+-----+
| i    | RAND(3)          |
+-----+
| 1   | 0.90576975597606 |
| 2   | 0.373079058130345 |
| 3   | 0.148086053457191 |
+-----+

SELECT i, RAND() FROM t;
+-----+
| i    | RAND()          |
+-----+
| 1   | 0.511478140495232 |
| 2   | 0.349447508668012 |
| 3   | 0.212803152588013 |
+-----+
```

Using the same seed, the same sequence will be returned:

```
SELECT i, RAND(3) FROM t;
+-----+
| i    | RAND(3)          |
+-----+
| 1   | 0.90576975597606 |
| 2   | 0.373079058130345 |
| 3   | 0.148086053457191 |
+-----+
```

Generating a random number from 5 to 15:

```
SELECT FLOOR(5 + (RAND() * 11));
```

## See Also

- [Techniques for Efficiently Finding a Random Row](#)
- [rand\\_seed1 and rand\\_seed2 system variables](#)

## 1.1.12.5.35 ROUND

### Syntax

```
ROUND(X), ROUND(X,D)
```

### Description

Rounds the argument

X  
to  
D  
decimal places. The rounding algorithm depends on the data type of  
X  
.

D  
defaults to  
0  
if not specified.  
D  
can be negative to cause  
D  
digits left of the decimal point of the value  
X  
to become zero.

## Examples

```
SELECT ROUND(-1.23);
+-----+
| ROUND(-1.23) |
+-----+
|          -1 |
+-----+  
  
SELECT ROUND(-1.58);
+-----+
| ROUND(-1.58) |
+-----+
|          -2 |
+-----+  
  
SELECT ROUND(1.58);
+-----+
| ROUND(1.58) |
+-----+
|          2 |
+-----+  
  
SELECT ROUND(1.298, 1);
+-----+
| ROUND(1.298, 1) |
+-----+
|          1.3 |
+-----+  
  
SELECT ROUND(1.298, 0);
+-----+
| ROUND(1.298, 0) |
+-----+
|          1 |
+-----+  
  
SELECT ROUND(23.298, -1);
+-----+
| ROUND(23.298, -1) |
+-----+
|          20 |
+-----+
```

## 1.1.12.5.36 SIGN

### Syntax

```
SIGN(X)
```

### Description

Returns the sign of the argument as -1, 0, or 1, depending on whether X is negative, zero, or positive.

## Examples

```
SELECT SIGN(-32);
+-----+
| SIGN(-32) |
+-----+
|      -1   |
+-----+

SELECT SIGN(0);
+-----+
| SIGN(0) |
+-----+
|      0   |
+-----+

SELECT SIGN(234);
+-----+
| SIGN(234) |
+-----+
|         1  |
+-----+
```

## See Also

- [ABS\(\)](#)

## 1.1.12.5.37 SIN

### Syntax

```
SIN(X)
```

### Description

Returns the sine of X, where X is given in radians.

### Examples

```
SELECT SIN(1.5707963267948966);
+-----+
| SIN(1.5707963267948966) |
+-----+
|          1   |
+-----+

SELECT SIN(PI());
+-----+
| SIN(PI()) |
+-----+
| 1.22460635382238e-16 |
+-----+

SELECT ROUND(SIN(PI()));
+-----+
| ROUND(SIN(PI())) |
+-----+
|        0   |
+-----+
```

## 1.1.12.5.38 SQRT

### Syntax

```
SQRT(X)
```

## Description

Returns the square root of X. If X is negative, NULL is returned.

## Examples

```
SELECT SQRT(4);
+-----+
| SQRT(4) |
+-----+
|      2   |
+-----+

SELECT SQRT(20);
+-----+
| SQRT(20)        |
+-----+
| 4.47213595499958 |
+-----+

SELECT SQRT(-16);
+-----+
| SQRT(-16) |
+-----+
|    NULL   |
+-----+

SELECT SQRT(1764);
+-----+
| SQRT(1764) |
+-----+
|      42   |
+-----+
```

## 1.1.12.5.39 TAN

### Syntax

```
TAN(X)
```

## Description

Returns the tangent of X, where X is given in radians.

## Examples

```
SELECT TAN(0.7853981633974483);
+-----+
| TAN(0.7853981633974483) |
+-----+
| 0.9999999999999999 |
+-----+  
  
SELECT TAN(PI());
+-----+
| TAN(PI()) |
+-----+
| -1.22460635382238e-16 |
+-----+  
  
SELECT TAN(PI()+1);
+-----+
| TAN(PI()+1) |
+-----+
| 1.5574077246549 |
+-----+  
  
SELECT TAN(RADIANS(PI()));
+-----+
| TAN(RADIANS(PI())) |
+-----+
| 0.0548861508080033 |
+-----+
```

## 1.1.12.5.40 TRUNCATE

This page documents the TRUNCATE function. See [TRUNCATE TABLE](#) for the DDL statement.

### Syntax

```
TRUNCATE(X,D)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns the number X, truncated to D decimal places. If D is 0, the result has no decimal point or fractional part. D can be negative to cause D digits left of the decimal point of the value X to become zero.

### Examples

```

SELECT TRUNCATE(1.223,1);
+-----+
| TRUNCATE(1.223,1) |
+-----+
|      1.2      |
+-----+

SELECT TRUNCATE(1.999,1);
+-----+
| TRUNCATE(1.999,1) |
+-----+
|      1.9      |
+-----+

SELECT TRUNCATE(1.999,0);
+-----+
| TRUNCATE(1.999,0) |
+-----+
|      1      |
+-----+

SELECT TRUNCATE(-1.999,1);
+-----+
| TRUNCATE(-1.999,1) |
+-----+
|     -1.9     |
+-----+

SELECT TRUNCATE(122,-2);
+-----+
| TRUNCATE(122,-2) |
+-----+
|      100      |
+-----+

SELECT TRUNCATE(10.28*100,0);
+-----+
| TRUNCATE(10.28*100,0) |
+-----+
|      1028      |
+-----+

```

## See Also

- [TRUNCATE TABLE](#)

### 1.1.12.6 Control Flow Functions

#### 1.1.12.6.1 CASE OPERATOR

## Syntax

```

CASE value WHEN [compare_value] THEN result [WHEN [compare_value] THEN
result ...] [ELSE result] END

CASE WHEN [condition] THEN result [WHEN [condition] THEN result ...]
[ELSE result] END

```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

## Description

The first version returns the result where value=compare\_value. The second version returns the result for the first condition that is true. If there was no matching result value, the result after ELSE is returned, or NULL if there is no ELSE part.

There is also a [CASE statement](#), which differs from the CASE operator described here.

## Examples

```
SELECT CASE 1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE 'more' END;
+-----+
| CASE 1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE 'more' END |
+-----+
| one |
+-----+

SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;
+-----+
| CASE WHEN 1>0 THEN 'true' ELSE 'false' END |
+-----+
| true |
+-----+

SELECT CASE BINARY 'B' WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;
+-----+
| CASE BINARY 'B' WHEN 'a' THEN 1 WHEN 'b' THEN 2 END |
+-----+
| NULL |
+-----+
```

### 1.1.12.6.2 DECODE

#### Syntax

```
DECODE(crypt_str,pass_str)
```

In [Oracle mode](#) from [MariaDB 10.3.2](#) :

```
DECODE(expr, search_expr, result_expr [, search_expr2, result_expr2 ...] [default_expr])
```

In all modes from [MariaDB 10.3.2](#) :

```
DECODE_ORACLE(expr, search_expr, result_expr [, search_expr2, result_expr2 ...] [default_expr])
```

#### Description

In the default mode,

DECODE

decrypts the encrypted string *crypt\_str* using *pass\_str* as the password. *crypt\_str* should be a string returned from [ENCODE\(\)](#). The resulting string will be the original string only if *pass\_str* is the same.

In [Oracle mode](#) from [MariaDB 10.3.2](#),

DECODE

compares *expr* to the search expressions, in order. If it finds a match, the corresponding result expression is returned. If no matches are found, the default expression is returned, or NULL if no default is provided.

NULLs are treated as equivalent.

DECODE\_ORACLE

is a synonym for the Oracle-mode version of the function, and is available in all modes.

## Examples

From [MariaDB 10.3.2](#) :

```

SELECT DECODE_ORACLE(2+1,3*1,'found1',3*2,'found2','default');
+
| DECODE_ORACLE(2+1,3*1,'found1',3*2,'found2','default') |
+
| found1 |
+
+-----+
| DECODE_ORACLE(2+4,3*1,'found1',3*2,'found2','default');
+
| DECODE_ORACLE(2+4,3*1,'found1',3*2,'found2','default') |
+
| found2 |
+
+-----+
| DECODE_ORACLE(2+2,3*1,'found1',3*2,'found2','default');
+
| DECODE_ORACLE(2+2,3*1,'found1',3*2,'found2','default') |
+
| default |
+
+-----+

```

Nulls are treated as equivalent:

```

SELECT DECODE_ORACLE(NULL,NULL,'Nulls are equivalent','Nulls are not equivalent');
+
| DECODE_ORACLE(NULL,NULL,'Nulls are equivalent','Nulls are not equivalent') |
+
| Nulls are equivalent |
+
+-----+

```

## 1.1.12.6.3 DECODE\_ORACLE

MariaDB starting with 10.3.2

`DECODE_ORACLE`  
is a synonym for the [Oracle mode](#) version of the [DECODE function](#), and is available in all modes.

## 1.1.12.6.4 IF Function

### Syntax

```
IF(expr1,expr2,expr3)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

If

```

expr1
is
TRUE
(
expr1 <> 0
and
expr1 <> NULL
) then
IF()
returns
expr2
; otherwise it returns
expr3
.
```

`IF()`

returns a numeric or string value, depending on the context in which it is used.

**Note:** There is also an [IF statement](#) which differs from the

`IF()`

function described here.

## Examples

```
SELECT IF(1>2,2,3);
+-----+
| IF(1>2,2,3) |
+-----+
|            3 |
+-----+
```

```
SELECT IF(1<2,'yes','no');
+-----+
| IF(1<2,'yes','no') |
+-----+
| yes                |
+-----+
```

```
SELECT IF(STRCMP('test','test1'),'no','yes');
+-----+
| IF(STRCMP('test','test1'),'no','yes') |
+-----+
| no                                |
+-----+
```

## See Also

There is also an [IF statement](#), which differs from the  
`IF()`  
function described above.

### 1.1.12.6.5 IFNULL

#### Syntax

```
IFNULL(expr1,expr2)
NVL(expr1,expr2)
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

If

*expr1*  
is not NULL, IFNULL() returns  
*expr1*  
; otherwise it returns  
*expr2*  
. IFNULL() returns a numeric or string value, depending on the context in which it is used.

From [MariaDB 10.3](#), NVL() is an alias for IFNULL().

## Examples

```

SELECT IFNULL(1,0);
+-----+
| IFNULL(1,0) |
+-----+
|      1      |
+-----+

SELECT IFNULL(NULL,10);
+-----+
| IFNULL(NULL,10) |
+-----+
|          10     |
+-----+

SELECT IFNULL(1/0,10);
+-----+
| IFNULL(1/0,10) |
+-----+
|      10.0000   |
+-----+

SELECT IFNULL(1/0,'yes');
+-----+
| IFNULL(1/0,'yes') |
+-----+
| yes               |
+-----+

```

## See Also

- [NULL values](#)
- [IS NULL operator](#)
- [IS NOT NULL operator](#)
- [COALESCE function](#)
- [NULLIF function](#)
- [CONNECT data types](#)

## 1.1.12.6.6 NULLIF

### Syntax

`NULLIF(expr1,expr2)`

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Returns NULL if expr1 = expr2 is true, otherwise returns expr1. This is the same as `CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END`.

### Examples

```

SELECT NULLIF(1,1);
+-----+
| NULLIF(1,1) |
+-----+
|      NULL    |
+-----+

SELECT NULLIF(1,2);
+-----+
| NULLIF(1,2) |
+-----+
|          1    |
+-----+

```

## See Also

- [NULL values](#)
- [IS NULL operator](#)
- [IS NOT NULL operator](#)
- [COALESCE function](#)
- [IFNULL function](#)
- [CONNECT data types](#)

## 1.1.12.6.7 NVL

MariaDB starting with [10.3](#)

From [MariaDB 10.3](#), NVL is a synonym for [IFNULL](#).

## 1.1.12.6.8 NVL2

MariaDB starting with [10.3](#)

The NVL2 function was introduced in [MariaDB 10.3.0](#).

## Syntax

```
NVL2(expr1,expr2,expr3)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

NVL2

function returns a value based on whether a specified expression is NULL or not. If *expr1* is not NULL, then NVL2 returns *expr2*. If *expr1* is NULL, then NVL2 returns *expr3*.

## Examples

```
SELECT NVL2(NULL,1,2);
+-----+
| NVL2(NULL,1,2) |
+-----+
|          2     |
+-----+  
  
SELECT NVL2('x',1,2);
+-----+
| NVL2('x',1,2) |
+-----+
|          1     |
+-----+
```

## See Also

- [IFNULL \(or NVL\)](#)

## 1.1.12.7 Pseudo Columns

### 1.1.12.7.1 \_rowid

## Syntax

\_rowid

## Description

The

\_rowid  
pseudo column is mapped to the primary key in the related table. This can be used as a replacement of the  
rowid  
pseudo column in other databases. Another usage is to simplify sql queries as one doesn't have to know the name of the primary key.

## Examples

```
create table t1 (a int primary key, b varchar(80));  
insert into t1 values (1,"one"),(2,"two");  
select * from t1 where _rowid=1;
```

a	b
1	one

```
update t1 set b="three" where _rowid=2;  
select * from t1 where _rowid>=1 and _rowid<=10;
```

a	b
1	one
2	three

### 1.1.12.8 Secondary Functions

#### 1.1.12.8.1 Bit Functions and Operators

##### 1.1.12.8.1.1 Operator Precedence

The precedence is the order in which the SQL operators are evaluated.

The following list shows the SQL operator precedence. Operators that appear first in the list have a higher precedence. Operators which are listed together have the same precedence.

•

INTERVAL

•

BINARY

COLLATE

•

!

•

(unary minus), [[bitwise-not]] (unary bit inversion)

•  
    ||  
    (string concatenation)

•  
    ^

•  
    \*

,

/

,

DIV

,

%

,

MOD

•  
    -

,

+

•  
    -

<<

,

>>

•  
    &

•  
    |

•  
    =

(comparison),

<=>

,

>=

,

>

,

<=

,

<

,

<>

,

!=

,

IS

,

LIKE

,

REGEXP

,

IN

•

BETWEEN

,

CASE

,

WHEN

,

THEN

,

ELSE

,

END

•

NOT

•

&&

,

AND

•

XOR

•

||

(logical or),

OR

•

=

(assignment),

:=

Functions precedence is always higher than operators precedence.

In this page

CASE

refers to the [CASE operator](#), not to the [CASE statement](#)

•

If the

HIGH\_NOT\_PRECEDENCE

SQL\_MODE is set,

NOT

has the same precedence as

!

•

The

||

operator's precedence, as well as its meaning, depends on the

PIPES\_AS\_CONCAT

SQL\_MODE flag: if it is on,

||

can be used to concatenate strings (like the [CONCAT\(\)](#) function) and has a higher precedence.

The

=

operator's precedence depends on the context - it is higher when

=

is used as a comparison operator.

[Parenthesis](#) can be used to modify the operators precedence in an expression.

## Short-circuit evaluation

The

AND

,

OR

,

&&

and

||

operators support short-circuit evaluation. This means that, in some cases, the expression on the right of those operators is not evaluated, because its result cannot affect the result. In the following cases, short-circuit evaluation is used and

x()

is not evaluated:

•

FALSE AND x()

•

FALSE && x()

•

TRUE OR x()

•

TRUE || x()

•  
NULL BETWEEN x() AND x()

Note however that the short-circuit evaluation does *not* apply to

```
NULL AND x()  
. Also,  
BETWEEN  
's right operands are not evaluated if the left operand is  
NULL  
, but in all other cases all the operands are evaluated.
```

This is a speed optimization. Also, since functions can have side-effects, this behavior can be used to choose whether execute them or not using a concise syntax:

```
SELECT some_function() OR log_error();
```

## 1.1.12.8.1.2 &

### Syntax

```
&
```

### Description

Bitwise AND. Converts the values to binary and compares bits. Only if both the corresponding bits are 1 is the resulting bit also 1.

See also [bitwise OR](#).

### Examples

```
SELECT 2&1;  
+----+  
| 2&1 |  
+----+  
|   0 |  
+----+  
  
SELECT 3&1;  
+----+  
| 3&1 |  
+----+  
|   1 |  
+----+  
  
SELECT 29 & 15;  
+-----+  
| 29 & 15 |  
+-----+  
|      13 |  
+-----+
```

## 1.1.12.8.1.3 <<

### Syntax

```
value1 << value2
```

### Description

Converts a longlong ( [BIGINT](#) ) number ( *value1* ) to binary and shifts *value2* units to the left.

### Examples

```
SELECT 1 << 2;
+-----+
| 1 << 2 |
+-----+
|      4 |
+-----+
```

## 1.1.12.8.1.4 >>

### Syntax

```
value1 >> value2
```

### Description

Converts a longlong ( [BIGINT](#) ) number ( *value1* ) to binary and shifts *value2* units to the right.

### Examples

```
SELECT 4 >> 2;
+-----+
| 4 >> 2 |
+-----+
|      1 |
+-----+
```

## 1.1.12.8.1.5 BIT\_COUNT

### Syntax

```
BIT_COUNT(N)
```

### Description

Returns the number of bits that are set in the argument N.

### Examples

```
SELECT BIT_COUNT(29), BIT_COUNT(b'101010');
+-----+-----+
| BIT_COUNT(29) | BIT_COUNT(b'101010') |
+-----+-----+
|        4 |          3 |
+-----+-----+
```

## 1.1.12.8.1.6 ^

### Syntax

```
^
```

### Description

Bitwise XOR. Converts the values to binary and compares bits. If one (and only one) of the corresponding bits is 1 is the resulting bit also 1.

### Examples

```
SELECT 1 ^ 1;
+-----+
| 1 ^ 1 |
+-----+
|     0 |
+-----+

SELECT 1 ^ 0;
+-----+
| 1 ^ 0 |
+-----+
|      1 |
+-----+

SELECT 11 ^ 3;
+-----+
| 11 ^ 3 |
+-----+
|      8 |
+-----+
```

## 1.1.12.8.1.7 |

### Syntax

|

### Description

Bitwise OR. Converts the values to binary and compares bits. If either of the corresponding bits has a value of 1, the resulting bit is also 1.

See also [bitwise AND](#).

### Examples

```
SELECT 2|1;
+-----+
| 2|1 |
+-----+
|     3 |
+-----+

SELECT 29 | 15;
+-----+
| 29 | 15 |
+-----+
|      31 |
+-----+
```

## 1.1.12.8.1.8 ~

### Syntax

~

### Description

Bitwise NOT. Converts the value to 4 bytes binary and inverts all bits.

### Examples

```
SELECT 3 & ~1;
+-----+
| 3 & ~1 |
+-----+
|      2 |
+-----+

SELECT 5 & ~1;
+-----+
| 5 & ~1 |
+-----+
|      4 |
+-----+
```

## 1.1.12.8.1.9 Parentheses

Parentheses are sometimes called precedence operators - this means that they can be used to change the other [operator's precedence](#) in an expression. The expressions that are written between parentheses are computed before the expressions that are written outside. Parentheses must always contain an expression (that is, they cannot be empty), and can be nested.

For example, the following expressions could return different results:

- NOT a OR b
- NOT (a OR b)

In the first case,

NOT  
applies to  
a  
, so if  
a  
is  
FALSE  
or  
b  
is  
TRUE  
, the expression returns  
TRUE  
. In the second case,  
NOT  
applies to the result of  
a OR b  
, so if at least one of  
a  
or  
b  
is  
TRUE  
, the expression is  
TRUE  
.

When the precedence of operators is not intuitive, you can use parentheses to make it immediately clear for whoever reads the statement.

The precedence of the

NOT  
operator can also be affected by the  
HIGH\_NOT\_PRECEDENCE  
[SQL\\_MODE](#) flag.

## Other uses

Parentheses must always be used to enclose [subqueries](#).

Parentheses can also be used in a

[JOIN](#)

statement between multiple tables to determine which tables must be joined first.

Also, parentheses are used to enclose the list of parameters to be passed to built-in functions, user-defined functions and stored routines. However, when no parameter is passed to a stored procedure, parentheses are optional. For builtin functions and user-defined functions, spaces are not allowed between the function name and the open parenthesis, unless the

IGNORE\_SPACE

SQL\_MODE is set. For stored routines (and for functions if

IGNORE\_SPACE

is set) spaces are allowed before the open parenthesis, including tab characters and new line characters.

## Syntax errors

If there are more open parentheses than closed parentheses, the error usually looks like this:

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use near '' a
t line 1
```

Note the empty string.

If there are more closed parentheses than open parentheses, the error usually looks like this:

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use near ')'
at line 1
```

Note the quoted closed parenthesis.

## 1.1.12.8.1.10 TRUE FALSE

### Description

The constants TRUE and FALSE evaluate to 1 and 0, respectively. The constant names can be written in any lettercase.

### Examples

```
SELECT TRUE, true, FALSE, false;
+-----+-----+-----+
| TRUE | TRUE | FALSE | FALSE |
+-----+-----+-----+
|   1  |    1 |     0 |     0 |
+-----+-----+-----+
```

## 1.1.12.8.2 Encryption, Hashing and Compression Functions

### 1.1.12.8.2.1 AES\_DECRYPT

#### Syntax

```
AES_DECRYPT(crypt_str,key_str)
```

#### Description

This function allows decryption of data using the official AES (Advanced Encryption Standard) algorithm. For more information, see the description of

[AES\\_ENCRYPT\(\)](#)

### 1.1.12.8.2.2 AES\_ENCRYPT

#### Syntax

```
AES_ENCRYPT(str,key_str)
```

## Description

AES\_ENCRYPT()  
and

[AES\\_DECRYPT\(\)](#)

allow encryption and decryption of data using the official AES (Advanced Encryption Standard) algorithm, previously known as "Rijndael." Encoding with a 128-bit key length is used, but you can extend it up to 256 bits by modifying the source. We chose 128 bits because it is much faster and it is secure enough for most purposes.

AES\_ENCRYPT()  
encrypts a string  
*str*  
using the key  
*key\_str*  
, and returns a binary string.

AES\_DECRYPT()  
decrypts the encrypted string and returns the original string.

The input arguments may be any length. If either argument is NULL, the result of this function is also  
NULL

Because AES is a block-level algorithm, padding is used to encode uneven length strings and so the result string length may be calculated using this formula:

```
16 x (trunc(string_length / 16) + 1)
```

If

AES\_DECRYPT()  
detects invalid data or incorrect padding, it returns  
NULL  
. However, it is possible for  
AES\_DECRYPT()  
to return a non-  
NULL  
value (possibly garbage) if the input data or the key is invalid.

## Examples

```
INSERT INTO t VALUES (AES_ENCRYPT('text',SHA2('password',512)));
```

## 1.1.12.8.2.3 COMPRESS

### Syntax

```
COMPRESS(string_to_compress)
```

## Description

Compresses a string and returns the result as a binary string. This function requires MariaDB to have been compiled with a compression library such as zlib. Otherwise, the return value is always

NULL  
. The compressed string can be uncompressed with

[UNCOMPRESS\(\)](#)

The `have_compress` server system variable indicates whether a compression library is present.

## Examples

```
SELECT LENGTH(COMPRESS(REPEAT('a',1000)));
+-----+
| LENGTH(COMPRESS(REPEAT('a',1000))) |
+-----+
|                               21 |
+-----+


SELECT LENGTH(COMPRESS(''));
+-----+
| LENGTH(COMPRESS('')) |
+-----+
|                      0 |
+-----+


SELECT LENGTH(COMPRESS('a'));
+-----+
| LENGTH(COMPRESS('a')) |
+-----+
|                         13 |
+-----+


SELECT LENGTH(COMPRESS(REPEAT('a',16)));
+-----+
| LENGTH(COMPRESS(REPEAT('a',16))) |
+-----+
|                               15 |
+-----+
```

### 1.1.12.8.2.4 DECODE

### 1.1.12.8.2.5 DES\_DECRYPT

#### Syntax

```
DES_DECRYPT(crypt_str[,key_str])
```

#### Description

Decrypts a string encrypted with

```
DES_ENCRYPT()
```

. If an error occurs, this function returns  
NULL

This function works only if MariaDB has been configured with [TLS support](#).

If no

```
key_str  
argument is given,  
DES_DECRYPT()
```

examines the first byte of the encrypted string to determine the DES key number that was used to encrypt the original string, and then reads the key from the DES key file to decrypt the message. For this to work, the user must have the SUPER privilege. The key file can be specified with the

```
--des-key-file  
server option.
```

If you pass this function a

```
key_str  
argument, that string is used as the key for decrypting the message.
```

If the

```
crypt_str  
argument does not appear to be an encrypted string, MariaDB returns the given crypt_str.
```

## 1.1.12.8.2.6 DES\_ENCRYPT

### Syntax

```
DES_ENCRYPT(str[,{key_num|key_str}])
```

### Description

Encrypts the string with the given key using the Triple-DES algorithm.

This function works only if MariaDB has been configured with [TLS support](#).

The encryption key to use is chosen based on the second argument to

```
DES_ENCRYPT()  
, if one was given. With no argument, the first key from the DES key file is used. With a  
key_num  
argument, the given key number (0-9) from the DES key file is used. With a  
key_str  
argument, the given key string is used to encrypt  
str
```

The key file can be specified with the

```
--des-key-file  
server option.
```

The return string is a binary string where the first character is

```
CHAR(128 | key_num)  
. If an error occurs,  
DES_ENCRYPT()  
returns  
NULL
```

The 128 is added to make it easier to recognize an encrypted key. If you use a string key,

```
key_num  
is 127.
```

The string length for the result is given by this formula:

```
new_len = orig_len + (8 - (orig_len % 8)) + 1
```

Each line in the DES key file has the following format:

```
key_num des_key_str
```

Each

```
key_num  
value must be a number in the range from 0 to 9. Lines in the file may be in any order.  
des_key_str  
is the string that is used to encrypt the message. There should be at least one space between the number and the key. The first key is the  
default key that is used if you do not specify any key argument to  
DES_ENCRYPT()
```

You can tell MariaDB to read new key values from the key file with the FLUSH DES\_KEY\_FILE statement. This requires the RELOAD privilege.

One benefit of having a set of default keys is that it gives applications a way to check for the existence of encrypted column values, without giving the end user the right to decrypt those values.

### Examples

```
SELECT customer_address FROM customer_table  
WHERE crypted_credit_card = DES_ENCRYPT('credit_card_number');
```

### See Also

- [DES\\_DECRYPT\(\)](#)

## 1.1.12.8.2.7 ENCODE

### Syntax

```
ENCODE(str,pass_str)
```

### Description

ENCODE is not considered cryptographically secure, and should not be used for password encryption.

#### Encrypt

str  
using  
pass\_str  
as the password. To decrypt the result, use

[DECODE\(\)](#)

The result is a binary string of the same length as

str

The strength of the encryption is based on how good the random generator is.

It is not recommended to rely on the encryption performed by the ENCODE function. Using a salt value (changed when a password is updated) will improve matters somewhat, but for storing passwords, consider a more cryptographically secure function, such as [SHA2\(\)](#).

### Examples

```
ENCODE('not so secret text', CONCAT('random_salt','password'))
```

## 1.1.12.8.2.8 ENCRYPT

### Syntax

```
ENCRYPT(str[,salt])
```

### Description

Encrypts a string using the Unix crypt() system call, returning an encrypted binary string. The

salt

argument should be a string with at least two characters or the returned result will be NULL. If no salt argument is given, a random value of sufficient length is used.

It is not recommended to use ENCRYPT() with utf16, utf32 or ucs2 multi-byte character sets because the crypt() system call expects a string terminated with a zero byte.

Note that the underlying crypt() system call may have some limitations, such as ignoring all but the first eight characters.

If the `have_crypt` system variable is set to

NO

(because the crypt() system call is not available), the ENCRYPT function will always return NULL.

### Examples

```
SELECT ENCRYPT('encrypt me');  
+-----+  
| ENCRYPT('encrypt me') |  
+-----+  
| 4I5BsEx0lqTDk |  
+-----+
```

## 1.1.12.8.2.9 MD5

### Syntax

```
MD5(str)
```

### Description

Calculates an MD5 128-bit checksum for the string.

The return value is a 32-hex digit string, and as of MariaDB 5.5 , is a nonbinary string in the connection [character set and collation](#) , determined by the values of the [character\\_set\\_connection](#) and [collation\\_connection](#) system variables. Before 5.5, the return value was a binary string.

NULL is returned if the argument was NULL.

### Examples

```
SELECT MD5('testing');
+-----+
| MD5('testing') |
+-----+
| ae2b1fc... |
+-----+
```

## 1.1.12.8.2.10 PASSWORD

### Syntax

```
PASSWORD(str)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

The PASSWORD() function is used for hashing passwords for use in authentication by the MariaDB server. It is not intended for use in other applications.

Calculates and returns a hashed password string from the plaintext password *str* . Returns an empty string (>= MariaDB 10.0.4 ) if the argument was NULL.

The return value is a nonbinary string in the connection [character set and collation](#) , determined by the values of the [character\\_set\\_connection](#) and [collation\\_connection](#) system variables.

This is the function that is used for hashing MariaDB passwords for storage in the Password column of the [user table](#) (see [privileges](#) ), usually used with the [SET PASSWORD](#) statement. It is not intended for use in other applications.

Until MariaDB 10.3 , the return value is 41-bytes in length, and the first character is always \*\*. From MariaDB 10.4 , the function takes into account the authentication plugin where applicable (A [CREATE USER](#) or [SET PASSWORD](#) statement). For example, when used in conjunction with a user authenticated by the [ed25519 plugin](#) , the statement will create a longer hash:

```

CREATE USER edtest@localhost IDENTIFIED VIA ed25519 USING PASSWORD('secret');

CREATE USER edtest2@localhost IDENTIFIED BY 'secret';

SELECT CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)) FROM mysql.global_priv
 WHERE user LIKE 'edtest%\G
***** 1. row *****
CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)): edtest@localhost => {
...
  "plugin": "ed25519",
  "authentication_string": "ZIgUREUg5PVgQ6LskhXm0+eZLS0nC8be6HPjYWR4YJY",
...
}
***** 2. row *****
CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)): edtest2@localhost => {
...
  "plugin": "mysql_native_password",
  "authentication_string": "*14E65567ABDB5135D0CFD9A70B3032C179A49EE7",
...
}

```

The behavior of this function is affected by the value of the [old\\_passwords](#) system variable. If this is set to

```

1
(
0

```

is default), MariaDB reverts to using the [mysql\\_old\\_password authentication plugin](#) by default for newly created users and passwords.

## Examples

```

SELECT PASSWORD('notagoodpwd');
+-----+
| PASSWORD('notagoodpwd') |
+-----+
| *3A70EE9FC6594F88CE9E959CD51C5A1C002DC937 |
+-----+

```

```

SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass');

```

## See Also

- [Password Validation Plugins](#) - permits the setting of basic criteria for passwords
- [OLD\\_PASSWORD\(\)](#) - pre-MySQL 4.1 password function

## 1.1.12.8.2.11 PASSWORD

### Syntax

```
PASSWORD(str)
```

### Contents

- [Syntax](#)
- [Description](#)
- [Examples](#)
- [See Also](#)

### Description

The `PASSWORD()` function is used for hashing passwords for use in authentication by the MariaDB server. It is not intended for use in other applications.

Calculates and returns a hashed password string from the plaintext password `str`. Returns an empty string ( $\geq$  [MariaDB 10.0.4](#)) if the argument was `NULL`.

The return value is a nonbinary string in the connection [character set and collation](#), determined by the values of the [character\\_set\\_connection](#) and

`collation_connection` system variables.

This is the function that is used for hashing MariaDB passwords for storage in the `Password` column of the `user` table (see [privileges](#)), usually used with the `SET PASSWORD` statement. It is not intended for use in other applications.

Until [MariaDB 10.3](#), the return value is 41-bytes in length, and the first character is always `**`. From [MariaDB 10.4](#), the function takes into account the authentication plugin where applicable (A `CREATE USER` or `SET PASSWORD` statement). For example, when used in conjunction with a user authenticated by the [ed25519 plugin](#), the statement will create a longer hash:

```
CREATE USER edtest@localhost IDENTIFIED VIA ed25519 USING PASSWORD('secret');

CREATE USER edtest2@localhost IDENTIFIED BY 'secret';

SELECT CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)) FROM mysql.global_priv
  WHERE user LIKE 'edtest%'\G
***** 1. row *****
CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)): edtest@localhost => {
...
  "plugin": "ed25519",
  "authentication_string": "ZIGUREUg5PVgQ6LskhXm0+eZLS0nC8be6HPjYWR4YJY",
...
}
***** 2. row *****
CONCAT(user, '@', host, ' => ', JSON_DETAILED(priv)): edtest2@localhost => {
...
  "plugin": "mysql_native_password",
  "authentication_string": "*14E65567ABDB5135D0CFD9A70B3032C179A49EE7",
...
}
```

The behavior of this function is affected by the value of the `old_passwords` system variable. If this is set to

```
1
(
0
```

is default), MariaDB reverts to using the [mysql\\_old\\_password authentication plugin](#) by default for newly created users and passwords.

## Examples

```
SELECT PASSWORD('notagoodpwd');
+-----+
| PASSWORD('notagoodpwd') |
+-----+
| *3A70EE9FC6594F88CE9E959CD51C5A1C002DC937 |
+-----+
```

```
SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass');
```

## See Also

- [Password Validation Plugins](#) - permits the setting of basic criteria for passwords
- [OLD\\_PASSWORD\(\)](#) - pre-MySQL 4.1 password function

## 1.1.12.8.2.12 SHA1

### Syntax

```
SHA1(str), SHA(str)
```

### Description

Calculates an SHA-1 160-bit checksum for the string

`str`

, as described in RFC 3174 (Secure Hash Algorithm).

The value is returned as a string of 40 hex digits, or NULL if the argument was NULL. As of [MariaDB 5.5](#), the return value is a nonbinary string in the connection `character set and collation`, determined by the values of the `character_set_connection` and `collation_connection` system variables. Before 5.5, the return value was a binary string.

## Examples

```
SELECT SHA1('some boring text');
+-----+
| SHA1('some boring text')           |
+-----+
| af969fc2085b1bb6d31e517d5c456def5cdd7093 |
+-----+
```

## 1.1.12.8.2.13 SHA2

### Syntax

```
SHA2(str,hash_len)
```

### Description

Given a string

*str*

, calculates an SHA-2 checksum, which is considered more cryptographically secure than its [SHA-1](#) equivalent. The SHA-2 family includes SHA-224, SHA-256, SHA-384, and SHA-512, and the

*hash\_len*

must correspond to one of these, i.e. 224, 256, 384 or 512. 0 is equivalent to 256.

The return value is a nonbinary string in the connection [character set and collation](#) , determined by the values of the [character\\_set\\_connection](#) and [collation\\_connection](#) system variables.

NULL is returned if the hash length is not valid, or the string

*str*

is NULL.

SHA2 will only work if MariaDB was has been configured with [TLS support](#) .

## Examples

```
SELECT SHA2('Maria',224);
+-----+
| SHA2('Maria',224)           |
+-----+
| 6cc67add32286412efcab9d0e1675a43a5c2ef3cec8879f81516ff83 |
+-----+

SELECT SHA2('Maria',256);
+-----+
| SHA2('Maria',256)           |
+-----+
| 9ff18ebe7449349f358e3af0b57cf7a032c1c6b2272cb2656ff85eb112232f16 |
+-----+

SELECT SHA2('Maria',0);
+-----+
| SHA2('Maria',0)           |
+-----+
| 9ff18ebe7449349f358e3af0b57cf7a032c1c6b2272cb2656ff85eb112232f16 |
+-----+
```

## 1.1.12.8.2.14 UNCOMPRESS

## 1.1.12.8.2.15 UNCOMPRESSED\_LENGTH

### 1.1.12.8.3 Information Functions

#### 1.1.12.8.3.1 BENCHMARK

## Syntax

```
BENCHMARK(count,expr)
```

## Description

The BENCHMARK() function executes the expression

expr  
repeatedly  
count

times. It may be used to time how quickly MariaDB processes the expression. The result value is always 0. The intended use is from within the [mysql client](#), which reports query execution times.

## Examples

```
SELECT BENCHMARK(1000000,ENCODE('hello','goodbye'));
+-----+
| BENCHMARK(1000000,ENCODE('hello','goodbye')) |
+-----+
|          0 |
+-----+
1 row in set (0.21 sec)
```

## 1.1.12.8.3.2 BINLOG\_GTIID\_POS

### Syntax

```
BINLOG_GTIID_POS(binlog_filename,binlog_offset)
```

### Description

The BINLOG\_GTIID\_POS() function takes as input an old-style [binary log](#) position in the form of a file name and a file offset. It looks up the position in the current binlog, and returns a string representation of the corresponding [GTID](#) position. If the position is not found in the current binlog, NULL is returned.

### Examples

```
SELECT BINLOG_GTIID_POS("master-bin.000001", 600);
```

### See Also

- [SHOW BINLOG EVENTS](#) - Show events and their positions in the binary log

## 1.1.12.8.3.3 CHARSET

### Syntax

```
CHARSET(str)
```

### Description

Returns the [character set](#) of the string argument. If

str  
is not a string, it is considered as a binary string (so the function returns 'binary'). This applies to  
NULL  
, too. The return value is a string in the utf8 [character set](#).

## Examples

```

SELECT CHARSET('abc');
+-----+
| CHARSET('abc') |
+-----+
| latin1          |
+-----+

SELECT CHARSET(CONVERT('abc' USING utf8));
+-----+
| CHARSET(CONVERT('abc' USING utf8)) |
+-----+
| utf8            |
+-----+

SELECT CHARSET(USER());
+-----+
| CHARSET(USER()) |
+-----+
| utf8           |
+-----+

```

## 1.1.12.8.3.4 COERCIBILITY

### Syntax

```
COERCIBILITY(str)
```

### Description

Returns the collation coercibility value of the string argument. Coercibility defines what will be converted to what in case of collation conflict, with an expression with higher coercibility being converted to the collation of an expression with lower coercibility.

Coercibility	Description	Example
0	Explicit	Value using a COLLATE clause
1	No collation	Concatenated strings using different collations
2	Implicit	Column value
3	Constant	USER() return value
4	Coercible	Literal string
5	Ignorable	NULL or derived from NULL

### Examples

```

SELECT COERCIBILITY('abc' COLLATE latin1_swedish_ci);
+-----+
| COERCIBILITY('abc' COLLATE latin1_swedish_ci) |
+-----+
|          0 |
+-----+

SELECT COERCIBILITY(USER());
+-----+
| COERCIBILITY(USER()) |
+-----+
|          3 |
+-----+

SELECT COERCIBILITY('abc');
+-----+
| COERCIBILITY('abc') |
+-----+
|          4 |
+-----+

```

## 1.1.12.8.3.5 COLLATION

# Syntax

COLLATION(str)

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Returns the collation of the string argument. If

str  
is not a string, it is considered as a binary string (so the function returns 'binary'). This applies to  
NULL  
, too. The return value is a string in the utf8 [character set](#).

See [Character Sets and Collations](#).

## Examples

```
SELECT COLLATION('abc');
+-----+
| COLLATION('abc') |
+-----+
| latin1_swedish_ci |
+-----+

SELECT COLLATION(_utf8'abc');
+-----+
| COLLATION(_utf8'abc') |
+-----+
| utf8_general_ci      |
+-----+
```

## See Also

- [String literals](#)
- [CAST\(\)](#)
- [CONVERT\(\)](#)

## 1.1.12.8.3.6 CONNECTION\_ID

### Syntax

CONNECTION\_ID()

## Description

Returns the connection ID (thread ID) for the connection. Every thread (including events) has an ID that is unique among the set of currently connected clients.

Until [MariaDB 10.3.1](#), returns

MYSQL\_TYPE\_LONGLONG  
, or [bigint\(10\)](#), in all cases. From [MariaDB 10.3.1](#), returns  
MYSQL\_TYPE\_LONG  
, or [int\(10\)](#), when the result would fit within 32-bits.

## Examples

```
SELECT CONNECTION_ID();
+-----+
| CONNECTION_ID() |
+-----+
|      3 |
+-----+
```

## See Also

- [SHOW PROCESSLIST](#)
- [INFORMATION\\_SCHEMA.PROCESSLIST](#)

## 1.1.12.8.3.7 CURRENT\_ROLE

### Syntax

```
CURRENT_ROLE, CURRENT_ROLE()
```

### Description

Returns the current role name. This determines your access privileges. The return value is a string in the utf8 character set .

If there is no current role, NULL is returned.

The output of

```
SELECT CURRENT_ROLE
```

is equivalent to the contents of the [ENABLED\\_ROLES](#) Information Schema table.

[USER\(\)](#) returns the combination of user and host used to login. [CURRENT\\_USER\(\)](#) returns the account used to determine current connection's privileges.

### Examples

```
SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |
+-----+
| NULL         |
+-----+

SET ROLE staff;

SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |
+-----+
| staff        |
+-----+
```

## 1.1.12.8.3.8 CURRENT\_USER

### Syntax

```
CURRENT_USER, CURRENT_USER()
```

### Description

Returns the user name and host name combination for the MariaDB account that the server used to authenticate the current client. This account determines your access privileges. The return value is a string in the utf8 character set .

The value of

```
CURRENT_USER()
```

can differ from the value of [USER\(\)](#) . [CURRENT\\_ROLE\(\)](#) returns the current active role.

## Examples

```
shell> mysql --user="anonymous"

select user(),current_user();
+-----+-----+
| user() | current_user() |
+-----+-----+
| anonymous@localhost | @localhost      |
+-----+-----+
```

When calling

CURRENT\_USER()  
in a stored procedure, it returns the owner of the stored procedure, as defined with  
DEFINER

## See Also

- [USER\(\)](#)
- [CREATE PROCEDURE](#)

## 1.1.12.8.3.9 DATABASE

### Syntax

```
DATABASE()
```

### Description

Returns the default (current) database name as a string in the utf8 [character set](#). If there is no default database, DATABASE() returns NULL. Within a [stored routine](#), the default database is the database that the routine is associated with, which is not necessarily the same as the database that is the default in the calling context.

SCHEMA() is a synonym for DATABASE().

To select a default database, the [USE](#) statement can be run. Another way to set the default database is specifying its name at [mysql](#) command line client startup.

## Examples

```
SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| NULL       |
+-----+

USE test;
Database changed

SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| test        |
+-----+
```

## 1.1.12.8.3.10 DECODE\_HISTOGRAM

### Syntax

```
DECODE_HISTOGRAM(hist_type,histogram)
```

## Description

Returns a string of comma separated numeric values corresponding to a probability distribution represented by the histogram of type

```
hist_type
(
SINGLE_PREC_HB
or
DOUBLE_PREC_HB
). The
hist_type
and
histogram
would be commonly used from the mysql.column\_stats table.
```

See [Histogram Based Statistics](#) for details.

## Examples

```
CREATE TABLE origin (
  i INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  v INT UNSIGNED NOT NULL
);

INSERT INTO origin(v) VALUES
(1),(2),(3),(4),(5),(10),(20),
(30),(40),(50),(60),(70),(80),
(90),(100),(200),(400),(800);

SET histogram_size=10,histogram_type=SINGLE_PREC_HB;

ANALYZE TABLE origin PERSISTENT FOR ALL;
+-----+-----+-----+
| Table | Op      | Msg_type | Msg_text
+-----+-----+-----+
| test.origin | analyze | status    | Engine-independent statistics collected |
| test.origin | analyze | status    | OK
+-----+-----+-----+

SELECT db_name,table_name,column_name,hist_type,
hex(histogram),decode_histogram(hist_type,histogram)
FROM mysql.column_stats WHERE db_name='test' and table_name='origin';
+-----+-----+-----+-----+
| db_name | table_name | column_name | hist_type       | hex(histogram)      | decode_histogram(hist_type,histogram)
+-----+-----+-----+-----+
| test    | origin     | i           | SINGLE_PREC_HB | 0F2D3C5A7887A5C3D2F0 | 0.059,0.118,0.059,0.118,0.059,0.118,0.059,0.118,0
| test    | origin     | v           | SINGLE_PREC_HB | 000001060C0F161C1F7F | 0.000,0.000,0.004,0.020,0.024,0.012,0.027,0.024,0
+-----+-----+-----+-----+

SET histogram_size=20,histogram_type=DOUBLE_PREC_HB;

ANALYZE TABLE origin PERSISTENT FOR ALL;
+-----+-----+-----+
| Table | Op      | Msg_type | Msg_text
+-----+-----+-----+
| test.origin | analyze | status    | Engine-independent statistics collected |
| test.origin | analyze | status    | OK
+-----+-----+-----+

SELECT db_name,table_name,column_name,
hist_type,hex(histogram),decode_histogram(hist_type,histogram)
FROM mysql.column_stats WHERE db_name='test' and table_name='origin';
+-----+-----+-----+-----+
| db_name | table_name | column_name | hist_type       | hex(histogram)      | decode_histogram(hist_type,histogram)
+-----+-----+-----+-----+
| test    | origin     | i           | DOUBLE_PREC_HB | 0F0F2D2D3C3C5A5A78788787A5A5C3C3D2D2F0F0 | 0.05882,0.11765,0.05882,0.117
| test    | origin     | v           | DOUBLE_PREC_HB | 5200F600480116067E0CB30F1B16831CB81FD67F | 0.00125,0.00250,0.00125,0.018
+-----+-----+-----+-----+
```

## 1.1.12.8.3.11 DEFAULT

### Syntax

```
DEFAULT(col_name)
```

## Description

Returns the default value for a table column. If the column has no default value (and is not NULLABLE - NULLABLE fields have a NULL default), an error is returned.

For integer columns using [AUTO\\_INCREMENT](#),

0

is returned.

When using

DEFAULT

as a value to set in an [INSERT](#) or [UPDATE](#) statement, you can use the bare keyword

DEFAULT

without the parentheses and argument to refer to the column in context. You can only use

DEFAULT

as a bare keyword if you are using it alone without a surrounding expression or function.

## Examples

Select only non-default values for a column:

```
SELECT i FROM t WHERE i != DEFAULT(i);
```

Update values to be one greater than the default value:

```
UPDATE t SET i = DEFAULT(i)+1 WHERE i < 100;
```

When referring to the default value exactly in

UPDATE

or

INSERT

, you can omit the argument:

```
INSERT INTO t (i) VALUES (DEFAULT);
UPDATE t SET i = DEFAULT WHERE i < 100;
```

```
CREATE OR REPLACE TABLE t (
    i INT NOT NULL AUTO_INCREMENT,
    j INT NOT NULL,
    k INT DEFAULT 3,
    l INT NOT NULL DEFAULT 4,
    m INT,
    PRIMARY KEY (i)
);
```

```
DESC t;
```

Field	Type	Null	Key	Default	Extra
i	int(11)	NO	PRI	NULL	auto_increment
j	int(11)	NO		NULL	
k	int(11)	YES		3	
l	int(11)	NO		4	
m	int(11)	YES		NULL	

```
INSERT INTO t (j) VALUES (1);
INSERT INTO t (j,m) VALUES (2,2);
INSERT INTO t (j,l,m) VALUES (3,3,3);
```

```
SELECT * FROM t;
```

i	j	k	l	m
1	1	3	4	NULL
2	2	3	4	2
3	3	3	3	3

```
SELECT DEFAULT(i), DEFAULT(k), DEFAULT(l), DEFAULT(m) FROM t;
+-----+-----+-----+
| DEFAULT(i) | DEFAULT(k) | DEFAULT(l) | DEFAULT(m) |
+-----+-----+-----+
|      0 |      3 |      4 |    NULL |
|      0 |      3 |      4 |    NULL |
|      0 |      3 |      4 |    NULL |
+-----+-----+-----+


SELECT DEFAULT(i), DEFAULT(k), DEFAULT(l), DEFAULT(m), DEFAULT(j) FROM t;
ERROR 1364 (HY000): Field 'j' doesn't have a default value

SELECT * FROM t WHERE i = DEFAULT(i);
Empty set (0.001 sec)

SELECT * FROM t WHERE j = DEFAULT(j);
ERROR 1364 (HY000): Field 'j' doesn't have a default value

SELECT * FROM t WHERE k = DEFAULT(k);
+-----+-----+-----+
| i | j | k | l | m |
+-----+-----+-----+
| 1 | 1 | 3 | 4 | NULL |
| 2 | 2 | 3 | 4 |   2 |
| 3 | 3 | 3 | 3 |   3 |
+-----+-----+-----+


SELECT * FROM t WHERE l = DEFAULT(l);
+-----+-----+-----+
| i | j | k | l | m |
+-----+-----+-----+
| 1 | 1 | 3 | 4 | NULL |
| 2 | 2 | 3 | 4 |   2 |
+-----+-----+-----+


SELECT * FROM t WHERE m = DEFAULT(m);
Empty set (0.001 sec)

SELECT * FROM t WHERE m <=> DEFAULT(m);
+-----+-----+-----+
| i | j | k | l | m |
+-----+-----+-----+
| 1 | 1 | 3 | 4 | NULL |
+-----+-----+-----+
```

#### See Also

- CREATE TABLE DEFAULT Clause

## 1.1.12.8.3.12 FOUND\_ROWS

## Syntax

`FOUND_ROWS()`

## Description

A `SELECT` statement may include a `LIMIT` clause to restrict the number of rows the server returns to the client. In some cases, it is desirable to know how many rows the statement would have returned without the `LIMIT`, but without running the statement again. To obtain this row count, include a `SQL_CALC_FOUND_ROWS` option in the `SELECT` statement, and then invoke `FOUND_ROWS()` afterwards.

You can also use `FOUND_ROWS()` to obtain the number of rows returned by a `SELECT` which does not contain a `LIMIT` clause. In this case you don't need to use the `SQL_CALC_FOUND_ROWS` option. This can be useful for example in a `stored procedure`.

Also, this function works with some other statements which return a resultset, including `SHOW` , `DESC` and `HELP` . For `DELETE ... RETURNING` you should use `ROW_COUNT()` . It also works as a [prepared statement](#) , or after executing a prepared statement.

Statements which don't return any results don't affect FOUND\_ROWS() - the previous value will still be returned.

**Warning:** When used after a `CALL` statement, this function returns the number of rows selected by the last query in the procedure, not by the whole procedure.

Statements using the FOUND\_ROWS() function are not safe for replication.

# Examples

```
SHOW ENGINES;
+-----+-----+-----+-----+
| Engine | Support | Comment | Transactions | XA | Savepoint |
+-----+-----+-----+-----+
| InnoDB | DEFAULT | Supports transactions, row-level locking, and foreign keys | YES | YES | YES |
...
| SPHINX | YES | Sphinx storage engine | NO | NO | NO |
+-----+-----+-----+-----+
11 rows in set (0.01 sec)

SELECT FOUND_ROWS();
+-----+
| FOUND_ROWS() |
+-----+
|      11 |
+-----+

SELECT SQL_CALC_FOUND_ROWS * FROM tbl_name WHERE id > 100 LIMIT 10;

SELECT FOUND_ROWS();
+-----+
| FOUND_ROWS() |
+-----+
|      23 |
+-----+
```

## See Also

- [ROW\\_COUNT\(\)](#)

## 1.1.12.8.3.13 LAST\_INSERT\_ID

### Syntax

```
LAST_INSERT_ID(), LAST_INSERT_ID(expr)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

`LAST_INSERT_ID()` (no arguments) returns the first automatically generated value successfully inserted for an `AUTO_INCREMENT` column as a result of the most recently executed `INSERT` statement. The value of `LAST_INSERT_ID()` remains unchanged if no rows are successfully inserted.

If one gives an argument to `LAST_INSERT_ID()`, then it will return the value of the expression and the next call to `LAST_INSERT_ID()` will return the same value. The value will also be sent to the client and can be accessed by the `mysql_insert_id` function.

For example, after inserting a row that generates an `AUTO_INCREMENT` value, you can get the value like this:

```
SELECT LAST_INSERT_ID();
+-----+
| LAST_INSERT_ID() |
+-----+
|      9 |
+-----+
```

You can also use `LAST_INSERT_ID()` to delete the last inserted row:

```
DELETE FROM product WHERE id = LAST_INSERT_ID();
```

If no rows were successfully inserted, `LAST_INSERT_ID()` returns 0.

The value of `LAST_INSERT_ID()` will be consistent across all versions if all rows in the `INSERT` or `UPDATE` statement were successful.

The currently executing statement does not affect the value of LAST\_INSERT\_ID(). Suppose that you generate an AUTO\_INCREMENT value with one statement, and then refer to LAST\_INSERT\_ID() in a multiple-row INSERT statement that inserts rows into a table with its own AUTO\_INCREMENT column. The value of LAST\_INSERT\_ID() will remain stable in the second statement; its value for the second and later rows is not affected by the earlier row insertions. (However, if you mix references to LAST\_INSERT\_ID() and LAST\_INSERT\_ID(expr), the effect is undefined.)

If the previous statement returned an error, the value of LAST\_INSERT\_ID() is undefined. For transactional tables, if the statement is rolled back due to an error, the value of LAST\_INSERT\_ID() is left undefined. For manual [ROLLBACK](#), the value of LAST\_INSERT\_ID() is not restored to that before the transaction; it remains as it was at the point of the ROLLBACK.

Within the body of a stored routine (procedure or function) or a trigger, the value of LAST\_INSERT\_ID() changes the same way as for statements executed outside the body of these kinds of objects. The effect of a stored routine or trigger upon the value of LAST\_INSERT\_ID() that is seen by following statements depends on the kind of routine:

- If a [stored procedure](#) executes statements that change the value of LAST\_INSERT\_ID(), the new value will be seen by statements that follow the procedure call.
- For [stored functions](#) and [triggers](#) that change the value, the value is restored when the function or trigger ends, so following statements will not see a changed value.

## Examples

```
CREATE TABLE t (
  id INTEGER UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  f VARCHAR(1))
ENGINE = InnoDB;
```

```
INSERT INTO t(f) VALUES('a');
```

```
SELECT LAST_INSERT_ID();
+-----+
| LAST_INSERT_ID() |
+-----+
|          1 |
+-----+
```

```
INSERT INTO t(f) VALUES('b');
```

```
INSERT INTO t(f) VALUES('c');
```

```
SELECT LAST_INSERT_ID();
+-----+
| LAST_INSERT_ID() |
+-----+
|          3 |
+-----+
```

```
INSERT INTO t(f) VALUES('d'), ('e');
```

```
SELECT LAST_INSERT_ID();
+-----+
| LAST_INSERT_ID() |
+-----+
|          4 |
+-----+
```

```
SELECT * FROM t;
+-----+
| id | f    |
+-----+
| 1  | a    |
| 2  | b    |
| 3  | c    |
| 4  | d    |
| 5  | e    |
+-----+
```

```
SELECT LAST_INSERT_ID(12);
+-----+
| LAST_INSERT_ID(12) |
+-----+
|          12 |
+-----+
```

```
SELECT LAST_INSERT_ID();
+-----+
| LAST_INSERT_ID() |
+-----+
```

```

|           12 |
+-----+
INSERT INTO t(f) VALUES('f');

SELECT LAST_INSERT_ID();
+-----+
| LAST_INSERT_ID() |
+-----+
|          6 |
+-----+

SELECT * FROM t;
+-----+
| id | f    |
+-----+
| 1  | a    |
| 2  | b    |
| 3  | c    |
| 4  | d    |
| 5  | e    |
| 6  | f    |
+-----+

SELECT LAST_INSERT_ID(12);
+-----+
| LAST_INSERT_ID(12) |
+-----+
|          12 |
+-----+

INSERT INTO t(f) VALUES('g');

SELECT * FROM t;
+-----+
| id | f    |
+-----+
| 1  | a    |
| 2  | b    |
| 3  | c    |
| 4  | d    |
| 5  | e    |
| 6  | f    |
| 7  | g    |
+-----+

```

## See Also

- [mysql\\_insert\\_id](#)
- [AUTO\\_INCREMENT](#)
- [AUTO\\_INCREMENT handling in InnoDB](#)
- [Sequences](#) - an alternative to auto\_increment available from [MariaDB 10.3](#)

## 1.1.12.8.3.14 LAST\_VALUE

### Syntax

```
LAST_VALUE(expr,[expr,...])
```

```

LAST_VALUE(expr) OVER (
  [ PARTITION BY partition_expression ]
  [ ORDER BY order_list ]
)

```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

# Description

`LAST_VALUE()`  
evaluates all expressions and returns the last.

This is useful together with [setting user variables to a value with @var:=expr](#), for example when you want to get data of rows updated/deleted without having to do two queries against the table.

Since [MariaDB 10.2.2](#), `LAST_VALUE` can be used as a [window function](#).

Returns NULL if no last value exists.

## Examples

```
CREATE TABLE t1 (a int, b int);
INSERT INTO t1 VALUES(1,10),(2,20);
DELETE FROM t1 WHERE a=1 AND last_value(@a:=a,@b:=b,1);
SELECT @a,@b;
+-----+-----+
| @a   | @b   |
+-----+-----+
|     1 |    10 |
+-----+-----+
```

As a [window function](#):

```
CREATE TABLE t1 (
  pk int primary key,
  a int,
  b int,
  c char(10),
  d decimal(10, 3),
  e real
);

INSERT INTO t1 VALUES
( 1, 0, 1, 'one', 0.1, 0.001),
( 2, 0, 2, 'two', 0.2, 0.002),
( 3, 0, 3, 'three', 0.3, 0.003),
( 4, 1, 2, 'three', 0.4, 0.004),
( 5, 1, 1, 'two', 0.5, 0.005),
( 6, 1, 1, 'one', 0.6, 0.006),
( 7, 2, NULL, 'n_one', 0.5, 0.007),
( 8, 2, 1, 'n_two', NULL, 0.008),
( 9, 2, 2, NULL, 0.7, 0.009),
(10, 2, 0, 'n_four', 0.8, 0.010),
(11, 2, 10, NULL, 0.9, NULL);

SELECT pk, FIRST_VALUE(pk) OVER (ORDER BY pk) AS first_asc,
       LAST_VALUE(pk) OVER (ORDER BY pk) AS last_asc,
       FIRST_VALUE(pk) OVER (ORDER BY pk DESC) AS first_desc,
       LAST_VALUE(pk) OVER (ORDER BY pk DESC) AS last_desc
FROM t1
ORDER BY pk DESC;

+-----+-----+-----+-----+
| pk | first_asc | last_asc | first_desc | last_desc |
+-----+-----+-----+-----+
| 11 |      1 |     11 |      11 |      11 |
| 10 |      1 |     10 |      11 |      10 |
|  9 |      1 |      9 |      11 |       9 |
|  8 |      1 |      8 |      11 |       8 |
|  7 |      1 |      7 |      11 |       7 |
|  6 |      1 |      6 |      11 |       6 |
|  5 |      1 |      5 |      11 |       5 |
|  4 |      1 |      4 |      11 |       4 |
|  3 |      1 |      3 |      11 |       3 |
|  2 |      1 |      2 |      11 |       2 |
|  1 |      1 |      1 |      11 |       1 |
+-----+-----+-----+-----+
```

```

CREATE OR REPLACE TABLE t1 (i int);
INSERT INTO t1 VALUES (1),(2),(3),(4),(5),(6),(7),(8),(9),(10);

SELECT i,
    FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN CURRENT ROW and 1 FOLLOWING) AS f_1f,
    LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN CURRENT ROW and 1 FOLLOWING) AS l_1f,
    FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS f_1p1f,
    LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS f_1p1f,
    FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 2 PRECEDING AND 1 PRECEDING) AS f_2p1p,
    LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 2 PRECEDING AND 1 PRECEDING) AS f_2p1p,
    FIRST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING) AS f_1f2f,
    LAST_VALUE(i) OVER (ORDER BY i ROWS BETWEEN 1 FOLLOWING AND 2 FOLLOWING) AS f_1f2f
FROM t1;

```

i	f_1f	l_1f	f_1p1f	f_1p1f	f_2p1p	f_2p1p	f_1f2f	f_1f2f
1	1	2	1	2	NULL	NULL	2	3
2	2	3	1	3	1	1	3	4
3	3	4	2	4	1	2	4	5
4	4	5	3	5	2	3	5	6
5	5	6	4	6	3	4	6	7
6	6	7	5	7	4	5	7	8
7	7	8	6	8	5	6	8	9
8	8	9	7	9	6	7	9	10
9	9	10	8	10	7	8	10	10
10	10	10	9	10	8	9	NULL	NULL

## See Also

- [Setting a variable to a value](#)

## 1.1.12.8.3.15 PROCEDURE ANALYSE

### Syntax

```
analyse([max_elements[,max_memory]])
```

### Description

This procedure is defined in the sql/sql\_analyse.cc file. It examines the result from a query and returns an analysis of the results that suggests optimal data types for each column. To obtain this analysis, append PROCEDURE ANALYSE to the end of a `SELECT` statement:

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max_elements,[max_memory]])
```

For example:

```
SELECT col1, col2 FROM table1 PROCEDURE ANALYSE(10, 2000);
```

The results show some statistics for the values returned by the query, and propose an optimal data type for the columns. This can be helpful for checking your existing tables, or after importing new data. You may need to try different settings for the arguments so that PROCEDURE ANALYSE() does not suggest the ENUM data type when it is not appropriate.

The arguments are optional and are used as follows:

- `max_elements` (default 256) is the maximum number of distinct values that analyse notices per column. This is used by analyse to check whether the optimal data type should be of type ENUM; if there are more than `max_elements` distinct values, then ENUM is not a suggested type.
- `max_memory` (default 8192) is the maximum amount of memory that analyse should allocate per column while trying to find all distinct values.

## See Also

- [PROCEDURE](#)
- [SELECT](#)

## 1.1.12.8.3.16 ROWNUM

From MariaDB 10.6.1 , the  
ROWNUM()  
function is supported.

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [Optimizations](#)
5. [Other Changes Related to ROWNUM](#)
6. [Other Considerations](#)
7. [See Also](#)

## Syntax

ROWNUM()

In Oracle mode one can just use

ROWNUM  
, without the parentheses.

## Description

ROWNUM()  
returns the current number of accepted rows in the current context. Its main purpose is to emulate the  
ROWNUM  
[pseudo column in Oracle](#). For MariaDB native applications, we recommend the usage of [LIMIT](#) , as it is easier to use and gives more predictable results than the usage of  
ROWNUM()

The main difference between using

LIMIT  
and  
ROWNUM()  
to limit the rows in the result is that  
LIMIT  
works on the result set while  
ROWNUM  
works on the number of accepted rows (before any  
ORDER  
or  
GROUP BY  
clauses).

The following queries will return the same results:

```
SELECT * from t1 LIMIT 10;  
SELECT * from t1 WHERE ROWNUM() <= 10;
```

While the following may return different results based on in which orders the rows are found:

```
SELECT * from t1 ORDER BY a LIMIT 10;  
SELECT * from t1 ORDER BY a WHERE ROWNUM() <= 10;
```

The recommended way to use

ROWNUM  
to limit the number of returned rows and get predictable results is to have the query in a subquery and test for  
ROWNUM()  
in the outer query:

```
SELECT * FROM (select * from t1 ORDER BY a) WHERE ROWNUM() <= 10;
```

ROWNUM()  
can be used in the following contexts:

- `SELECT`
- `INSERT`
- `UPDATE`
- `DELETE`
- `LOAD DATA INFILE`

Used in other contexts,

`ROWNUM()`  
will return 0.

## Examples

```
INSERT INTO t1 VALUES (1,ROWNUM()),(2,ROWNUM()),(3,ROWNUM());

INSERT INTO t1 VALUES (1),(2) returning a, ROWNUM();

UPDATE t1 SET row_num_column=ROWNUM();

DELETE FROM t1 WHERE a < 10 AND ROWNUM() < 2;

LOAD DATA INFILE 'filename' into table t1 fields terminated by ','  

lines terminated by "\r\n" (a,b) set c=ROWNUM();
```

## Optimizations

In many cases where

`ROWNUM()`  
is used, MariaDB will use the same optimizations it uses with `LIMIT`.

`LIMIT`  
optimization is possible when using  
`ROWNUM`  
in the following manner:

- When one is in a top level
    - WHERE clause comparing `ROWNUM()` with a numerical constant using any of the following expressions:
      - `ROWNUM() < number`
      - `ROWNUM() <= number`
      - `ROWNUM() = 1`
- `ROWNUM()` can be also be the right argument to the comparison function.

In the above cases,

`LIMIT`  
optimization can be done in the following cases:

- For the current sub query when the `ROWNUM` comparison is done on the top level:

```
SELECT * from t1 WHERE ROWNUM() <= 2 AND t1.a > 0
```

- For an inner sub query, when the upper level has only a

`ROWNUM()`  
comparison in the  
WHERE  
clause:

```
SELECT * from (select * from t1) as t WHERE ROWNUM() <= 2
```

## Other Changes Related to ROWNUM

When

ROWNUM()  
is used anywhere in a query, the optimization to ignore  
ORDER BY  
in subqueries are disabled.

This was done to get the following common Oracle query to work as expected:

```
select * from (select * from t1 order by a desc) as t where rownum() <= 2;
```

By default MariaDB ignores any  
ORDER BY  
in subqueries both because the SQL standard defines results sets in subqueries to be un-ordered and because of performance reasons (especially when using views in subqueries). See [MDEV-3926](#) "Wrong result with GROUP BY ... WITH ROLLUP" for a discussion of this topic.

## Other Considerations

While MariaDB tries to emulate Oracle's usage of

ROWNUM()  
as closely as possible, there are cases where the result is different:

- When the optimizer finds rows in a different order (because of different storage methods or optimization). This may also happen in Oracle if one adds or deletes an index, in which case the rows may be found in a different order.

Note that usage of

ROWNUM()  
in functions or [stored procedures](#) will use their own context, not the caller's context.

## See Also

- [MDEV-24089](#) support oracle syntax: rownum
- [LIMIT clause](#)

## 1.1.12.8.3.17 ROW\_COUNT

### Syntax

```
ROW_COUNT()
```

### Description

ROW\_COUNT() returns the number of rows updated, inserted or deleted by the preceding statement. This is the same as the row count that the mysql client displays and the value from the [mysql\\_affected\\_rows\(\)](#) C API function.

Generally:

- For statements which return a result set (such as [SELECT](#) , [SHOW](#) , [DESC](#) or [HELP](#) ), returns -1, even when the result set is empty. This is also true for administrative statements, such as [OPTIMIZE](#) .
- For DML statements other than [SELECT](#) and for [ALTER TABLE](#) , returns the number of affected rows.
- For DDL statements (including [TRUNCATE](#) ) and for other statements which don't return any result set (such as [USE](#) , [DO](#) , [SIGNAL](#) or [DEALLOCATE PREPARE](#) ), returns 0.

For [UPDATE](#) , affected rows is by default the number of rows that were actually changed. If the [CLIENT\\_FOUND\\_ROWS](#) flag to [mysql\\_real\\_connect\(\)](#) is specified when connecting to mysqld, affected rows is instead the number of rows matched by the WHERE clause.

For [REPLACE](#) , deleted rows are also counted. So, if REPLACE deletes a row and adds a new row, ROW\_COUNT() returns 2.

For [INSERT ... ON DUPLICATE KEY](#) , updated rows are counted twice. So, if INSERT adds a new rows and modifies another row, ROW\_COUNT() returns 3.

ROW\_COUNT() does not take into account rows that are not directly deleted/updated by the last statement. This means that rows deleted by foreign keys or triggers are not counted.

**Warning:** You can use ROW\_COUNT() with prepared statements, but you need to call it after EXECUTE, not after [DEALLOCATE PREPARE](#) , because the row count for allocate prepare is always 0.

**Warning:** When used after a [CALL](#) statement, this function returns the number of rows affected by the last statement in the procedure, not by the whole procedure.

**Warning:** After [INSERT DELAYED](#) , ROW\_COUNT() returns the number of the rows you tried to insert, not the number of the successful writes.

This information can also be found in the [diagnostics area](#) .

Statements using the ROW\_COUNT() function are not [safe for replication](#) .

## Examples

```
CREATE TABLE t (A INT);

INSERT INTO t VALUES(1),(2),(3);

SELECT ROW_COUNT();
+-----+
| ROW_COUNT() |
+-----+
|      3      |
+-----+

DELETE FROM t WHERE A IN(1,2);

SELECT ROW_COUNT();
+-----+
| ROW_COUNT() |
+-----+
|      2      |
+-----+
```

Example with prepared statements:

```
SET @q = 'INSERT INTO t VALUES(1),(2),(3);';

PREPARE stmt FROM @q;

EXECUTE stmt;
Query OK, 3 rows affected (0.39 sec)
Records: 3  Duplicates: 0  Warnings: 0

SELECT ROW_COUNT();
+-----+
| ROW_COUNT() |
+-----+
|      3      |
+-----+
```

## See Also

- [FOUND\\_ROWS\(\)](#)

## 1.1.12.8.3.18 SCHEMA

### Syntax

```
SCHEMA()
```

### Description

This function is a synonym for [DATABASE\(\)](#).

## 1.1.12.8.3.19 SESSION\_USER

### Syntax

```
SESSION_USER()
```

### Description

`SESSION_USER()` is a synonym for [USER\(\)](#).

## 1.1.12.8.3.20 SYSTEM\_USER

## Syntax

```
SYSTEM_USER()
```

## Description

SYSTEM\_USER() is a synonym for [USER\(\)](#).

## 1.1.12.8.3.21 USER

## Syntax

```
USER()
```

## Description

Returns the current MariaDB user name and host name, given when authenticating to MariaDB, as a string in the utf8 [character set](#).

Note that the value of USER() may differ from the value of [CURRENT\\_USER\(\)](#), which is the user used to authenticate the current client.

[CURRENT\\_ROLE\(\)](#)

returns the current active role.

SYSTEM\_USER()  
and  
SESSION\_USER  
are synonyms for  
USER()

Statements using the

USER()  
function or one of its synonyms are not [safe for statement level replication](#).

## Examples

```
shell> mysql --user="anonymous"

SELECT USER(),CURRENT_USER();
+-----+-----+
| USER() | CURRENT_USER() |
+-----+-----+
| anonymous@localhost | @localhost |
+-----+-----+
```

To select only the IP address, use [SUBSTRING\\_INDEX\(\)](#),

```
SELECT SUBSTRING_INDEX(USER(), '@', -1);
+-----+
| SUBSTRING_INDEX(USER(), '@', -1) |
+-----+
| 192.168.0.101 |
+-----+
```

## See Also

- [CURRENT\\_USER\(\)](#)

## 1.1.12.8.3.22 VERSION

## Syntax

## Description

Returns a string that indicates the MariaDB server version. The string uses the utf8 character set .

## Examples

```
SELECT VERSION();
+-----+
| VERSION()      |
+-----+
| 10.4.7-MariaDB |
+-----+
```

The

VERSION()  
string may have one or more of the following suffixes:

Suffix	Description
-embedded	The server is an embedded server (libmysqld).
-log	General logging, slow logging or binary (replication) logging is enabled.
-debug	The server is compiled for debugging.
-valgrind	The server is compiled to be instrumented with valgrind.

## Changing the Version String

Some old legacy code may break because they are parsing the

VERSION  
string and expecting a MySQL string or a simple version string like Joomla til API17, see MDEV-7780 .

From MariaDB 10.2 , one can fool these applications by setting the version string from the command line or the my.cnf files with --version=.... .

### 1.1.12.8.4 Miscellaneous Functions

#### 1.1.12.8.4.1 GET\_LOCK

## Syntax

```
GET_LOCK(str,timeout)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Tries to obtain a lock with a name given by the string

```
str
, using a timeout of
timeout
seconds. Returns
1
if the lock was obtained successfully,
0
if the attempt timed out (for example, because another client has previously locked the name), or
NULL
if an error occurred (such as running out of memory or the thread was killed with mysqladmin kill).
```

A lock is released with [RELEASE\\_LOCK\(\)](#) , when the connection terminates (either normally or abnormally). A connection can hold multiple locks at the

same time, so a lock that is no longer needed needs to be explicitly released.

The `IS_FREE_LOCK` function returns whether a specified lock a free or not, and the `IS_USED_LOCK` whether the function is in use or not.

Locks obtained with

```
GET_LOCK()  
do not interact with transactions. That is, committing a transaction does not release any such locks obtained during the transaction.
```

It is also possible to recursively set the same lock. If a lock with the same name is set

```
n  
times, it needs to be released  
n  
times as well.
```

```
str  
is case insensitive for  
GET_LOCK()  
and related functions. If  
str  
is an empty string or  
NULL  
,  
GET_LOCK()  
returns  
NULL  
and does nothing. From MariaDB 10.2.2,  
timeout  
supports microseconds. Before then, it was rounded to the closest integer.
```

If the `metadata_lock_info` plugin is installed, locks acquired with this function are visible in the [Information Schema METADATA\\_LOCK\\_INFO](#) table.

This function can be used to implement application locks or to simulate record locks. Names are locked on a server-wide basis. If a name has been locked by one client,

```
GET_LOCK()  
blocks any request by another client for a lock with the same name. This allows clients that agree on a given lock name to use the name to perform cooperative advisory locking. But be aware that it also allows a client that is not among the set of cooperating clients to lock a name, either inadvertently or deliberately, and thus prevent any of the cooperating clients from locking that name. One way to reduce the likelihood of this is to use lock names that are database-specific or application-specific. For example, use lock names of the form
```

```
db_name.str  
or  
app_name.str
```

Statements using the

```
GET_LOCK  
function are not safe for statement-based replication.
```

The patch to permit multiple locks was [contributed by Konstantin "Kostja" Osipov](#) ([MDEV-3917](#)).

## Examples

```
SELECT GET_LOCK('lock1',10);  
+-----+  
| GET_LOCK('lock1',10) |  
+-----+  
| 1 |  
+-----+  
  
SELECT IS_FREE_LOCK('lock1'), IS_USED_LOCK('lock1');  
+-----+-----+  
| IS_FREE_LOCK('lock1') | IS_USED_LOCK('lock1') |  
+-----+-----+  
| 0 | 46 |  
+-----+-----+  
  
SELECT IS_FREE_LOCK('lock2'), IS_USED_LOCK('lock2');  
+-----+-----+  
| IS_FREE_LOCK('lock2') | IS_USED_LOCK('lock2') |  
+-----+-----+  
| 1 | NULL |  
+-----+-----+
```

Multiple locks can be held:

```
SELECT GET_LOCK('lock2',10);
+-----+
| GET_LOCK('lock2',10) |
+-----+
|          1 |
+-----+



SELECT IS_FREE_LOCK('lock1'), IS_FREE_LOCK('lock2');
+-----+
| IS_FREE_LOCK('lock1') | IS_FREE_LOCK('lock2') |
+-----+
|          0 |          0 |
+-----+



SELECT RELEASE_LOCK('lock1'), RELEASE_LOCK('lock2');
+-----+
| RELEASE_LOCK('lock1') | RELEASE_LOCK('lock2') |
+-----+
|          1 |          1 |
+-----+
```

It is possible to hold the same lock recursively. This example is viewed using the `metadata_lock_info` plugin:

```
SELECT GET_LOCK('lock3',10);
+-----+
| GET_LOCK('lock3',10) |
+-----+
|          1 |
+-----+


SELECT GET_LOCK('lock3',10);
+-----+
| GET_LOCK('lock3',10) |
+-----+
|          1 |
+-----+


SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
+-----+-----+-----+-----+-----+
| THREAD_ID | LOCK_MODE           | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----+-----+-----+-----+-----+
|      46 | MDL_SHARED_NO_WRITE | NULL          | User lock  | lock3       |
+-----+-----+-----+-----+-----+


SELECT RELEASE_LOCK('lock3');
+-----+
| RELEASE_LOCK('lock3') |
+-----+
|          1 |
+-----+


SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
+-----+-----+-----+-----+-----+
| THREAD_ID | LOCK_MODE           | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----+-----+-----+-----+-----+
|      46 | MDL_SHARED_NO_WRITE | NULL          | User lock  | lock3       |
+-----+-----+-----+-----+-----+


SELECT RELEASE_LOCK('lock3');
+-----+
| RELEASE_LOCK('lock3') |
+-----+
|          1 |
+-----+


SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
Empty set (0.000 sec)
```

Timeout example: Connection 1:

```
SELECT GET_LOCK('lock4',10);
+-----+
| GET_LOCK('lock4',10) |
+-----+
|          1 |
+-----+
```

Connection 2:

```
SELECT GET_LOCK('lock4',10);
```

After 10 seconds...

```
+-----+
| GET_LOCK('lock4',10) |
+-----+
|          0 |
+-----+
```

Deadlocks are automatically detected and resolved. Connection 1:

```
SELECT GET_LOCK('lock5',10);
+-----+
| GET_LOCK('lock5',10) |
+-----+
|          1 |
+-----+
```

Connection 2:

```
SELECT GET_LOCK('lock6',10);
+-----+
| GET_LOCK('lock6',10) |
+-----+
|          1 |
+-----+
```

Connection 1:

```
SELECT GET_LOCK('lock6',10);
+-----+
| GET_LOCK('lock6',10) |
+-----+
|          0 |
+-----+
```

Connection 2:

```
SELECT GET_LOCK('lock5',10);
ERROR 1213 (40001): Deadlock found when trying to get lock; try restarting transaction
```

## See Also

- [RELEASE\\_LOCK](#)
- [IS\\_FREE\\_LOCK](#)
- [IS\\_USED\\_LOCK](#)
- [RELEASE\\_ALL\\_LOCKS](#)

## 1.1.12.8.4.2 INET6\_ATON

### Syntax

```
INET6_ATON(expr)
```

### Description

Given an IPv6 or IPv4 network address as a string, returns a binary string that represents the numeric value of the address.

No trailing zone ID's or trailing network masks are permitted. For IPv4 addresses, or IPv6 addresses with IPv4 address parts, no classful addresses or trailing port numbers are permitted and octal numbers are not supported.

The returned binary string will be `VARBINARY(16)` or `VARBINARY(4)` for IPv6 and IPv4 addresses respectively.

Returns NULL if the argument is not understood.

MariaDB starting with 10.5.0

From MariaDB 10.5.0 ,

INET6\_ATON

can take INET6 as an argument.

## Examples

```
SELECT HEX(INET6_ATON('10.0.1.1'));
+-----+
| HEX(INET6_ATON('10.0.1.1')) |
+-----+
| 0A000101 |
+-----+  
  
SELECT HEX(INET6_ATON('48f3::d432:1431:ba23:846f'));
+-----+
| HEX(INET6_ATON('48f3::d432:1431:ba23:846f')) |
+-----+
| 48F300000000000D4321431BA23846F |
+-----+
```

## See Also

- [INET6\\_NTOA\(\)](#)
- [INET\\_ATON\(\)](#)
- [INET6 Data Type](#)

## 1.1.12.8.4.3 INET6\_NTOA

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Syntax

```
INET6_NTOA(expr)
```

## Description

Given an IPv6 or IPv4 network address as a numeric binary string, returns the address as a nonbinary string in the connection character set.

The return string is lowercase, and is platform independent, since it does not use functions specific to the operating system. It has a maximum length of 39 characters.

Returns NULL if the argument is not understood.

## Examples

```

SELECT INET6_NTOA(UNHEX('0A000101'));
+-----+
| INET6_NTOA(UNHEX('0A000101')) |
+-----+
| 10.0.1.1 |
+-----+

SELECT INET6_NTOA(UNHEX('48F300000000000D4321431BA23846F'));
+-----+
| INET6_NTOA(UNHEX('48F300000000000D4321431BA23846F')) |
+-----+
| 48f3::d432:1431:ba23:846f |
+-----+

```

## See Also

- [INET6\\_ATON\(\)](#)
- [INET\\_NTOA\(\)](#)

## 1.1.12.8.4.4 INET\_ATON

### Syntax

```
INET_ATON(expr)
```

### Description

Given the dotted-quad representation of an IPv4 network address as a string, returns an integer that represents the numeric value of the address. Addresses may be 4- or 8-byte addresses.

Returns NULL if the argument is not understood.

### Examples

```

SELECT INET_ATON('192.168.1.1');
+-----+
| INET_ATON('192.168.1.1') |
+-----+
| 3232235777 |
+-----+

```

This is calculated as follows:  $192 \times 256^3 + 168 \times 256^2 + 1 \times 256 + 1$

## See Also

- [INET6\\_ATON\(\)](#)
- [INET\\_NTOA\(\)](#)

## 1.1.12.8.4.5 INET\_NTOA

### Syntax

```
INET_NTOA(expr)
```

### Description

Given a numeric IPv4 network address in network byte order (4 or 8 byte), returns the dotted-quad representation of the address as a string.

### Examples

```
SELECT INET_NTOA(3232235777);
+-----+
| INET_NTOA(3232235777) |
+-----+
| 192.168.1.1           |
+-----+
```

192.168.1.1 corresponds to 3232235777 since  $192 \times 256^3 + 168 \times 256^2 + 1 \times 256 + 1 = 3232235777$

## See Also

- [INET6\\_NTOA\(\)](#)
- [INET\\_ATON\(\)](#)

## 1.1.12.8.4.6 IS\_FREE\_LOCK

### Syntax

```
IS_FREE_LOCK(str)
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

### Description

Checks whether the lock named

```
str  
is free to use (that is, not locked). Returns  
1  
if the lock is free (no one is using the lock),  
0  
if the lock is in use, and  
NULL  
if an error occurs (such as an incorrect argument, like an empty string or  
NULL  
).  
str  
is case insensitive.
```

If the [metadata\\_lock\\_info](#) plugin is installed, the [Information Schema metadata\\_lock\\_info](#) table contains information about locks of this kind (as well as [metadata locks](#) ).

Statements using the

```
IS_FREE_LOCK  
function are not safe for statement-based replication.
```

## See Also

- [GET\\_LOCK](#)
- [RELEASE\\_LOCK](#)
- [IS\\_USED\\_LOCK](#)
- [RELEASE\\_ALL\\_LOCKS](#)

## 1.1.12.8.4.7 IS\_IPV4

### Syntax

```
IS_IPV4(expr)
```

### Description

If the expression is a valid IPv4 address, returns 1, otherwise returns 0.

`IS_IPV4()` is stricter than `INET_ATON()`, but as strict as `INET6_ATON()`, in determining the validity of an IPv4 address. This implies that if `IS_IPV4` returns 1, the same expression will always return a non-NULL result when passed to `INET_ATON()`, but that the reverse may not apply.

## Examples

```
SELECT IS_IPV4('1110.0.1.1');
+-----+
| IS_IPV4('1110.0.1.1') |
+-----+
|          0          |
+-----+  
  
SELECT IS_IPV4('48f3::d432:1431:ba23:846f');
+-----+
| IS_IPV4('48f3::d432:1431:ba23:846f') |
+-----+
|          0          |
+-----+
```

## 1.1.12.8.4.8 IS\_IPV4\_COMPAT

### Syntax

```
IS_IPV4_COMPAT(expr)
```

### Description

Returns 1 if a given numeric binary string IPv6 address, such as returned by `INET6_ATON()`, is IPv4-compatible, otherwise returns 0.

MariaDB starting with 10.5.0

From MariaDB 10.5.0, when the argument is not `INET6`, automatic implicit `CAST` to `INET6` is applied. As a consequence,

now understands arguments in both text representation and binary(16) representation. Before MariaDB 10.5.0, the function understood only binary(16) representation.

## Examples

```
SELECT IS_IPV4_COMPAT(INET6_ATON('::10.0.1.1'));
+-----+
| IS_IPV4_COMPAT(INET6_ATON('::10.0.1.1')) |
+-----+
|          1          |
+-----+  
  
SELECT IS_IPV4_COMPAT(INET6_ATON('::48f3::d432:1431:ba23:846f'));
+-----+
| IS_IPV4_COMPAT(INET6_ATON('::48f3::d432:1431:ba23:846f')) |
+-----+
|          0          |
+-----+
```

## 1.1.12.8.4.9 IS\_IPV4\_MAPPED

### Syntax

```
IS_IPV4_MAPPED(expr)
```

### Description

Returns 1 if a given a numeric binary string IPv6 address, such as returned by `INET6_ATON()`, is a valid IPv4-mapped address, otherwise returns 0.

MariaDB starting with 10.5.0

From MariaDB 10.5.0 , when the argument is not INET6 , automatic implicit CAST to INET6 is applied. As a consequence,

IS\_IPV4\_MAPPED

now understands arguments in both text representation and binary(16) representation. Before MariaDB 10.5.0 , the function understood only binary(16) representation.

## Examples

```
SELECT IS_IPV4_MAPPED(INET6_ATON('::10.0.1.1'));
+-----+
| IS_IPV4_MAPPED(INET6_ATON('::10.0.1.1')) |
+-----+
|          0 |
+-----+

SELECT IS_IPV4_MAPPED(INET6_ATON('::ffff:10.0.1.1'));
+-----+
| IS_IPV4_MAPPED(INET6_ATON('::ffff:10.0.1.1')) |
+-----+
|          1 |
+-----+
```

## 1.1.12.8.4.10 IS\_IPV6

### Syntax

```
IS_IPV6(expr)
```

### Description

Returns 1 if the expression is a valid IPv6 address specified as a string, otherwise returns 0. Does not consider IPv4 addresses to be valid IPv6 addresses.

### Examples

```
SELECT IS_IPV6('48f3::d432:1431:ba23:846f');
+-----+
| IS_IPV6('48f3::d432:1431:ba23:846f') |
+-----+
|          1 |
+-----+
1 row in set (0.02 sec)

SELECT IS_IPV6('10.0.1.1');
+-----+
| IS_IPV6('10.0.1.1') |
+-----+
|          0 |
+-----+
```

### See Also

- [INET6 data type](#)
- [INET6\\_ATON](#)
- [INET6\\_NTOA](#)

## 1.1.12.8.4.11 IS\_USED\_LOCK

### Syntax

```
IS_USED_LOCK(str)
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [See Also](#)

## Description

Checks whether the lock named

```
str  
is in use (that is, locked). If so, it returns the connection identifier of the client that holds the lock. Otherwise, it returns  
NULL  
. .  
str  
is case insensitive.
```

If the `metadata_lock_info` plugin is installed, the [Information Schema metadata\\_lock\\_info](#) table contains information about locks of this kind (as well as [metadata locks](#) ).

Statements using the

```
IS_USED_LOCK  
function are not safe for statement-based replication.
```

## See Also

- [GET\\_LOCK](#)
- [RELEASE\\_LOCK](#)
- [IS\\_FREE\\_LOCK](#)
- [RELEASE\\_ALL\\_LOCKS](#)

## 1.1.12.8.4.12 MASTER\_GTID\_WAIT

### Syntax

```
MASTER_GTID_WAIT(gtid-list[, timeout])
```

### Description

This function takes a string containing a comma-separated list of [global transaction id's](#) (similar to the value of, for example, `gtid_binlog_pos` ). It waits until the value of `gtid_slave_pos` has the same or higher seq\_no within all replication domains specified in the gtid-list; in other words, it waits until the slave has reached the specified GTID position.

An optional second argument gives a timeout in seconds. If the timeout expires before the specified GTID position is reached, then the function returns -1. Passing NULL or a negative number for the timeout means no timeout, and the function will wait indefinitely.

If the wait completes without a timeout, 0 is returned. Passing NULL for the gtid-list makes the function return NULL immediately, without waiting.

The gtid-list may be the empty string, in which case `MASTER_GTID_WAIT()` returns immediately. If the gtid-list contains fewer domains than `gtid_slave_pos` , then only those domains are waited upon. If gtid-list contains a domain that is not present in `@@gtid_slave_pos` , then `MASTER_GTID_WAIT()` will wait until an event containing such domain\_id arrives on the slave (or until timed out or killed).

`MASTER_GTID_WAIT()` can be useful to ensure that a slave has caught up to a master. Simply take the value of `gtid_binlog_pos` on the master, and use it in a `MASTER_GTID_WAIT()` call on the slave; when the call completes, the slave will have caught up with that master position.

`MASTER_GTID_WAIT()` can also be used in client applications together with the `last_gtid` session variable. This is useful in a read-scaleout [replication](#) setup, where the application writes to a single master but divides the reads out to a number of slaves to distribute the load. In such a setup, there is a risk that an application could first do an update on the master, and then a bit later do a read on a slave, and if the slave is not fast enough, the data read from the slave might not include the update just made, possibly confusing the application and/or the end-user. One way to avoid this is to request the value of `last_gtid` on the master just after the update. Then before doing the read on the slave, do a `MASTER_GTID_WAIT()` on the value obtained from the master; this will ensure that the read is not performed until the slave has replicated sufficiently far for the update to have become visible.

Note that `MASTER_GTID_WAIT()` can be used even if the slave is configured not to use GTID for connections ([CHANGE MASTER TO master\\_use\\_gtid=no](#) ). This is because from MariaDB 10, GTIDs are always logged on the master server, and always recorded on the slave servers.

### Differences to MASTER\_POS\_WAIT()

- `MASTER_GTID_WAIT()` is global; it waits for any master connection to reach the specified GTID position. `MASTER_POS_WAIT()` works only against a specific connection. This also means that while `MASTER_POS_WAIT()` aborts if its master connection is terminated with `STOP SLAVE` or due to an error, `MASTER_GTID_WAIT()` continues to wait while slaves are stopped.
- `MASTER_GTID_WAIT()` can take its timeout as a floating-point value, so a timeout in fractional seconds is supported, eg. `MASTER_GTID_WAIT("0-1-100", 0.5)`. (The minimum wait is one microsecond, 0.000001 seconds).

- `MASTER_GTID_WAIT()` allows one to specify a timeout of zero in order to do a non-blocking check to see if the slaves have progressed to a specific GTID position (`MASTER_POS_WAIT()`) takes a zero timeout as meaning an infinite wait). To do an infinite `MASTER_GTID_WAIT()`, specify a negative timeout, or omit the timeout argument.
- `MASTER_GTID_WAIT()` does not return the number of events executed since the wait started, nor does it return `NULL` if a slave thread is stopped. It always returns either `0` for successful wait completed, or `-1` for timeout reached (or `NULL` if the specified gtid-pos is `NULL`).

Since `MASTER_GTID_WAIT()` looks only at the `seq_no` part of the GTIDs, not the `server_id`, care is needed if a slave becomes diverged from another server so that two different GTIDs with the same `seq_no` (in the same domain) arrive at the same server. This situation is in any case best avoided; setting `gtid_strict_mode` is recommended, as this will prevent any such out-of-order sequence numbers from ever being replicated on a slave.

## 1.1.12.8.4.13 MASTER\_POS\_WAIT

### Syntax

```
MASTER_POS_WAIT(log_name,log_pos[,timeout,[ "connection_name" ]])
```

### Description

This function is useful in [replication](#) for controlling primary/replica synchronization. It blocks until the replica has read and applied all updates up to the specified position (

`log_name,log_pos`

) in the primary log. The return value is the number of log events the replica had to wait for to advance to the specified position. The function returns `NULL` if the replica SQL thread is not started, the replica's primary information is not initialized, the arguments are incorrect, or an error occurs. It returns `-1` if the timeout has been exceeded. If the replica SQL thread stops while

`MASTER_POS_WAIT()`

is waiting, the function returns `NULL`. If the replica is past the specified position, the function returns immediately.

If a

`timeout`

`value` is specified,

`MASTER_POS_WAIT()`

stops waiting when

`timeout`

seconds have elapsed.

`timeout`

must be greater than `0`; a zero or negative

`timeout`

means no

`timeout`

.

The

`connection_name`

is used when you are using [multi-source-replication](#). If you don't specify it, it's set to the value of the `default_master_connection` system variable.

Statements using the `MASTER_POS_WAIT()` function are not [safe for replication](#).

## 1.1.12.8.4.14 NAME\_CONST

### Syntax

```
NAME_CONST(name,value)
```

### Description

Returns the given value. When used to produce a result set column,

`NAME_CONST()`

causes the column to have the given name. The arguments should be constants.

This function is used internally when replicating stored procedures. It makes little sense to use it explicitly in SQL statements, and it was not supposed to be used like that.

```
SELECT NAME_CONST('myname', 14);
+-----+
| myname |
+-----+
|   14   |
+-----+
```

## 1.1.12.8.4.15

## 1.1.12.8.4.16 RELEASE\_LOCK

### Syntax

```
RELEASE_LOCK(str)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

Releases the lock named by the string

str

that was obtained with [GET\\_LOCK\(\)](#). Returns 1 if the lock was released, 0 if the lock was not established by this thread (in which case the lock is not released), and

NULL

if the named lock did not exist. The lock does not exist if it was never obtained by a call to

[GET\\_LOCK\(\)](#)

or if it has previously been released.

```
str  
is case insensitive. If  
str  
is an empty string or  
NULL  
,  
RELEASE_LOCK()  
returns  
NULL  
and does nothing.
```

Statements using the RELEASE\_LOCK() function are not [safe for replication](#).

The [DO statement](#) is convenient to use with

```
RELEASE_LOCK()
```

### Examples

Connection1:

```
SELECT GET_LOCK('lock1',10);
+-----+
| GET_LOCK('lock1',10) |
+-----+
|           1          |
+-----+
```

Connection 2:

```
SELECT GET_LOCK('lock2',10);
```

+	-----+
	GET_LOCK('lock2',10)
+	-----+
	1
+	-----+

## Connection 1:

```
SELECT RELEASE_LOCK('lock1'), RELEASE_LOCK('lock2'), RELEASE_LOCK('lock3');

+-----+-----+
| RELEASE_LOCK('lock1') | RELEASE_LOCK('lock2') | RELEASE_LOCK('lock3') |
+-----+-----+
|           1 |             0 |        NULL |
+-----+-----+
```

From MariaDB 10.0.2 , it is possible to hold the same lock recursively. This example is viewed using the `metadata_lock_info` plugin:

```
SELECT GET_LOCK('lock3',10);
+-----+
| GET_LOCK('lock3',10) |
+-----+
|          1 |
+-----+


SELECT GET_LOCK('lock3',10);
+-----+
| GET_LOCK('lock3',10) |
+-----+
|          1 |
+-----+


SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
+-----+-----+-----+-----+-----+
| THREAD_ID | LOCK_MODE           | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----+-----+-----+-----+-----+
|      46 | MDL_SHARED_NO_WRITE | NULL          | User lock  | lock3       |          |
+-----+-----+-----+-----+-----+


SELECT RELEASE_LOCK('lock3');
+-----+
| RELEASE_LOCK('lock3') |
+-----+
|          1 |
+-----+


SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
+-----+-----+-----+-----+-----+
| THREAD_ID | LOCK_MODE           | LOCK_DURATION | LOCK_TYPE | TABLE_SCHEMA | TABLE_NAME |
+-----+-----+-----+-----+-----+
|      46 | MDL_SHARED_NO_WRITE | NULL          | User lock  | lock3       |          |
+-----+-----+-----+-----+-----+


SELECT RELEASE_LOCK('lock3');
+-----+
| RELEASE_LOCK('lock3') |
+-----+
|          1 |
+-----+


SELECT * FROM INFORMATION_SCHEMA.METADATA_LOCK_INFO;
Empty set (0.000 sec)
```

#### See Also

- GET\_LOCK
  - IS\_FREE\_LOCK
  - IS\_USED\_LOCK
  - RELEASE\_ALL\_LOCKS

## 1.1.12.8.4.17 SLEEP

## Syntax

```
SLEEP(duration)
```

## Description

Sleeps (pauses) for the number of seconds given by the duration argument, then returns

- 0
- . If
- SLEEP()
- is interrupted, it returns
- 1
- . The duration may have a fractional part given in microseconds.

Statements using the SLEEP() function are not [safe for replication](#).

## Example

```
SELECT SLEEP(5.5);
+-----+
| SLEEP(5.5) |
+-----+
|      0 |
+-----+
1 row in set (5.50 sec)
```

## 1.1.12.8.4.18 SYS\_GUID

MariaDB starting with [10.6.1](#)

The SYS\_GUID function was introduced in [MariaDB 10.6.1](#) to enhance Oracle compatibility. Similar functionality can be achieved with the [UUID](#) function.

## Syntax

```
SYS_GUID()
```

## Description

Returns a 16-byte globally unique identifier (GUID), similar to the [UUID](#) function, but without the

character.

## Example

```
SELECT SYS_GUID();
+-----+
| SYS_GUID()           |
+-----+
| 2C574E45BA2811EBB265F859713E4BE4 |
+-----+
```

## See Also

- [UUID](#)
- [UUID\\_SHORT](#)
- [UUID data type](#)

## 1.1.12.8.4.19 UUID

## Syntax

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Returns a Universally Unique Identifier (UUID).

A UUID is designed as a number that is globally unique in space and time. Two calls to

```
UUID()
```

are expected to generate two different values, even if these calls are performed on two separate computers that are not connected to each other.

**UUID()** results are intended to be unique, but cannot always be relied upon to unpredictable and unguessable, so should not be relied upon for these purposes.

A UUID is a 128-bit number represented by a utf8 string of five hexadecimal numbers in

```
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeee
```

format:

- The first three numbers are generated from a timestamp.
- The fourth number preserves temporal uniqueness in case the timestamp value loses monotonicity (for example, due to daylight saving time).
- The fifth number is an IEEE 802 node number that provides spatial uniqueness. A random number is substituted if the latter is not available (for example, because the host computer has no Ethernet card, or we do not know how to find the hardware address of an interface on your operating system). In this case, spatial uniqueness cannot be guaranteed. Nevertheless, a collision should have very low probability.

Currently, the MAC address of an interface is taken into account only on FreeBSD and Linux. On other operating systems, MariaDB uses a randomly generated 48-bit number.

Statements using the `UUID()` function are not [safe for replication](#).

The results are generated according to the "DCE 1.1:Remote Procedure Call" (Appendix A) CAE (Common Applications Environment) Specifications published by The Open Group in October 1997 ( [Document Number C706](#) ).

## Examples

```
SELECT UUID();
+-----+
| UUID()           |
+-----+
| cd41294a-afb0-11df-bc9b-00241dd75637 |
+-----+
```

## See Also

- [UUID\\_SHORT\(\)](#) - Return short (64 bit) Universal Unique Identifier
- [SYS\\_GUID](#) - UUID without the
- character for Oracle compatibility
- [UUID data type](#)

## 1.1.12.8.4.20 UUID\_SHORT

### Syntax

```
UUID_SHORT()
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

# Description

Returns a "short" universally unique identifier as a 64-bit unsigned integer (rather than a string-form 128-bit identifier as returned by the [UUID\(\)](#) function).

The value of

`UUID_SHORT()`

is guaranteed to be unique if the following conditions hold:

- The server\_id of the current host is unique among your set of master and slave servers
- - server\_id  
is between 0 and 255
- You don't set back your system time for your server between mysqld restarts
- You do not invoke
  - `UUID_SHORT()`  
on average more than 16 million times per second between mysqld restarts

The `UUID_SHORT()` return value is constructed this way:

```
(server_id & 255) << 56
+ (server_startup_time_in_seconds << 24)
+ incremented_variable++;
```

Statements using the `UUID_SHORT()` function are not [safe for statement-based replication](#).

## Examples

```
SELECT UUID_SHORT();
+-----+
| UUID_SHORT() |
+-----+
| 21517162376069120 |
+-----+
```

```
create table t1 (a bigint unsigned default(uuid_short()) primary key);
insert into t1 values(),();
select * from t1;
+-----+
| a      |
+-----+
| 98113699159474176 |
| 98113699159474177 |
+-----+
```

## See Also

- [UUID\(\)](#) ; Return full (128 bit) Universally Unique Identifier
- [AUTO\\_INCREMENT](#)
- [Sequences](#) - an alternative to auto\_increment available from [MariaDB 10.3](#)
- [SYS\\_GUID](#) - UUID without the
  - character for Oracle compatibility
- [UUID data type](#)

## 1.1.12.8.4.21 VALUES / VALUE

### Syntax

MariaDB starting with [10.3.3](#)

```
VALUE(col_name)
```

MariaDB until [10.3.2](#)

```
VALUES(col_name)
```

## Description

In an `INSERT ... ON DUPLICATE KEY UPDATE` statement, you can use the

```
VALUES(col_name)
function in the UPDATE clause to refer to column values from the INSERT portion of the statement. In other words,
VALUES(col_name)
in the
UPDATE
clause refers to the value of col_name that would be inserted, had no duplicate-key conflict occurred. This function is especially useful in
multiple-row inserts.
```

The

```
VALUES()
function is meaningful only in
INSERT ... ON DUPLICATE KEY UPDATE
statements and returns
NULL
otherwise.
```

In MariaDB 10.3.3 this function was renamed to

```
VALUE()
, because it's incompatible with the standard Table Value Constructors syntax, implemented in MariaDB 10.3.3 .
```

The

```
VALUES()
function can still be used even from MariaDB 10.3.3 , but only in
INSERT ... ON DUPLICATE KEY UPDATE
statements; it's a syntax error otherwise.
```

## Examples

MariaDB starting with 10.3.3

```
INSERT INTO t (a,b,c) VALUES (1,2,3),(4,5,6)
ON DUPLICATE KEY UPDATE c=VALUE(a)+VALUE(b);
```

MariaDB until 10.3.2

```
INSERT INTO t (a,b,c) VALUES (1,2,3),(4,5,6)
ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);
```

## 1.1.12.9 Special Functions

### 1.1.12.9.1 Dynamic Columns Functions

#### 1.1.12.9.2 Galera Functions

##### 1.1.12.9.2.1 WSREP\_LAST\_SEEN\_GTID

MariaDB starting with 10.4.2

WSREP\_LAST\_SEEN\_GTID was added as part of Galera 4 in MariaDB 10.4.2 .

## Syntax

```
WSREP_LAST_SEEN_GTID()
```

## Description

Returns the Global Transaction ID of the most recent write transaction observed by the client.

The result can be useful to determine the transaction to provide to [WSREP\\_SYNC\\_WAIT\\_UPTO\\_GTID](#) for waiting and unblocking purposes.

## 1.1.12.9.2.2 WSREP\_LAST\_WRITTEN\_GTID

MariaDB starting with [10.4.2](#)

`WSREP_LAST_WRITTEN_GTID` was added as part of Galera 4 in [MariaDB 10.4.2](#).

### Syntax

```
WSREP_LAST_WRITTEN_GTID()
```

### Description

Returns the [Global Transaction ID](#) of the most recent write transaction performed by the client.

## 1.1.12.9.2.3 WSREP\_SYNC\_WAIT\_UPTO\_GTID

MariaDB starting with [10.4.2](#)

`WSREP_SYNC_WAIT_UPTO_GTID` was added as part of Galera 4 in [MariaDB 10.4.2](#).

### Syntax

```
WSREP_SYNC_WAIT_UPTO_GTID(gtid[,timeout])
```

### Description

Blocks the client until the transaction specified by the given [Global Transaction ID](#) is applied and committed by the node.

The optional *timeout* argument can be used to specify a block timeout in seconds. If not provided, the timeout will be indefinite.

Returns the node that applied and committed the Global Transaction ID,

`ER_LOCAL_WAIT_TIMEOUT`

if the function is timed out before this, or

`ER_WRONG_ARGUMENTS`

if the function is given an invalid GTID.

The result from [WSREP\\_LAST\\_SEEN\\_GTID](#) can be useful to determine the transaction to provide to `WSREP_SYNC_WAIT_UPTO_GTID` for waiting and unblocking purposes.

## 1.1.12.9.3 Geographic Functions

### 1.1.12.9.3.1 Geometry Constructors

#### 1.1.12.9.3.1.1 BUFFER

A synonym for [ST\\_BUFFER](#).

#### 1.1.12.9.3.1.2 CONVEXHULL

A synonym for [ST\\_CONVEXHULL](#).

#### 1.1.12.9.3.1.3 GEOMETRYCOLLECTION

### Syntax

```
GeometryCollection(g1,g2,...)
```

### Description

Constructs a [WKB](#) `GeometryCollection`. If any argument is not a well-formed WKB representation of a geometry, the return value is `NULL`.

## Examples

```
CREATE TABLE gis_geometrycollection (g GEOMETRYCOLLECTION);
SHOW FIELDS FROM gis_geometrycollection;
INSERT INTO gis_geometrycollection VALUES
(GeomCollFromText('GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(0 0,10 10)))'),
(GeometryFromWKB(AsWKB(GeometryCollection(Point(44, 6), LineString(Point(3, 6), Point(7, 9))))),
(GeomFromText('GeometryCollection()')),
(GeomFromText('GeometryCollection EMPTY')));
```

### 1.1.12.9.3.1.4 LINESTRING

#### Syntax

```
LineString(pt1,pt2,...)
```

#### Description

Constructs a [WKB](#) LineString value from a number of WKB [Point](#) arguments. If any argument is not a WKB Point, the return value is

NULL  
. If the number of [Point](#) arguments is less than two, the return value is  
NULL

## Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';
SELECT AsText(EndPoint(GeomFromText(@ls)));
+-----+
| AsText(EndPoint(GeomFromText(@ls))) |
+-----+
| POINT(3 3)                         |
+-----+
CREATE TABLE gis_line (g LINESTRING);
INSERT INTO gis_line VALUES
(LineFromText('LINESTRING(0 0,0 10,10 0)'), 
(LineStringFromText('LINESTRING(10 10,20 10,20 20,10 20,10 10)'), 
(LineStringFromWKB(AsWKB(LineString(Point(10, 10), Point(40, 10)))));
```

### 1.1.12.9.3.1.5 MULTILINESTRING

#### Syntax

```
MultiLineString(ls1,ls2,...)
```

#### Description

Constructs a WKB MultiLineString value using [WKB LineString](#) arguments. If any argument is not a WKB LineString, the return value is  
NULL

#### Example

```

CREATE TABLE gis_multi_line (g MULTILINESTRING);
INSERT INTO gis_multi_line VALUES
(MultiLineStringFromText('MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48))')),
(MLineFromText('MULTILINESTRING((10 48,10 21,10 0))')),
(MLineFromWKB(AsWKB(MultiLineString(LineString(Point(1, 2), Point(3, 5)), LineString(Point(2, 5),Point(5, 8),Point(21, 7))))));

```

## 1.1.12.9.3.1.6 MULTIPOINT

### Syntax

```
MultiPoint(pt1,pt2,...)
```

### Description

Constructs a [WKB](#) MultiPoint value using WKB [Point](#) arguments. If any argument is not a WKB Point, the return value is `NULL`.

### Examples

```

SET @g = ST_GEOMFROMTEXT('MultiPoint( 1 1, 2 2, 5 3, 7 2, 9 3, 8 4, 6 6, 6 9, 4 9, 1 5 )');

CREATE TABLE gis_multi_point (g MULTIPOINT);
INSERT INTO gis_multi_point VALUES
(MultiPointFromText('MULTIPOINT(0 0,10 10,10 20,20 20)'),,
(MPointFromText('MULTIPOINT(1 1,11 11,11 21,21 21)'),,
(MPointFromWKB(AsWKB(MultiPoint(Point(3, 6), Point(4, 10))))));

```

## 1.1.12.9.3.1.7 MULTIPOLYGON

### Syntax

```
MultiPolygon(poly1,poly2,...)
```

### Description

Constructs a [WKB](#) MultiPolygon value from a set of WKB [Polygon](#) arguments. If any argument is not a WKB Polygon, the return value is `NULL`.

### Example

```

CREATE TABLE gis_multi_polygon (g MULTIPOLYGON);
INSERT INTO gis_multi_polygon VALUES
(MultiPolygonFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67 13,59 13,
(MPolyFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67 13,59 13,59 18))
(MPolyFromWKB(AsWKB(MultiPolygon(Polygon(LineString(Point(0, 3), Point(3, 3), Point(3, 0), Point(0, 3))))));

```

## 1.1.12.9.3.1.8 POINT

### Syntax

```
Point(x,y)
```

### Description

Constructs a [WKB](#) Point using the given coordinates.

## Examples

```
SET @g = ST_GeomFromText('Point(1 1)';

CREATE TABLE gis_point  (g POINT);
INSERT INTO gis_point VALUES
  (PointFromText('POINT(10 10'))),
  (PointFromText('POINT(20 10'))),
  (PointFromText('POINT(20 20'))),
  (PointFromWKB(AsWKB(PointFromText('POINT(10 20')))));
```

### 1.1.12.9.3.1.9 PointOnSurface

A synonym for [ST\\_PointOnSurface](#).

### 1.1.12.9.3.1.10 POLYGON

#### Syntax

```
Polygon(ls1,ls2,...)
```

#### Description

Constructs a WKB Polygon value from a number of [WKB LineString](#) arguments. If any argument does not represent the WKB of a LinearRing (that is, not a closed and simple LineString) the return value is

NULL

Note that according to the OpenGIS standard, a POLYGON should have exactly one ExteriorRing and all other rings should lie within that ExteriorRing and thus be the InteriorRings. Practically, however, some systems, including MariaDB's, permit polygons to have several 'ExteriorRings'. In the case of there being multiple, non-overlapping exterior rings [ST\\_NUMINTERIORRINGS\(\)](#) will return 1.

## Examples

```
SET @g = ST_GeomFromText('POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1))';

CREATE TABLE gis_polygon  (g POLYGON);
INSERT INTO gis_polygon VALUES
  (PolygonFromText('POLYGON((10 10,20 10,20 20,10 20,10 10))')),
  (PolyFromText('POLYGON((0 0,50 0,50 50,0 50,0 0), (10 10,20 10,20 20,10 20,10 10))'),
  (PolyFromWKB(AsWKB(Polygon(LineString(Point(0, 0), Point(30, 0), Point(30, 30), Point(0, 0))))));
```

Non-overlapping 'polygon':

```
SELECT ST_NumInteriorRings(ST_PolyFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),
  (-1 -1,-5 -5,-1 -5,-1 -1))') AS NumInteriorRings;
+-----+
| NumInteriorRings |
+-----+
|          1 |
+-----+
```

### 1.1.12.9.3.1.11 ST\_BUFFER

#### Syntax

```
ST_BUFFER(g1,r)
BUFFER(g1,r)
```

#### Description

Returns a geometry that represents all points whose distance from geometry

*g1*

is less than or equal to distance, or radius,

$r$

.

Uses for this function could include creating for example a new geometry representing a buffer zone around an island.

BUFFER() is a synonym.

## Examples

Determining whether a point is within a buffer zone:

```
SET @g1 = ST_GEOFROMTEXT('POLYGON((10 10, 10 20, 20 20, 20 10, 10 10))');

SET @g2 = ST_GEOFROMTEXT('POINT(8 8)');

SELECT ST_WITHIN(@g2,ST_BUFFER(@g1,5));
+-----+
| ST_WITHIN(@g2,ST_BUFFER(@g1,5)) |
+-----+
| 1 |
+-----+

SELECT ST_WITHIN(@g2,ST_BUFFER(@g1,1));
+-----+
| ST_WITHIN(@g2,ST_BUFFER(@g1,1)) |
+-----+
| 0 |
+-----+
```

## 1.1.12.9.3.1.12 ST\_CONVEXHULL

MariaDB starting with 10.1.2

ST\_ConvexHull() was introduced in MariaDB 10.1.2

## Syntax

```
ST_ConvexHull(g)
ConvexHull(g)
```

## Description

Given a geometry, returns a geometry that is the minimum convex geometry enclosing all geometries within the set. Returns NULL if the geometry value is NULL or an empty value.

ST\_ConvexHull() and ConvexHull() are synonyms.

## Examples

The ConvexHull of a single point is simply the single point:

```
SET @g = ST_GEOFROMTEXT('Point(0 0)');

SELECT ST_ASTEXT(ST_CONVEXHULL(@g));
+-----+
| ST_ASTEXT(ST_CONVEXHULL(@g)) |
+-----+
| POINT(0 0) |
+-----+
```

```

SET @g = ST_GEOFROMTEXT('MultiPoint(0 0, 1 2, 2 3)');

SELECT ST_ASTEXT(ST_CONVEXHULL(@g));
+-----+
| ST_ASTEXT(ST_CONVEXHULL(@g)) |
+-----+
| POLYGON((0 0,1 2,2 3,0 0)) |
+-----+

```

```

SET @g = ST_GEOFROMTEXT('MultiPoint( 1 1, 2 2, 5 3, 7 2, 9 3, 8 4, 6 6, 6 9, 4 9, 1 5 )');

SELECT ST_ASTEXT(ST_CONVEXHULL(@g));
+-----+
| ST_ASTEXT(ST_CONVEXHULL(@g)) |
+-----+
| POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1)) |
+-----+

```

## 1.1.12.9.3.1.13 ST\_INTERSECTION

### Syntax

```
ST_INTERSECTION(g1,g2)
```

### Description

Returns a geometry that is the intersection, or shared portion, of geometry

*g1*  
and geometry  
*g2*

### Examples

```

SET @g1 = ST_GEOFROMTEXT('POINT(2 1)');

SET @g2 = ST_GEOFROMTEXT('LINESTRING(2 1, 0 2)');

SELECT ASTEXT(ST_INTERSECTION(@g1,@g2));
+-----+
| ASTEXT(ST_INTERSECTION(@g1,@g2)) |
+-----+
| POINT(2 1) |
+-----+

```

## 1.1.12.9.3.1.14 ST\_POINTONSURFACE

MariaDB starting with [10.1.2](#)

ST\_POINTONSURFACE() was introduced in [MariaDB 10.1.2](#).

### Syntax

```
ST_PointOnSurface(g)
PointOnSurface(g)
```

### Description

Given a geometry, returns a [POINT](#) guaranteed to intersect a surface. However, see [MDEV-7514](#).

ST\_PointOnSurface() and PointOnSurface() are synonyms.

## 1.1.12.9.3.1.15 ST\_SYMDIFFERENCE

### Syntax

```
ST_SYMDIFFERENCE(g1,g2)
```

### Description

Returns a geometry that represents the portions of geometry

*g1*  
and geometry  
*g2*  
that don't intersect.

### Examples

```
SET @g1 = ST_GeomFromText('LINESTRING(10 20, 10 40)');

SET @g2 = ST_GeomFromText('LINESTRING(10 15, 10 25)');

SELECT ASTEXT(ST_SYMDIFFERENCE(@g1,@g2));
+-----+
| ASTEXT(ST_SYMDIFFERENCE(@g1,@g2)) |
+-----+
| MULTILINESTRING((10 15,10 20),(10 25,10 40)) |
+-----+

SET @g2 = ST_GeomFromText('LINESTRING(10 20, 10 41)';

SELECT ASTEXT(ST_SYMDIFFERENCE(@g1,@g2));
+-----+
| ASTEXT(ST_SYMDIFFERENCE(@g1,@g2)) |
+-----+
| LINESTRING(10 40,10 41) |
+-----+
```

## 1.1.12.9.3.1.16 ST\_UNION

### Syntax

```
ST_UNION(g1,g2)
```

### Description

Returns a geometry that is the union of the geometry

*g1*  
and geometry  
*g2*

### Examples

```
SET @g1 = GEOMFROMTEXT('POINT (0 2)');

SET @g2 = GEOMFROMTEXT('POINT (2 0)');

SELECT ASTEXT(ST_UNION(@g1,@g2));
+-----+
| ASTEXT(ST_UNION(@g1,@g2)) |
+-----+
| MULTIPOINT(2 0,0 2) |
+-----+
```

```
SET @g1 = GEOMFROMTEXT('POLYGON((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GEOMFROMTEXT('POLYGON((2 2,4 2,4 4,2 4,2 2))');
SELECT ASTEXT(ST_UNION(@g1,@g2));
+-----+
| ASTEXT(ST_UNION(@g1,@g2)) |
+-----+
| POLYGON((0 0,0 3,2 3,2 4,4 4,4 2,3 2,3 0,0 0)) |
+-----+
```

## 1.1.12.9.3.2 Geometry Properties

### 1.1.12.9.3.2.1 BOUNDARY

A synonym for [ST\\_BOUNDARY](#).

### 1.1.12.9.3.2.2 DIMENSION

A synonym for [ST\\_DIMENSION](#).

### 1.1.12.9.3.2.3 ENVELOPE

A synonym for [ST\\_ENVELOPE](#).

### 1.1.12.9.3.2.4 GeometryN

A synonym for [ST\\_GeometryN](#).

### 1.1.12.9.3.2.5 GeometryType

A synonym for [ST\\_GeometryType](#).

### 1.1.12.9.3.2.6 IsClosed

A synonym for [ST\\_IsClosed](#).

### 1.1.12.9.3.2.7 IsEmpty

A synonym for [ST\\_IsEmpty](#).

### 1.1.12.9.3.2.8 IsRing

A synonym for [ST\\_IsRing](#).

### 1.1.12.9.3.2.9 IsSimple

A synonym for [ST\\_IsSimple](#).

### 1.1.12.9.3.2.10 NumGeometries

A synonym for [ST\\_NumGeometries](#).

### 1.1.12.9.3.2.11 SRID

A synonym for [ST\\_SRID](#).

### 1.1.12.9.3.2.12 ST\_BOUNDARY

MariaDB starting with [10.1.2](#)

The `ST_BOUNDARY` function was introduced in [MariaDB 10.1.2](#)

## Syntax

```
ST_BOUNDARY(g)
BOUNDARY(g)
```

## Description

Returns a geometry that is the closure of the combinatorial boundary of the geometry value

*g*

.  
BOUNDARY() is a synonym.

## Examples

```
SELECT ST_AsText(ST_Boundary(ST_GeomFromText('LINESTRING(3 3,0 0, -3 3)')));  
+-----+  
| ST_AsText(ST_Boundary(ST_GeomFromText('LINESTRING(3 3,0 0, -3 3)'))) |  
+-----+  
| MULTIPOINT(3 3,-3 3) |  
+-----+  
  
SELECT ST_AsText(ST_Boundary(ST_GeomFromText('POLYGON((3 3,0 0, -3 3, 3 3))')));  
+-----+  
| ST_AsText(ST_Boundary(ST_GeomFromText('POLYGON((3 3,0 0, -3 3, 3 3))'))) |  
+-----+  
| LINESTRING(3 3,0 0,-3 3,3 3) |  
+-----+
```

## 1.1.12.9.3.2.13 ST\_DIMENSION

### Syntax

```
ST_Dimension(g)  
Dimension(g)
```

## Description

Returns the inherent dimension of the geometry value

*g*

. The result can be

Dimension	Definition
-1	empty geometry
0	geometry with no length or area
1	geometry with no area but nonzero length
2	geometry with nonzero area

ST\_Dimension()  
and  
Dimension()  
are synonyms.

## Examples

```
SELECT Dimension(GeomFromText('LineString(1 1,2 2)'));
+-----+
| Dimension(GeomFromText('LineString(1 1,2 2)')) |
+-----+
| 1 |
+-----+
```

## 1.1.12.9.3.2.14 ST\_ENVELOPE

### Syntax

```
ST_ENVELOPE(g)
ENVELOPE(g)
```

### Description

Returns the Minimum Bounding Rectangle (MBR) for the geometry value

g  
. The result is returned as a Polygon value.

The polygon is defined by the corner points of the bounding box:

```
POLYGON((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
```

ST\_ENVELOPE()  
and  
ENVELOPE()  
are synonyms.

### Examples

```
SELECT AsText(ST_ENVELOPE(GeomFromText('LineString(1 1,4 4)')));
+-----+
| AsText(ST_ENVELOPE(GeomFromText('LineString(1 1,4 4)'))) |
+-----+
| POLYGON((1 1,4 1,4 4,1 4,1 1)) |
+-----+
```

## 1.1.12.9.3.2.15 ST\_GEOEMTRYN

### Syntax

```
ST_GeometryN(gc,N)
GeometryN(gc,N)
```

### Description

Returns the N-th geometry in the GeometryCollection

gc  
. Geometries are numbered beginning with 1.

ST\_GeometryN()  
and  
GeometryN()  
are synonyms.

### Example

```
SET @gc = 'GeometryCollection(Point(1 1),LineString(12 14, 9 11))';

SELECT AsText(GeometryN(GeomFromText(@gc),1));
+-----+
| AsText(GeometryN(GeomFromText(@gc),1)) |
+-----+
| POINT(1 1)                                |
+-----+
```

## 1.1.12.9.3.2.16 ST\_GEOMETRYTYPE

### Syntax

```
ST_GeometryType(g)
GeometryType(g)
```

### Description

Returns as a string the name of the geometry type of which the geometry instance

g

is a member. The name corresponds to one of the instantiable Geometry subclasses.

ST\_GeometryType()  
and  
GeometryType()  
are synonyms.

### Examples

```
SELECT GeometryType(GeomFromText('POINT(1 1)'));

+-----+
| GeometryType(GeomFromText('POINT(1 1)')) |
+-----+
| POINT                                         |
+-----+
```

## 1.1.12.9.3.2.17 ST\_ISCLOSED

### Syntax

```
ST_IsClosed(g)
IsClosed(g)
```

### Description

Returns 1 if a given LINESTRING's start and end points are the same, or 0 if they are not the same. Before MariaDB 10.1.5 , returns NULL if not given a LINESTRING. After MariaDB 10.1.5 , returns -1.

ST\_IsClosed()  
and  
IsClosed()  
are synonyms.

### Examples

```

SET @ls = 'LineString(0 0, 0 4, 4 4, 0 0)';
SELECT ST_ISCLOSED(GEOMFROMTEXT(@ls));
+-----+
| ST_ISCLOSED(GEOMFROMTEXT(@ls)) |
+-----+
| 1 |
+-----+

SET @ls = 'LineString(0 0, 0 4, 4 4, 0 1)';
SELECT ST_ISCLOSED(GEOMFROMTEXT(@ls));
+-----+
| ST_ISCLOSED(GEOMFROMTEXT(@ls)) |
+-----+
| 0 |
+-----+

```

## 1.1.12.9.3.2.18 ST\_ISEMPTY

### Syntax

```
ST_IsEmpty(g)
IsEmpty(g)
```

### Description

`IsEmpty` is a function defined by the OpenGIS specification, but is not fully implemented by MariaDB or MySQL.

Since MariaDB and MySQL do not support GIS EMPTY values such as POINT EMPTY, as implemented it simply returns

```

1
if the geometry value
g
is invalid,
0
if it is valid, and
NULL
if the argument is
NULL
.
```

`ST_IsEmpty()`  
and  
`IsEmpty()`  
are synonyms.

## 1.1.12.9.3.2.19 ST\_IsRing

MariaDB starting with [10.1.2](#)

The `ST_IsRing` function was introduced in [MariaDB 10.1.2](#).

### Syntax

```
ST_IsRing(g)
IsRing(g)
```

### Description

Returns true if a given `LINESTRING` is a ring, that is, both `ST_IsClosed` and `ST_IsSimple`. A simple curve does not pass through the same point more than once. However, see [MDEV-7510](#).

`St_IsRing()`  
and  
`IsRing()`  
are synonyms.

## 1.1.12.9.3.2.20 ST\_IsSimple

### Syntax

```
ST_IsSimple(g)
IsSimple(g)
```

### Description

Returns true if the given Geometry has no anomalous geometric points, false if it does, or NULL if given a NULL value.

ST\_IsSimple() and IsSimple() are synonyms.

### Examples

A POINT is always simple.

```
SET @g = 'Point(1 2)';

SELECT ST_ISSIMPLE(GEOMFROMTEXT(@g));
+-----+
| ST_ISSIMPLE(GEOMFROMTEXT(@g)) |
+-----+
|                         1 |
+-----+
```

## 1.1.12.9.3.2.21 ST\_NUMGEOMETRIES

### Syntax

```
ST_NumGeometries(gc)
NumGeometries(gc)
```

### Description

Returns the number of geometries in the GeometryCollection

gc

ST\_NumGeometries()  
and  
NumGeometries()  
are synonyms.

### Example

```
SET @gc = 'GeometryCollection(Point(1 1),LineString(2 2, 3 3))';

SELECT NUMGEOMETRIES(GeomFromText(@gc));
+-----+
| NUMGEOMETRIES(GeomFromText(@gc)) |
+-----+
|                           2 |
+-----+
```

## 1.1.12.9.3.2.22 ST\_RELATE

MariaDB starting with 10.1.2

The ST\_RELATE() function was introduced in MariaDB 10.1.2

## Syntax

```
ST_Relate(g1, g2, i)
```

## Description

Returns true if Geometry

g1  
is spatially related to Geometry  
g2  
by testing for intersections between the interior, boundary and exterior of the two geometries as specified by the values in intersection matrix  
pattern  
i  
.

## 1.1.12.9.3.2.23 ST\_SRID

## Syntax

```
ST_SRID(g)  
SRID(g)
```

## Description

Returns an integer indicating the Spatial Reference System ID for the geometry value g.

In MariaDB, the SRID value is just an integer associated with the geometry value. All calculations are done assuming Euclidean (planar) geometry.

ST\_SRID()  
and  
SRID()  
are synonyms.

## Examples

```
SELECT SRID(GeomFromText('LineString(1 1,2 2)',101));  
+-----+  
| SRID(GeomFromText('LineString(1 1,2 2)',101)) |  
+-----+  
| 101 |  
+-----+
```

## 1.1.12.9.3.3 Geometry Relations

### 1.1.12.9.3.3.1 CONTAINS

## Syntax

```
Contains(g1,g2)
```

## Description

Returns

1  
or  
0  
to indicate whether a geometry  
g1  
completely contains geometry  
g2  
. CONTAINS() is based on the original MySQL implementation and uses object bounding rectangles, while [ST\\_CONTAINS\(\)](#) uses object shapes.

This tests the opposite relationship to [Within\(\)](#) .

## 1.1.12.9.3.3.2 CROSSES

### Syntax

```
Crosses(g1,g2)
```

### Description

Returns

```
1  
if  
g1  
spatially crosses  
g2  
. Returns  
NULL  
if  
g1  
is a Polygon or a MultiPolygon , or if  
g2  
is a Point or a MultiPoint . Otherwise, returns  
0  
. .
```

The term spatially crosses denotes a spatial relation between two given geometries that has the following properties:

- The two geometries intersect
- Their intersection results in a geometry that has a dimension that is one less than the maximum dimension of the two given geometries
- Their intersection is not equal to either of the two given geometries

CROSSES() is based on the original MySQL implementation, and uses object bounding rectangles, while [ST\\_CROSSES\(\)](#) uses object shapes.

## 1.1.12.9.3.3.3 DISJOINT

### Syntax

```
Disjoint(g1,g2)
```

### Description

Returns

```
1  
or  
0  
to indicate whether  
g1  
is spatially disjoint from (does not intersect)  
g2  
. .
```

DISJOINT() tests the opposite relationship to [INTERSECTS\(\)](#) .

DISJOINT() is based on the original MySQL implementation and uses object bounding rectangles, while [ST\\_DISJOINT\(\)](#) uses object shapes.

## 1.1.12.9.3.3.4 EQUALS

### Syntax

```
Equals(g1,g2)
```

From [MariaDB 10.2.3](#) :

```
MBREQUALS(g1,g2)
```

## Description

Returns

1  
or  
0  
to indicate whether  
*g1*  
is spatially equal to  
*g2*

EQUALS() is based on the original MySQL implementation and uses object bounding rectangles, while [ST\\_EQUALS\(\)](#) uses object shapes.

From [MariaDB 10.2.3](#),

MBREQUALS  
is a synonym for  
Equals

## 1.1.12.9.3.3.5 INTERSECTS

### Syntax

```
INTERSECTS(g1,g2)
```

## Description

Returns

1  
or  
0  
to indicate whether geometry  
*g1*  
spatially intersects geometry  
*g2*

INTERSECTS() is based on the original MySQL implementation and uses object bounding rectangles, while [ST\\_INTERSECTS\(\)](#) uses object shapes.

INTERSECTS() tests the opposite relationship to [DISJOINT\(\)](#).

## 1.1.12.9.3.3.6 OVERLAPS

### Syntax

```
OVERLAPS(g1,g2)
```

## Description

Returns

1  
or  
0  
to indicate whether  
*g1*  
spatially overlaps  
*g2*  
. The term spatially overlaps is used if two geometries intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.

OVERLAPS() is based on the original MySQL implementation and uses object bounding rectangles, while [ST\\_OVERLAPS\(\)](#) uses object shapes.

## 1.1.12.9.3.3.7 ST\_CONTAINS

### Syntax

```
ST_CONTAINS(g1,g2)
```

## Description

Returns

```
1  
or  
0  
to indicate whether a geometry  
g1  
completely contains geometry  
g2  
.
```

ST\_CONTAINS() uses object shapes, while [CONTAINS\(\)](#) , based on the original MySQL implementation, uses object bounding rectangles.

ST\_CONTAINS tests the opposite relationship to [ST\\_WITHIN\(\)](#) .

## Examples

```
SET @g1 = ST_GEOFROMTEXT('POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))');  
  
SET @g2 = ST_GEOFROMTEXT('POINT(174 149)');  
  
SELECT ST_CONTAINS(@g1,@g2);  
+-----+  
| ST_CONTAINS(@g1,@g2) |  
+-----+  
| 1 |  
+-----+  
  
SET @g2 = ST_GEOFROMTEXT('POINT(175 151)');  
  
SELECT ST_CONTAINS(@g1,@g2);  
+-----+  
| ST_CONTAINS(@g1,@g2) |  
+-----+  
| 0 |  
+-----+
```

## 1.1.12.9.3.3.8 ST\_CROSSES

### Syntax

```
ST_CROSSES(g1,g2)
```

## Description

Returns

```
1  
if geometry  
g1  
spatially crosses geometry  
g2  
. Returns  
NULL  
if  
g1  
is a Polygon or a MultiPolygon , or if  
g2  
is a Point or a MultiPoint . Otherwise, returns  
0  
.
```

The term spatially crosses denotes a spatial relation between two given geometries that has the following properties:

- The two geometries intersect
- Their intersection results in a geometry that has a dimension that is one less than the maximum dimension of the two given geometries

- Their intersection is not equal to either of the two given geometries

ST\_CROSSES() uses object shapes, while [CROSSES\(\)](#), based on the original MySQL implementation, uses object bounding rectangles.

## Examples

```
SET @g1 = ST_GEOFROMTEXT('LINESTRING(174 149, 176 151)');
SET @g2 = ST_GEOFROMTEXT('POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))');

SELECT ST_CROSSES(@g1,@g2);
+-----+
| ST_CROSSES(@g1,@g2) |
+-----+
| 1 |
+-----+

SET @g1 = ST_GEOFROMTEXT('LINESTRING(176 149, 176 151)');
SELECT ST_CROSSES(@g1,@g2);
+-----+
| ST_CROSSES(@g1,@g2) |
+-----+
| 0 |
+-----+
```

### 1.1.12.9.3.3.9 ST\_DIFFERENCE

#### Syntax

```
ST_DIFFERENCE(g1,g2)
```

#### Description

Returns a geometry representing the point set difference of the given geometry values.

#### Example

```
SET @g1 = POINT(10,10), @g2 = POINT(20,20);
SELECT ST_AsText(ST_Difference(@g1, @g2));
+-----+
| ST_AsText(ST_Difference(@g1, @g2)) |
+-----+
| POINT(10 10) |
+-----+
```

### 1.1.12.9.3.3.10 ST\_DISJOINT

#### Syntax

```
ST_DISJOINT(g1,g2)
```

#### Description

Returns

1  
or  
0  
to indicate whether geometry  
*g1*  
is spatially disjoint from (does not intersect with) geometry  
*g2*  
.

`ST_DISJOINT()` uses object shapes, while `DISJOINT()` , based on the original MySQL implementation, uses object bounding rectangles.  
`ST_DISJOINT()` tests the opposite relationship to `ST_INTERSECTS()` .

## Examples

```
SET @g1 = ST_GEOFROMTEXT('POINT(0 0)');
SET @g2 = ST_GEOFROMTEXT('LINESTRING(2 0, 0 2)');
SELECT ST_DISJOINT(@g1,@g2);
+-----+
| ST_DISJOINT(@g1,@g2) |
+-----+
|          1 |
+-----+

SET @g2 = ST_GEOFROMTEXT('LINESTRING(0 0, 0 2)');
SELECT ST_DISJOINT(@g1,@g2);
+-----+
| ST_DISJOINT(@g1,@g2) |
+-----+
|          0 |
+-----+
```

## 1.1.12.9.3.3.11 ST\_DISTANCE

### Syntax

```
ST_DISTANCE(g1,g2)
```

### Description

Returns the distance between two geometries, or null if not given valid inputs.

### Example

```
SELECT ST_Distance(POINT(1,2),POINT(2,2));
+-----+
| ST_Distance(POINT(1,2),POINT(2,2)) |
+-----+
|          1 |
+-----+
```

## 1.1.12.9.3.3.12 ST\_DISTANCE\_SPHERE

MariaDB starting with 10.2.38

`ST_DISTANCE_SPHERE`  
was introduced in MariaDB 10.2.38 , MariaDB 10.3.29 , MariaDB 10.4.19 and MariaDB 10.5.10 .

### Syntax

```
ST_DISTANCE_SPHERE(g1,g2,[r])
```

### Description

Returns the spherical distance between two geometries (point or multipoint) on a sphere with the optional radius  $r$  (default is the Earth radius if  $r$  is not specified), or NULL if not given valid inputs.

## Example

```
set @zenica = ST_GeomFromText('POINT(17.907743 44.203438)');
set @sarajevo = ST_GeomFromText('POINT(18.413076 43.856258)');
SELECT ST_Distance_Sphere(@zenica, @sarajevo);
55878.59337591705
```

## 1.1.12.9.3.3.13 ST\_EQUALS

### Syntax

```
ST_EQUALS(g1,g2)
```

### Description

Returns

1  
or  
0  
to indicate whether geometry  
*g1*  
is spatially equal to geometry  
*g2*

ST\_EQUALS() uses object shapes, while [EQUALS\(\)](#), based on the original MySQL implementation, uses object bounding rectangles.

### Examples

```
SET @g1 = ST_GEOMFROMTEXT('LINESTRING(174 149, 176 151)');
SET @g2 = ST_GEOMFROMTEXT('LINESTRING(176 151, 174 149)');
SELECT ST_EQUALS(@g1,@g2);
+-----+
| ST_EQUALS(@g1,@g2) |
+-----+
|      1      |
+-----+
```

```
SET @g1 = ST_GEOMFROMTEXT('POINT(0 2)');
SET @g1 = ST_GEOMFROMTEXT('POINT(2 0)');
SELECT ST_EQUALS(@g1,@g2);
+-----+
| ST_EQUALS(@g1,@g2) |
+-----+
|      0      |
+-----+
```

## 1.1.12.9.3.3.14 ST\_INTERSECTS

### Syntax

```
ST_INTERSECTS(g1,g2)
```

### Description

Returns

1  
or  
0

to indicate whether geometry

*g1*

spatially intersects geometry

*g2*

.

ST\_INTERSECTS() uses object shapes, while [INTERSECTS\(\)](#), based on the original MySQL implementation, uses object bounding rectangles.

ST\_INTERSECTS() tests the opposite relationship to [ST\\_DISJOINT\(\)](#).

## Examples

```
SET @g1 = ST_GEOMFROMTEXT('POINT(0 0)');
SET @g2 = ST_GEOMFROMTEXT('LINESTRING(0 0, 0 2)');
SELECT ST_INTERSECTS(@g1,@g2);
+-----+
| ST_INTERSECTS(@g1,@g2) |
+-----+
| 1 |
+-----+
```

```
SET @g2 = ST_GEOMFROMTEXT('LINESTRING(2 0, 0 2)');
SELECT ST_INTERSECTS(@g1,@g2);
+-----+
| ST_INTERSECTS(@g1,@g2) |
+-----+
| 0 |
+-----+
```

## 1.1.12.9.3.3.15 ST\_LENGTH

### Syntax

```
ST_LENGTH(ls)
```

### Description

Returns as a double-precision number the length of the [LineString](#) value

*ls*

in its associated spatial reference.

## Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';
SELECT ST_LENGTH(ST_GeomFromText(@ls));
+-----+
| ST_LENGTH(ST_GeomFromText(@ls)) |
+-----+
| 2.82842712474619 |
+-----+
```

## 1.1.12.9.3.3.16 ST\_OVERLAPS

### Syntax

```
ST_OVERLAPS(g1,g2)
```

### Description

Returns

```
1  
or  
0  
to indicate whether geometry  
g1  
spatially overlaps geometry  
g2  
.
```

The term spatially overlaps is used if two geometries intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.

ST\_OVERLAPS() uses object shapes, while [OVERLAPS\(\)](#), based on the original MySQL implementation, uses object bounding rectangles.

## 1.1.12.9.3.3.17 ST\_TOUCHES

### Syntax

```
ST_TOUCHES(g1,g2)
```

### Description

Returns

```
1  
or  
0  
to indicate whether geometry  
g1  
spatially touches geometry  
g2  
. Two geometries spatially touch if the interiors of the geometries do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.
```

ST\_TOUCHES() uses object shapes, while [TOUCHES\(\)](#), based on the original MySQL implementation, uses object bounding rectangles.

### Examples

```
SET @g1 = ST_GEOFROMTEXT('POINT(2 0)');  
  
SET @g2 = ST_GEOFROMTEXT('LINESTRING(2 0, 0 2)');  
  
SELECT ST_TOUCHES(@g1,@g2);  
+-----+  
| ST_TOUCHES(@g1,@g2) |  
+-----+  
| 1 |  
+-----+  
  
SET @g1 = ST_GEOFROMTEXT('POINT(2 1)');  
  
SELECT ST_TOUCHES(@g1,@g2);  
+-----+  
| ST_TOUCHES(@g1,@g2) |  
+-----+  
| 0 |  
+-----+
```

## 1.1.12.9.3.3.18 ST\_WITHIN

### Syntax

```
ST_WITHIN(g1,g2)
```

### Description

Returns

1  
or  
0  
to indicate whether geometry  
*g1*  
is spatially within geometry  
*g2*

This tests the opposite relationship as

[ST\\_CONTAINS\(\)](#)

ST\_WITHIN() uses object shapes, while [WITHIN\(\)](#), based on the original MySQL implementation, uses object bounding rectangles.

## Examples

```
SET @g1 = ST_GEOFROMTEXT('POINT(174 149)';

SET @g2 = ST_GEOFROMTEXT('POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))');

SELECT ST_WITHIN(@g1,@g2);
+-----+
| ST_WITHIN(@g1,@g2) |
+-----+
|          1          |
+-----+

SET @g1 = ST_GEOFROMTEXT('POINT(176 151)';

SELECT ST_WITHIN(@g1,@g2);
+-----+
| ST_WITHIN(@g1,@g2) |
+-----+
|          0          |
+-----+
```

## 1.1.12.9.3.3.19 TOUCHES

### Syntax

`Touches(g1,g2)`

### Description

Returns

1  
or  
0  
to indicate whether  
*g1*  
spatially touches  
*g2*  
. Two geometries spatially touch if the interiors of the geometries do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.

TOUCHES() is based on the original MySQL implementation and uses object bounding rectangles, while [ST\\_TOUCHES\(\)](#) uses object shapes.

## 1.1.12.9.3.3.20 WITHIN

### Syntax

`Within(g1,g2)`

## Description

Returns

1  
or  
0  
to indicate whether  
g1  
is spatially within  
g2  
. This tests the opposite relationship as

`Contains()`

`WITHIN()` is based on the original MySQL implementation, and uses object bounding rectangles, while `ST_WITHIN()` uses object shapes.

## Examples

```
SET @g1 = GEOMFROMTEXT('POINT(174 149)');
SET @g2 = GEOMFROMTEXT('POINT(176 151)');
SET @g3 = GEOMFROMTEXT('POLYGON((175 150, 20 40, 50 60, 125 100, 175 150))');

SELECT within(@g1,@g3);
+-----+
| within(@g1,@g3) |
+-----+
|           1 |
+-----+

SELECT within(@g2,@g3);
+-----+
| within(@g2,@g3) |
+-----+
|           0 |
+-----+
```

## 1.1.12.9.3.4 LineString Properties

### 1.1.12.9.3.4.1 ENDPOINT

A synonym for `ST_ENDPOINT`.

### 1.1.12.9.3.4.2 GLENGTH

## Syntax

```
GLength(ls)
```

## Description

Returns as a double-precision number the length of the `LineString` value

`ls`  
in its associated spatial reference.

## Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT GLength(GeomFromText(@ls));
+-----+
| GLength(GeomFromText(@ls)) |
+-----+
|           2.82842712474619 |
+-----+
```

## See Also

- [ST\\_LENGTH\(\)](#) is the OpenGIS equivalent.

### 1.1.12.9.3.4.3 NumPoints

A synonym for [ST\\_NumPoints](#).

### 1.1.12.9.3.4.4 PointN

A synonym for [ST\\_PointN](#).

### 1.1.12.9.3.4.5 STARTPOINT

A synonym for [ST\\_STARTPOINT](#).

### 1.1.12.9.3.4.6 ST\_ENDPOINT

## Syntax

```
ST_EndPoint(ls)
EndPoint(ls)
```

## Description

Returns the [Point](#) that is the endpoint of the [LineString](#) value

ls

ST\_EndPoint()  
and  
EndPoint()  
are synonyms.

## Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';
SELECT AsText(EndPoint(GeomFromText(@ls)));
+-----+
| AsText(EndPoint(GeomFromText(@ls))) |
+-----+
| POINT(3 3)                         |
+-----+
```

### 1.1.12.9.3.4.7 ST\_NUMPOINTS

## Syntax

```
ST_NumPoints(ls)
NumPoints(ls)
```

## Description

Returns the number of [Point](#) objects in the [LineString](#) value

ls

ST\_NumPoints()  
and  
NumPoints()  
are synonyms.

## Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT NumPoints(GeomFromText(@ls));
+-----+
| NumPoints(GeomFromText(@ls)) |
+-----+
|          3 |
+-----+
```

## 1.1.12.9.3.4.8 ST\_POINTN

### Syntax

```
ST_PointN(ls,N)
PointN(ls,N)
```

### Description

Returns the N-th [Point](#) in the [LineString](#) value

ls  
. Points are numbered beginning with  
1  
.

ST\_PointN()  
and  
PointN()  
are synonyms.

## Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT AsText(PointN(GeomFromText(@ls),2));
+-----+
| AsText(PointN(GeomFromText(@ls),2)) |
+-----+
| POINT(2 2)                         |
+-----+
```

## 1.1.12.9.3.4.9 ST\_STARTPOINT

### Syntax

```
ST_StartPoint(ls)
StartPoint(ls)
```

### Description

Returns the [Point](#) that is the start point of the [LineString](#) value

ls  
.

ST\_StartPoint()  
and  
StartPoint()  
are synonyms.

## Examples

```
SET @ls = 'LineString(1 1,2 2,3 3)';

SELECT AsText(StartPoint(GeomFromText(@ls)));
+-----+
| AsText(StartPoint(GeomFromText(@ls))) |
+-----+
| POINT(1 1) |
+-----+
```

### 1.1.12.9.3.5 MBR (Minimum Bounding Rectangle)

#### 1.1.12.9.3.5.1 MBR Definition

##### Description

The MBR (Minimum Bounding Rectangle), or Envelope is the bounding geometry, formed by the minimum and maximum (X,Y) coordinates:

##### Examples

```
((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
```

### 1.1.12.9.3.5.2 MBRContains

##### Syntax

```
MBRContains(g1,g2)
```

##### Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangle of g1 contains the Minimum Bounding Rectangle of g2. This tests the opposite relationship as [MBRWithin\(\)](#).

##### Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))';

SET @g2 = GeomFromText('Point(1 1)');

SELECT MBRContains(@g1,@g2), MBRContains(@g2,@g1);
+-----+
| MBRContains(@g1,@g2) | MBRContains(@g2,@g1) |
+-----+
| 1 | 0 |
+-----+
```

### 1.1.12.9.3.5.3 MBDisjoint

##### Syntax

```
MBDisjoint(g1,g2)
```

##### Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries g1 and g2 are disjoint. Two geometries are disjoint if they do not intersect, that is touch or overlap.

## Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((4 4,4 7,7 7,7 4,4 4))');
SELECT mbrdisjoint(@g1,@g2);
+-----+
| mbrdisjoint(@g1,@g2) |
+-----+
|          1          |
+-----+  
  
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))';
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbrdisjoint(@g1,@g2);
+-----+
| mbrdisjoint(@g1,@g2) |
+-----+
|          0          |
+-----+
```

## 1.1.12.9.3.5.4 MBREqual

### Syntax

```
MBREqual(g1,g2)
```

### Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries g1 and g2 are the same.

## Examples

```
SET @g1=GEOMFROMTEXT('LINESTRING(0 0, 1 2)');
SET @g2=GEOMFROMTEXT('POLYGON((0 0, 0 2, 1 2, 1 0, 0 0))';
SELECT MbrEqual(@g1,@g2);
+-----+
| MbrEqual(@g1,@g2) |
+-----+
|          1          |
+-----+  
  
SET @g1=GEOMFROMTEXT('LINESTRING(0 0, 1 3)');
SET @g2=GEOMFROMTEXT('POLYGON((0 0, 0 2, 1 4, 1 0, 0 0))';
SELECT MbrEqual(@g1,@g2);
+-----+
| MbrEqual(@g1,@g2) |
+-----+
|          0          |
+-----+
```

## 1.1.12.9.3.5.5 MBRIntersects

### Syntax

```
MBRIntersects(g1,g2)
```

### Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries g1 and g2 intersect.

## Examples

```

SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbrintersects(@g1,@g2);
+-----+
| mbrintersects(@g1,@g2) |
+-----+
|      1      |
+-----+

SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((4 4,4 7,7 7,7 4,4 4))');
SELECT mbrintersects(@g1,@g2);
+-----+
| mbrintersects(@g1,@g2) |
+-----+
|      0      |
+-----+

```

## 1.1.12.9.3.5.6 MBROverlaps

### Syntax

```
MBROverlaps(g1,g2)
```

### Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries

g1  
and  
g2

overlap. The term spatially overlaps is used if two geometries intersect and their intersection results in a geometry of the same dimension but not equal to either of the given geometries.

### Examples

```

SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))';
SET @g2 = GeomFromText('Polygon((4 4,4 7,7 7,7 4,4 4))');
SELECT mbroverlaps(@g1,@g2);
+-----+
| mbroverlaps(@g1,@g2) |
+-----+
|      0      |
+-----+

SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))';
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))';
SELECT mbroverlaps(@g1,@g2);
+-----+
| mbroverlaps(@g1,@g2) |
+-----+
|      0      |
+-----+

SET @g1 = GeomFromText('Polygon((0 0,0 4,4 4,4 0,0 0))';
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))';
SELECT mbroverlaps(@g1,@g2);
+-----+
| mbroverlaps(@g1,@g2) |
+-----+
|      1      |
+-----+

```

## 1.1.12.9.3.5.7 MBRTouches

### Syntax

```
MBRTouches(g1,g2)
```

## Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangles of the two geometries

g1

and

g2

touch. Two geometries spatially touch if the interiors of the geometries do not intersect, but the boundary of one of the geometries intersects either the boundary or the interior of the other.

## Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((4 4,4 7,7 7,7 4,4 4))');
SELECT mbrtouches(@g1,@g2);
+-----+
| mbrtouches(@g1,@g2) |
+-----+
|          0          |
+-----+


SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbrtouches(@g1,@g2);
+-----+
| mbrtouches(@g1,@g2) |
+-----+
|          1          |
+-----+


SET @g1 = GeomFromText('Polygon((0 0,0 4,4 4,4 0,0 0))');
SET @g2 = GeomFromText('Polygon((3 3,3 6,6 6,6 3,3 3))');
SELECT mbrtouches(@g1,@g2);
+-----+
| mbrtouches(@g1,@g2) |
+-----+
|          0          |
+-----+
```

## 1.1.12.9.3.5.8 MBRWithin

### Syntax

```
MBRWithin(g1,g2)
```

## Description

Returns 1 or 0 to indicate whether the Minimum Bounding Rectangle of g1 is within the Minimum Bounding Rectangle of g2. This tests the opposite relationship as [MBRContains\(\)](#).

## Examples

```
SET @g1 = GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0))');
SET @g2 = GeomFromText('Polygon((0 0,0 5,5 5,5 0,0 0))');
SELECT MBRWithin(@g1,@g2), MBRWithin(@g2,@g1);
+-----+
| MBRWithin(@g1,@g2) | MBRWithin(@g2,@g1) |
+-----+
|          1          |          0          |
+-----+
```

## 1.1.12.9.3.6 Point Properties

## 1.1.12.9.3.6.1 ST\_X

### Syntax

```
ST_X(p)
X(p)
```

### Description

Returns the X-coordinate value for the point

p  
as a double-precision number.

ST\_X()  
and  
X()  
are synonyms.

### Examples

```
SET @pt = 'Point(56.7 53.34)';

SELECT X(GeomFromText(@pt));
+-----+
| X(GeomFromText(@pt)) |
+-----+
|      56.7 |
+-----+
```

## 1.1.12.9.3.6.2 ST\_Y

### Syntax

```
ST_Y(p)
Y(p)
```

### Description

Returns the Y-coordinate value for the point p as a double-precision number.

ST\_Y()  
and  
Y()  
are synonyms.

### Examples

```
SET @pt = 'Point(56.7 53.34)';

SELECT Y(GeomFromText(@pt));
+-----+
| Y(GeomFromText(@pt)) |
+-----+
|      53.34 |
+-----+
```

## 1.1.12.9.3.6.3 X

A synonym for [ST\\_X](#).

## 1.1.12.9.3.6.4 Y

A synonym for [ST\\_Y](#).

## 1.1.12.9.3.7 Polygon Properties

### 1.1.12.9.3.7.1 AREA

A synonym for [ST\\_AREA](#).

### 1.1.12.9.3.7.2 CENTROID

A synonym for [ST\\_CENTROID](#).

### 1.1.12.9.3.7.3 ExteriorRing

A synonym for [ST\\_ExteriorRing](#).

### 1.1.12.9.3.7.4 InteriorRingN

A synonym for [ST\\_InteriorRingN](#).

### 1.1.12.9.3.7.5 NumInteriorRings

A synonym for [ST\\_NumInteriorRings](#).

### 1.1.12.9.3.7.6 ST\_AREA

#### Syntax

```
ST_Area(poly)
Area(poly)
```

#### Description

Returns as a double-precision number the area of the Polygon value

`poly`  
, as measured in its spatial reference system.

`ST_Area()`  
and  
`Area()`  
are synonyms.

#### Examples

```
SET @poly = 'Polygon((0 0,0 3,3 0,0 0),(1 1,1 2,2 1,1 1))';

SELECT Area(GeomFromText(@poly));
+-----+
| Area(GeomFromText(@poly)) |
+-----+
|          4          |
+-----+
```

### 1.1.12.9.3.7.7 ST\_CENTROID

#### Syntax

```
ST_Centroid(mpoly)
Centroid(mpoly)
```

#### Description

Returns a point reflecting the mathematical centroid (geometric center) for the [MultiPolygon](#) *mpoly*. The resulting point will not necessarily be on the MultiPolygon.

`ST_Centroid()`  
and  
`Centroid()`  
are synonyms.

## Examples

```
SET @poly = ST_GeomFromText('POLYGON((0 0,20 0,20 20,0 20,0 0))');
SELECT ST_AsText(ST_Centroid(@poly)) AS center;
+-----+
| center      |
+-----+
| POINT(10 10) |
+-----+
```

## 1.1.12.9.3.7.8 ST\_ExteriorRing

### Syntax

```
ST_ExteriorRing(poly)
ExteriorRing(poly)
```

### Description

Returns the exterior ring of the Polygon value

`poly`  
as a LineString.

`ST_ExteriorRing()`  
and  
`ExteriorRing()`  
are synonyms.

## Examples

```
SET @poly = 'Polygon((0 0,0 3,3 3,3 0,0),(1 1,1 2,2 2,2 1,1 1))';
SELECT AsText(ExteriorRing(GeomFromText(@poly)));
+-----+
| AsText(ExteriorRing(GeomFromText(@poly))) |
+-----+
| LINESTRING(0 0,0 3,3 3,3 0,0 0)           |
+-----+
```

## 1.1.12.9.3.7.9 ST\_InteriorRingN

### Syntax

```
ST_InteriorRingN(poly,N)
InteriorRingN(poly,N)
```

### Description

Returns the N-th interior ring for the Polygon value

`poly`  
as a LineString. Rings are numbered beginning with 1.

`ST_InteriorRingN()`

and  
InteriorRingN()  
are synonyms.

## Examples

```
SET @poly = 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';
SELECT AsText(InteriorRingN(GeomFromText(@poly),1));
+-----+
| AsText(InteriorRingN(GeomFromText(@poly),1)) |
+-----+
| LINESTRING(1 1,1 2,2 2,2 1,1 1)           |
+-----+
```

### 1.1.12.9.3.7.10 ST\_NumInteriorRings

#### Syntax

```
ST_NumInteriorRings(poly)
NumInteriorRings(poly)
```

#### Description

Returns an integer containing the number of interior rings in the Polygon value

poly

Note that according to the OpenGIS standard, a **POLYGON** should have exactly one ExteriorRing and all other rings should lie within that ExteriorRing and thus be the InteriorRings. Practically, however, some systems, including MariaDB's, permit polygons to have several 'ExteriorRings'. In the case of there being multiple, non-overlapping exterior rings **ST\_NumInteriorRings()** will return

1

ST\_NumInteriorRings()  
and  
NumInteriorRings()  
are synonyms.

## Examples

```
SET @poly = 'Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2 2,2 1,1 1))';
SELECT NumInteriorRings(GeomFromText(@poly));
+-----+
| NumInteriorRings(GeomFromText(@poly)) |
+-----+
|                               1 |
```

Non-overlapping 'polygon':

```
SELECT ST_NumInteriorRings(ST_PolyFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),
(-1 -1,-5 -1,-5,-1 -5,-1 -1))')) AS NumInteriorRings;
+-----+
| NumInteriorRings |
+-----+
|               1 |
```

### 1.1.12.9.3.8 WKB

#### 1.1.12.9.3.8.1 Well-Known Binary (WKB) Format

WKB stands for Well-Known Binary, a format for representing geographical and geometrical data.

WKB uses 1-byte unsigned integers, 4-byte unsigned integers, and 8-byte double-precision numbers.

- The first byte indicates the byte order. 00 for big endian, or 01 for little endian.
- The next 4 bytes indicate the geometry type. Values from 1 to 7 indicate whether the type is Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, or GeometryCollection respectively.
- The 8-byte floats represent the co-ordinates.

Take the following example, a sequence of 21 bytes each represented by two hex digits:

```
00000000140000000000000000401000000000000
```

- It's big endian
  - 00 **00000001** 4000000000000000401000000000000
- It's a POINT
  - 00 **00000001** 4000000000000000401000000000000
- The X co-ordinate is 2.0
  - 0000000001 **4000000000000000** 4010000000000000
- The Y-co-ordinate is 4.0
  - 0000000000140000000000000000 **4010000000000000**

## 1.1.12.9.3.8.2 AsBinary

A synonym for [ST\\_AsBinary\(\)](#).

## 1.1.12.9.3.8.3 AsWKB

A synonym for [ST\\_AsBinary\(\)](#).

## 1.1.12.9.3.8.4 MLineFromWKB

### Syntax

```
MLineFromWKB(wkb[,srid])
MultiLineStringFromWKB(wkb[,srid])
```

### Description

Constructs a **MULTILINESTRING** value using its [WKB](#) representation and [SRID](#).

`MLineFromWKB()`  
and  
`MultiLineStringFromWKB()`  
are synonyms.

### Examples

```
SET @g = ST_AsBinary(MLineFromText('MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48))'));

SELECT ST_AsText(MLineFromWKB(@g));
+-----+
| ST_AsText(MLineFromWKB(@g))           |
+-----+
| MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48)) |
+-----+
```

## 1.1.12.9.3.8.5 MPointFromWKB

### Syntax

```
MPointFromWKB(wkb[,srid])
MultiPointFromWKB(wkb[,srid])
```

### Description

Constructs a **MULTIPOINT** value using its [WKB](#) representation and [SRID](#) .

`MPointFromWKB()`  
and  
`MultiPointFromWKB()`  
are synonyms.

## Examples

```
SET @g = ST_AsBinary(MPointFromText('MultiPoint( 1 1, 2 2, 5 3, 7 2, 9 3, 8 4, 6 6, 6 9, 4 9, 1 5 )'));  
  
SELECT ST_AsText(MPointFromWKB(@g));  
+-----+  
| ST_AsText(MPointFromWKB(@g)) |  
+-----+  
| MULTIPOLYGON(1 1,2 2,5 3,7 2,9 3,8 4,6 6,6 9,4 9,1 5) |  
+-----+
```

## 1.1.12.9.3.8.6 MPolyFromWKB

### Syntax

```
MPolyFromWKB(wkb[,srid])  
MultiPolygonFromWKB(wkb[,srid])
```

### Description

Constructs a **MULTIPOLYGON** value using its [WKB](#) representation and [SRID](#) .

`MPolyFromWKB()`  
and  
`MultiPolygonFromWKB()`  
are synonyms.

## Examples

```
SET @g = ST_AsBinary(MPolyFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67  
0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67 13,59 13,59 18))) |  
+-----+  
| ST_AsText(MPolyFromWKB(@g)) |  
+-----+  
| MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67 13,59 13,59 18))) |  
+-----+
```

## 1.1.12.9.3.8.7 GeomCollFromWKB

A synonym for [ST\\_GeomCollFromWKB](#) .

## 1.1.12.9.3.8.8 GeometryCollectionFromWKB

A synonym for [ST\\_GeomCollFromWKB](#) .

## 1.1.12.9.3.8.9 GeometryFromWKB

A synonym for [ST\\_GeomFromWKB](#) .

## 1.1.12.9.3.8.10 GeomFromWKB

A synonym for [ST\\_GeomFromWKB](#) .

## 1.1.12.9.3.8.11 LineFromWKB

A synonym for [ST\\_LineFromWKB](#) .

## 1.1.12.9.3.8.12 LineStringFromWKB

A synonym for [ST\\_LineFromWKB](#).

## 1.1.12.9.3.8.13 MultiLineStringFromWKB

A synonym for [MLineFromWKB\(\)](#).

## 1.1.12.9.3.8.14 MultiPointFromWKB

A synonym for [MPointFromWKB](#).

## 1.1.12.9.3.8.15 MultiPolygonFromWKB

Synonym for [MPolyFromWKB](#).

## 1.1.12.9.3.8.16 PointFromWKB

A synonym for [ST\\_PointFromWKB](#).

## 1.1.12.9.3.8.17 PolyFromWKB

A synonym for [ST\\_PolyFromWKB](#).

## 1.1.12.9.3.8.18 PolygonFromWKB

A synonym for [ST\\_PolyFromWKB](#).

## 1.1.12.9.3.8.19 ST\_AsBinary

### Syntax

```
ST_AsBinary(g)
AsBinary(g)
ST_AsWKB(g)
AsWKB(g)
```

### Description

Converts a value in internal geometry format to its [WKB](#) representation and returns the binary result.

`ST_AsBinary()`  
,

`AsBinary()`  
,

`ST_AsWKB()`  
and  
`AsWKB()`  
are synonyms,

### Examples

```
SET @poly = ST_GeomFromText('POLYGON((0 0,0 1,1 1,1 0,0 0))';
SELECT ST_AsBinary(@poly);

SELECT ST_AsText(ST_GeomFromWKB(ST_AsWKB(@poly)));
+-----+
| ST_AsText(ST_GeomFromWKB(ST_AsWKB(@poly))) |
+-----+
| POLYGON((0 0,0 1,1 1,1 0,0 0))               |
+-----+
```

## 1.1.12.9.3.8.20 ST\_AsWKB

A synonym for [ST\\_AsBinary\(\)](#).

## 1.1.12.9.3.8.21 ST\_GeomCollFromWKB

### Syntax

```
ST_GeomCollFromWKB(wkb[,srid])
ST_GeometryCollectionFromWKB(wkb[,srid])
GeomCollFromWKB(wkb[,srid])
GeometryCollectionFromWKB(wkb[,srid])
```

### Description

Constructs a GEOMETRYCOLLECTION value using its [WKB](#) representation and SRID.

ST\_GeomCollFromWKB()  
,  
ST\_GeometryCollectionFromWKB()  
,  
GeomCollFromWKB()  
and  
GeometryCollectionFromWKB()  
are synonyms.

### Examples

```
SET @g = ST_AsBinary(ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((5 5,10 5,10 10,5 5)),POINT(10 10))');

SELECT ST_AsText(ST_GeomCollFromWKB(@g));
+-----+
| ST_AsText(ST_GeomCollFromWKB(@g))           |
+-----+
| GEOMETRYCOLLECTION(POLYGON((5 5,10 5,10 10,5 5)),POINT(10 10)) |
+-----+
```

## 1.1.12.9.3.8.22 ST\_GeometryCollectionFromWKB

A synonym for [ST\\_GeomCollFromWKB](#).

## 1.1.12.9.3.8.23 ST\_GeometryFromWKB

A synonym for [ST\\_GeomFromWKB](#).

## 1.1.12.9.3.8.24 ST\_GeomFromWKB

### Syntax

```
ST_GeomFromWKB(wkb[,srid])
ST_GeometryFromWKB(wkb[,srid])
GeomFromWKB(wkb[,srid])
GeometryFromWKB(wkb[,srid])
```

### Description

Constructs a geometry value of any type using its [WKB](#) representation and SRID.

ST\_GeomFromWKB()  
,  
ST\_GeometryFromWKB()  
,  
GeomFromWKB()  
and  
GeometryFromWKB()  
are synonyms.

## Examples

```
SET @g = ST_AsBinary(ST_LineFromText('LINESTRING(0 4, 4 6)'));

SELECT ST_AsText(ST_GeomFromWKB(@g));
+-----+
| ST_AsText(ST_GeomFromWKB(@g)) |
+-----+
| LINESTRING(0 4,4 6)           |
+-----+
```

## 1.1.12.9.3.8.25 ST\_LineFromWKB

### Syntax

```
ST_LineFromWKB(wkb[,srid])
LineFromWKB(wkb[,srid])
ST_LineStringFromWKB(wkb[,srid])
LineStringFromWKB(wkb[,srid])
```

### Description

Constructs a LINESTRING value using its [WKB](#) representation and SRID.

`ST_LineFromWKB()`  
,  
`LineFromWKB()`  
,  
`ST_LineStringFromWKB()`  
, and  
`LineStringFromWKB()`  
are synonyms.

## Examples

```
SET @g = ST_AsBinary(ST_LineFromText('LineString(0 4,4 6)'));

SELECT ST_AsText(ST_LineFromWKB(@g)) AS l;
+-----+
| l          |
+-----+
| LINESTRING(0 4,4 6) |
+-----+
```

## 1.1.12.9.3.8.26 ST\_LineStringFromWKB

A synonym for [ST\\_LineFromWKB](#).

## 1.1.12.9.3.8.27 ST\_PointFromWKB

### Syntax

```
ST_PointFromWKB(wkb[,srid])
PointFromWKB(wkb[,srid])
```

### Description

Constructs a [POINT](#) value using its [WKB](#) representation and [SRID](#).

`ST_PointFromWKB()`  
and  
`PointFromWKB()`

are synonyms.

## Examples

```
SET @g = ST_AsBinary(ST_PointFromText('POINT(0 4)'));  
  
SELECT ST_AsText(ST_PointFromWKB(@g)) AS p;  
+-----+  
| p |  
+-----+  
| POINT(0 4) |  
+-----+
```

## 1.1.12.9.3.8.28 ST\_PolyFromWKB

### Syntax

```
ST_PolyFromWKB(wkb[,srid])  
ST_PolygonFromWKB(wkb[,srid])  
PolyFromWKB(wkb[,srid])  
PolygonFromWKB(wkb[,srid])
```

### Description

Constructs a **POLYGON** value using its **WKB** representation and **SRID**.

ST\_PolyFromWKB()  
,  
ST\_PolygonFromWKB()  
,  
PolyFromWKB()  
and  
PolygonFromWKB()  
are synonyms.

## Examples

```
SET @g = ST_AsBinary(ST_PolyFromText('POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1))'));  
  
SELECT ST_AsText(ST_PolyFromWKB(@g)) AS p;  
+-----+  
| p |  
+-----+  
| POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1)) |  
+-----+
```

## 1.1.12.9.3.8.29 ST\_PolyFromWKB

### Syntax

```
ST_PolyFromWKB(wkb[,srid])  
ST_PolygonFromWKB(wkb[,srid])  
PolyFromWKB(wkb[,srid])  
PolygonFromWKB(wkb[,srid])
```

### Description

Constructs a **POLYGON** value using its **WKB** representation and **SRID**.

ST\_PolyFromWKB()  
,  
ST\_PolygonFromWKB()

,  
PolyFromWKB()  
and  
PolygonFromWKB()  
are synonyms.

## Examples

```
SET @g = ST_AsBinary(ST_PolyFromText('POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1))');

SELECT ST_AsText(ST_PolyFromWKB(@g)) AS p;
+-----+
| p           |
+-----+
| POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1)) |
+-----+
```

### 1.1.12.9.3.9 WKT

#### 1.1.12.9.3.9.1 WKT Definition

#### Description

The Well-Known Text (WKT) representation of Geometry is designed to exchange geometry data in ASCII form. Examples of the basic geometry types include:

Geometry Types
POINT
LINESTRING
POLYGON
MULTIPOINT
MULTILINESTRING
MULTIPOLYGON
GEOMETRYCOLLECTION
GEOMETRY

#### See Also

- [Geometry Types](#)

#### 1.1.12.9.3.9.2 AsText

A synonym for [ST\\_AsText\(\)](#).

#### 1.1.12.9.3.9.3 AsWKT

A synonym for [ST\\_AsText\(\)](#).

#### 1.1.12.9.3.9.4 GeomCollFromText

A synonym for [ST\\_GeomCollFromText](#).

#### 1.1.12.9.3.9.5 GeometryCollectionFromText

A synonym for [ST\\_GeomCollFromText](#).

#### 1.1.12.9.3.9.6 GeometryFromText

A synonym for [ST\\_GeomFromText](#).

#### 1.1.12.9.3.9.7 GeomFromText

A synonym for [ST\\_GeomFromText](#).

## 1.1.12.9.3.9.8 LineFromText

A synonym for [ST\\_LineFromText](#) .

## 1.1.12.9.3.9.9 LineStringFromText

A synonym for [ST\\_LineFromText](#) .

## 1.1.12.9.3.9.10 MLineFromText

### Syntax

```
MLineFromText(wkt[,srid])
MultiLineStringFromText(wkt[,srid])
```

### Description

Constructs a [MULTILINESTRING](#) value using its [WKT](#) representation and [SRID](#) .

`MLineFromText()`  
and  
`MultiLineStringFromText()`  
are synonyms.

### Examples

```
CREATE TABLE gis_multi_line (g MULTILINESTRING);
SHOW FIELDS FROM gis_multi_line;
INSERT INTO gis_multi_line VALUES
  (MultiLineStringFromText('MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48))')),
   (MLineFromText('MULTILINESTRING((10 48,10 21,10 0))')),
   (MLineFromWKB(AsWKB(MultiLineString(LineString(Point(1, 2), Point(3, 5)), LineString(Point(2, 5), Point(5, 8), Point(21, 7))))))
```

## 1.1.12.9.3.9.11 MPointFromText

### Syntax

```
MPointFromText(wkt[,srid])
MultiPointFromText(wkt[,srid])
```

### Description

Constructs a [MULTIPOINT](#) value using its [WKT](#) representation and [SRID](#) .

`MPointFromText()`  
and  
`MultiPointFromText()`  
are synonyms.

### Examples

```
CREATE TABLE gis_multi_point (g MULTIPOINT);
SHOW FIELDS FROM gis_multi_point;
INSERT INTO gis_multi_point VALUES
  (MultiPointFromText('MULTIPOINT(0 0,10 10,10 20,20 20)')),
   (MPointFromText('MULTIPOINT(1 1,11 11,11 21,21 21)')),
   (MPointFromWKB(AsWKB(MultiPoint(Point(3, 6), Point(4, 10))))));
```

## 1.1.12.9.3.9.12 MPolyFromText

## Syntax

```
MPolyFromText(wkt[,srid])
MultiPolygonFromText(wkt[,srid])
```

## Description

Constructs a **MULTIPOINT** value using its [WKT](#) representation and [SRID](#).

`MPolyFromText()`  
and  
`MultiPolygonFromText()`  
are synonyms.

## Examples

```
CREATE TABLE gis_multi_polygon  (g MULTIPOINT);
SHOW FIELDS FROM gis_multi_polygon;
INSERT INTO gis_multi_polygon VALUES
(MultiPolygonFromText('MULTIPOINT(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67 13,59 13,
(MPolyFromText('MULTIPOINT(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67 13,59 13,59 18))
(MPolyFromWKB(AsWKB(MultiPolygon(Polygon(LineString(Point(0, 3), Point(3, 3), Point(3, 0), Point(0, 3)))))));
```

## 1.1.12.9.3.9.13 MultiLineStringFromText

A synonym for [MLineFromText](#).

## 1.1.12.9.3.9.14 MultiPointFromText

A synonym for [MPointFromText](#).

## 1.1.12.9.3.9.15 MultiPolygonFromText

A synonym for [MPolyFromText](#).

## 1.1.12.9.3.9.16 PointFromText

A synonym for [ST\\_PointFromText](#).

## 1.1.12.9.3.9.17 PolyFromText

A synonym for [ST\\_PolyFromText](#).

## 1.1.12.9.3.9.18 PolygonFromText

A synonym for [ST\\_PolyFromText](#).

## 1.1.12.9.3.9.19 ST\_AsText

## Syntax

```
ST_AsText(g)
AsText(g)
ST_AsWKT(g)
AsWKT(g)
```

## Description

Converts a value in internal geometry format to its [WKT](#) representation and returns the string result.

`ST_AsText()`  
,

`AsText()`  
,

`ST_AsWKT()`  
and  
`AsWKT()`  
are all synonyms.

## Examples

```
SET @g = 'LineString(1 1,4 4,6 6)';

SELECT ST_AsText(ST_GeomFromText(@g));
+-----+
| ST_AsText(ST_GeomFromText(@g)) |
+-----+
| LINESTRING(1 1,4 4,6 6)      |
+-----+
```

## 1.1.12.9.3.9.20 `ST_ASWKT`

A synonym for [ST\\_ASTEXT\(\)](#).

## 1.1.12.9.3.9.21 `ST_GeomCollFromText`

### Syntax

```
ST_GeomCollFromText(wkt[,srid])
ST_GeometryCollectionFromText(wkt[,srid])
GeomCollFromText(wkt[,srid])
GeometryCollectionFromText(wkt[,srid])
```

### Description

Constructs a `GEOMETRYCOLLECTION` value using its [WKT](#) representation and [SRID](#).

`ST_GeomCollFromText()`  
,  
`ST_GeometryCollectionFromText()`  
,  
`GeomCollFromText()`  
and  
`GeometryCollectionFromText()`  
are all synonyms.

### Example

```
CREATE TABLE gis_geometrycollection  (g GEOMETRYCOLLECTION);
SHOW FIELDS FROM gis_geometrycollection;
INSERT INTO gis_geometrycollection VALUES
(GeomCollFromText('GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(0 0,10 10)))'),
(GeometryFromWKB(AsWKB(GeometryCollection(Point(44, 6), LineString(Point(3, 6), Point(7, 9))))),
(GeomFromText('GeometryCollection()')),
(GeomFromText('GeometryCollection EMPTY')));
```

## 1.1.12.9.3.9.22 `ST_GeometryCollectionFromText`

A synonym for [ST\\_GeomCollFromText](#).

## 1.1.12.9.3.9.23 `ST_GeometryFromText`

A synonym for [ST\\_GeomFromText](#).

## 1.1.12.9.3.9.24 `ST_GeomFromText`

### Syntax

```
ST_GeomFromText(wkt[,srid])
ST_GeometryFromText(wkt[,srid])
GeomFromText(wkt[,srid])
GeometryFromText(wkt[,srid])
```

## Description

Constructs a geometry value of any type using its [WKT](#) representation and [SRID](#).

```
GeomFromText()
,
GeometryFromText()
,
ST_GeomFromText()
and
ST_GeometryFromText()
are all synonyms.
```

## Example

```
SET @g = ST_GEOMFROMTEXT('POLYGON((1 1,1 5,4 9,6 9,9 3,7 2,1 1));
```

## 1.1.12.9.3.9.25 ST\_LineFromText

### Syntax

```
ST_LineFromText(wkt[,srid])
ST_LineStringFromText(wkt[,srid])
LineFromText(wkt[,srid])
LineStringFromText(wkt[,srid])
```

## Description

Constructs a [LINESTRING](#) value using its [WKT](#) representation and [SRID](#).

```
ST_LineFromText()
,
ST_LineStringFromText()
,
ST_LineFromText()
and
ST_LineStringFromText()
are all synonyms.
```

## Examples

```
CREATE TABLE gis_line (g LINESTRING);
SHOW FIELDS FROM gis_line;
INSERT INTO gis_line VALUES
(LineFromText('LINESTRING(0 0,0 10,10 0)'),
(LineStringFromText('LINESTRING(10 10,20 10,20 20,10 20,10 10'))),
(LineStringFromWKB(AsWKB(LineString(Point(10, 10), Point(40, 10)))));
```

## 1.1.12.9.3.9.26 ST\_LineStringFromText

A synonym for [ST\\_LineFromText](#).

## 1.1.12.9.3.9.27 ST\_PointFromText

### Syntax

```
ST_PointFromText(wkt[,srid])
PointFromText(wkt[,srid])
```

## Description

Constructs a **POINT** value using its **WKT** representation and **SRID**.

ST\_PointFromText()  
and  
PointFromText()  
are synonyms.

## Examples

```
CREATE TABLE gis_point  (g POINT);
SHOW FIELDS FROM gis_point;
INSERT INTO gis_point VALUES
  (PointFromText('POINT(10 10'))),
  (PointFromText('POINT(20 10'))),
  (PointFromText('POINT(20 20'))),
  (PointFromWKB(PointFromText('POINT(10 20'))));
```

## 1.1.12.9.3.9.28 ST\_PolyFromText

### Syntax

```
ST_PolyFromText(wkt[,srid])
ST_PolygonFromText(wkt[,srid])
PolyFromText(wkt[,srid])
PolygonFromText(wkt[,srid])
```

## Description

Constructs a **POLYGON** value using its **WKT** representation and **SRID**.

ST\_PolyFromText()  
,  
ST\_PolygonFromText()  
,  
PolyFromText()  
and  
ST\_PolygonFromText()  
are all synonyms.

## Examples

```
CREATE TABLE gis_polygon  (g POLYGON);
INSERT INTO gis_polygon VALUES
  (PolygonFromText('POLYGON((10 10,20 10,20 20,10 20,10 10))')),
  (PolyFromText('POLYGON((0 0,50 0,50 50,0 50,0 0), (10 10,20 10,20 20,10 20,10 10))'));
```

## 1.1.12.9.3.9.29 ST\_PolygonFromText

A synonym for **ST\_PolyFromText**.

## 1.1.12.9.4 JSON Functions

### 1.1.12.9.4.1 Differences between JSON\_QUERY and JSON\_VALUE

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3.

The primary difference between the two functions is that `JSON_QUERY` returns an object or an array, while `JSON_VALUE` returns a scalar.

Take the following JSON document as an example

```
SET @json='{"x": [0,1], "y": "[0,1]", "z": "Monty"}';
```

Note that data member "x" is an array, and data members "y" and "z" are strings. The following examples demonstrate the differences between the two functions.

## 1.1.12.9.4.2 JSONPath Expressions

## Contents

1. JSON Path Syntax
    1. Object Member Selector
    2. Array Element Selector
    3. Wildcard
  2. Compatibility

A number of [JSON functions](#) accept JSON Path expressions. MariaDB defines this path as follows:

# JSON Path Syntax

path : ['lax'] '\$' [step]\*

The path starts with an optional *path mode*. At the moment, MariaDB supports only the "lax" mode, which is also the mode that is used when it is not explicitly specified.

The

4

symbol represents the context item. The search always starts from the context item; because of that, the path always starts with \$

Then, it is followed by zero or more steps, which select element(s) in the JSON document. A step may be one of the following:

- Object member selector
- Array element selector
- Wildcard selector

## Object Member Selector

To select member(s) in a JSON object, one can use one of the following:

- `.memberName`  
  selects the value of the member with name `memberName`.
- `."memberName"`  
  - the same as above but allows one to select a member with a name that's not a valid identifier (that is, has space, dot, and/or other characters)
- `.*`  
  - selects the values of all members of the object.

If the current item is an array (instead of an object), nothing will be selected.

## Array Element Selector

To select elements of an array, one can use one of the following:

- `[N]`  
  selects element number N in the array. The elements are counted from zero.
- `[*]`  
  selects all elements in the array.

If the current item is an object (instead of an array), nothing will be selected.

Starting from MariaDB server 10.9, JSON path also supports negative index in array and 'last' keyword for accessing array elements. Negative index starts from -1.

- `[-N]`  
  selects n th element from end.
- `[last-N]`  
  selects n th element from the last element.

Example:

```
SET @json='{
    "A": [0,
          [1, 2, 3],
          [4, 5, 6],
          "seven",
          0.8,
          true,
          false,
          "eleven",
          [12, [13, 14], {"key1":"value1"},[15]],
          true],
    "B": {"C": 1},
    "D": 2
}';

SELECT JSON_EXTRACT(@json, '$.A[-8][1]');

+-----+
| JSON_EXTRACT(@json, '$.A[-8][1]') |
+-----+
| 5                                     |
+-----+


SELECT JSON_EXTRACT(@json, '$.A[last-7][1]');

+-----+
| SELECT JSON_EXTRACT(@json, '$.A[last-7][1]'); |
+-----+
| 5                                     |
+-----+
```

This will produce output for first index of eighth from last element of a two dimensional array.

## Wildcard

The wildcard step,

\*\*

, recursively selects all child elements of the current element. Both array elements and object members are selected.

The wildcard step must not be the last step in the JSONPath expression. It must be followed by an array or object member selector step.

For example:

```
select json_extract(@json_doc, '$**.price');
```

will select all object members in the document that are named

    price  
    , while

```
select json_extract(@json_doc, '$**[2]');
```

will select the second element in each of the arrays present in the document.

## Compatibility

MariaDB's JSONPath syntax supports a subset of JSON Path's definition in the SQL Standard. The most notable things not supported are the

    strict  
    mode and filters.

MariaDB's JSONPath is close to MySQL's JSONPath. The wildcard step (

\*\*

) is a non-standard extension that has the same meaning as in MySQL. The differences between MariaDB and MySQL's JSONPath are: MySQL supports

    [last]  
    and  
    [M to N]  
    as array element selectors; MySQL doesn't allow one to specify the mode explicitly (but uses  
    lax  
    mode implicitly).

### 1.1.12.9.4.3 JSON\_ARRAY

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_ARRAY([value[, value2] ...])
```

## Description

Returns a JSON array containing the listed values. The list can be empty.

## Example

```
SELECT Json_Array(56, 3.1416, 'My name is "Foo"', NULL);
+-----+
| Json_Array(56, 3.1416, 'My name is "Foo"', NULL) |
+-----+
| [56, 3.1416, "My name is \"Foo\"", null]         |
+-----+
```

## See also

- [JSON\\_MAKE\\_ARRAY](#), the CONNECT storage engine function

## 1.1.12.9.4.4 JSON\_ARRAYAGG

## 1.1.12.9.4.5 JSON\_ARRAY\_APPEND

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

### Syntax

```
JSON_ARRAY_APPEND(json_doc, path, value[, path, value] ...)
```

### Description

Appends values to the end of the specified arrays within a JSON document, returning the result, or NULL if any of the arguments are NULL.

Evaluation is performed from left to right, with the resulting document from the previous pair becoming the new value against which the next pair is evaluated.

If the

```
json_doc  
is not a valid JSON document, or if any of the paths are not valid, or contain a  
*  
or  
**  
wildcard, an error is returned.
```

### Examples

```
SET @json = '[1, 2, [3, 4]]';  
  
SELECT JSON_ARRAY_APPEND(@json, '$[0]', 5)  
+-----+  
| JSON_ARRAY_APPEND(@json, '$[0]', 5) |  
+-----+  
| [[1, 5], 2, [3, 4]] |  
+-----+  
  
SELECT JSON_ARRAY_APPEND(@json, '$[1]', 6);  
+-----+  
| JSON_ARRAY_APPEND(@json, '$[1]', 6) |  
+-----+  
| [1, [2, 6], [3, 4]] |  
+-----+  
  
SELECT JSON_ARRAY_APPEND(@json, '$[1]', 6, '$[2]', 7);  
+-----+  
| JSON_ARRAY_APPEND(@json, '$[1]', 6, '$[2]', 7) |  
+-----+  
| [1, [2, 6], [3, 4, 7]] |  
+-----+  
  
SELECT JSON_ARRAY_APPEND(@json, '$', 5);  
+-----+  
| JSON_ARRAY_APPEND(@json, '$', 5) |  
+-----+  
| [1, 2, [3, 4], 5] |  
+-----+  
  
SET @json = '{"A": 1, "B": [2], "C": [3, 4]}';  
  
SELECT JSON_ARRAY_APPEND(@json, '$.B', 5);  
+-----+  
| JSON_ARRAY_APPEND(@json, '$.B', 5) |  
+-----+  
| [{"A": 1, "B": [2, 5], "C": [3, 4]}] |  
+-----+
```

## 1.1.12.9.4.6 JSON\_ARRAY\_INSERT

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

### Syntax

```
JSON_ARRAY_INSERT(json_doc, path, value[, path, value] ...)
```

### Description

Inserts a value into a JSON document, returning the modified document, or NULL if any of the arguments are NULL.

Evaluation is performed from left to right, with the resulting document from the previous pair becoming the new value against which the next pair is evaluated.

If the

```
json_doc  
is not a valid JSON document, or if any of the paths are not valid, or contain a  
*  
or  
**  
wildcard, an error is returned.
```

### Examples

```
SET @json = '[1, 2, [3, 4]]';  
  
SELECT JSON_ARRAY_INSERT(@json, '$[0]', 5);  
+-----+  
| JSON_ARRAY_INSERT(@json, '$[0]', 5) |  
+-----+  
| [5, 1, 2, [3, 4]] |  
+-----+  
  
SELECT JSON_ARRAY_INSERT(@json, '$[1]', 6);  
+-----+  
| JSON_ARRAY_INSERT(@json, '$[1]', 6) |  
+-----+  
| [1, 6, 2, [3, 4]] |  
+-----+  
  
SELECT JSON_ARRAY_INSERT(@json, '$[1]', 6, '$[2]', 7);  
+-----+  
| JSON_ARRAY_INSERT(@json, '$[1]', 6, '$[2]', 7) |  
+-----+  
| [1, 6, 7, 2, [3, 4]] |  
+-----+
```

## 1.1.12.9.4.7 JSON\_COMPACT

MariaDB starting with 10.2.4

This function was added in MariaDB 10.2.4 .

### Syntax

```
JSON_COMPACT(json_doc)
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [See Also](#)

## Description

Removes all unnecessary spaces so the json document is as short as possible.

## Example

```
SET @j = '{ "A": 1, "B": [2, 3]}';  
  
SELECT JSON_COMPACT(@j), @j;  
+-----+-----+  
| JSON_COMPACT(@j) | @j  
+-----+-----+  
| {"A":1,"B":[2,3]} | { "A": 1, "B": [2, 3]} |  
+-----+-----+
```

## See Also

- [JSON video tutorial covering JSON\\_COMPACT](#).

## 1.1.12.9.4.8 JSON\_CONTAINS

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_CONTAINS(json_doc, val[, path])
```

## Description

Returns whether or not the specified value is found in the given JSON document or, optionally, at the specified path within the document. Returns

1  
if it does,  
0  
if not and NULL if any of the arguments are null. An error occurs if the document or path is not valid, or contains the  
\*  
or  
\*\*  
wildcards.

## Examples

```

SET @json = '{"A": 0, "B": {"C": 1}, "D": 2}';

SELECT JSON_CONTAINS(@json, '2', '$.A');
+-----+
| JSON_CONTAINS(@json, '2', '$.A') |
+-----+
|          0          |
+-----+

SELECT JSON_CONTAINS(@json, '2', '$.D');
+-----+
| JSON_CONTAINS(@json, '2', '$.D') |
+-----+
|          1          |
+-----+

SELECT JSON_CONTAINS(@json, '{"C": 1}', '$.A');
+-----+
| JSON_CONTAINS(@json, '{"C": 1}', '$.A') |
+-----+
|          0          |
+-----+

SELECT JSON_CONTAINS(@json, '{"C": 1}', '$.B');
+-----+
| JSON_CONTAINS(@json, '{"C": 1}', '$.B') |
+-----+
|          1          |
+-----+

```

## 1.1.12.9.4.9 JSON\_CONTAINS\_PATH

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

### Syntax

```
JSON_CONTAINS_PATH(json_doc, return_arg, path[, path] ...)
```

### Description

Indicates whether the given JSON document contains data at the specified path or paths. Returns

- 1  
if it does,
- 0  
if not and NULL if any of the arguments are null.

The *return\_arg* can be

- one
- or
- all
- :
- - one
    - Returns
  - 1  
if at least one path exists within the JSON document.
- - all
    - Returns
  - 1  
only if all paths exist within the JSON document.

### Examples

```

SET @json = '{"A": 1, "B": [2], "C": [3, 4]}';

SELECT JSON_CONTAINS_PATH(@json, 'one', '$.A', '$.D');
+-----+
| JSON_CONTAINS_PATH(@json, 'one', '$.A', '$.D') |
+-----+
|          1          |
+-----+
1 row in set (0.00 sec)

SELECT JSON_CONTAINS_PATH(@json, 'all', '$.A', '$.D');
+-----+
| JSON_CONTAINS_PATH(@json, 'all', '$.A', '$.D') |
+-----+
|          0          |
+-----+

```

## 1.1.12.9.4.10 JSON\_DEPTH

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

### Syntax

```
JSON_DEPTH(json_doc)
```

### Description

Returns the maximum depth of the given JSON document, or NULL if the argument is null. An error will occur if the argument is an invalid JSON document.

- Scalar values or empty arrays or objects have a depth of 1.
- Arrays or objects that are not empty but contain only elements or member values of depth 1 will have a depth of 2.
- In other cases, the depth will be greater than 2.

### Examples

```

SELECT JSON_DEPTH('[]'), JSON_DEPTH('true'), JSON_DEPTH('{}');
+-----+-----+-----+
| JSON_DEPTH('[]') | JSON_DEPTH('true') | JSON_DEPTH('{}') |
+-----+-----+-----+
|      1      |         1        |       1       |
+-----+-----+-----+

SELECT JSON_DEPTH('[1, 2, 3]'), JSON_DEPTH('[], {}, []');
+-----+-----+
| JSON_DEPTH('[1, 2, 3]') | JSON_DEPTH('[], {}, []') |
+-----+-----+
|      2      |         2        |
+-----+-----+

SELECT JSON_DEPTH('[1, 2, [3, 4, 5, 6], 7]');
+-----+
| JSON_DEPTH('[1, 2, [3, 4, 5, 6], 7]') |
+-----+
|      3      |
+-----+

```

## 1.1.12.9.4.11 JSON\_DETAILED

MariaDB starting with [10.2.4](#)

This function was added in [MariaDB 10.2.4](#).

### Syntax

```
JSON_DETAILED(json_doc[, tab_size])
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [See Also](#)

## Description

Represents JSON in the most understandable way emphasizing nested structures.

## Example

```
SET @j = '{ "A":1,"B":[2,3]}';

SELECT @j;
+-----+
| @j      |
+-----+
| { "A":1,"B":[2,3]} |
+-----+

SELECT JSON_DETAILED(@j);
+-----+
| JSON_DETAILED(@j)          |
+-----+
| {                           |
|   "A": 1,                   |
|   "B": [                     |
|     2,                      |
|     3                       |
|   ]                         |
| } |                         |
+-----+
```

## See Also

- [JSON video tutorial covering JSON\\_DETAILED.](#)

## 1.1.12.9.4.12 JSON\_EQUALS

MariaDB starting with [10.7.0](#)

JSON\_EQUALS was added in [MariaDB 10.7.0](#).

## Syntax

```
JSON_EQUALS(json1, json2)
```

## Description

Checks if there is equality between two json objects. Returns

1  
if it there is,  
0  
if not, or NULL if any of the arguments are null.

## Examples

```

SELECT JSON_EQUALS('{"a": [1, 2, 3], "b": [4]}', '{"b": [4], "a": [1, 2, 3.0]}');
+-----+
| JSON_EQUALS('{"a": [1, 2, 3], "b": [4]}', '{"b": [4], "a": [1, 2, 3.0]}') |
+-----+
|                               1 |
+-----+

SELECT JSON_EQUALS('{"a": [1, 2, 3]}', '{"a": [1, 2, 3.01]}');
+-----+
| JSON_EQUALS('{"a": [1, 2, 3]}', '{"a": [1, 2, 3.01]}') |
+-----+
|                               0 |
+-----+

```

## See Also

- [JSON\\_NORMALIZE](#)

## 1.1.12.9.4.13 JSON\_EXISTS

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

### Description

Determines whether a specified JSON value exists in the given data. Returns

```

1
if found,
0
if not, or
NULL
if any of the inputs were NULL.

```

## Examples

```

SELECT JSON_EXISTS('{"key1": "xxxx", "key2": [1, 2, 3]}', "$.key2");
+-----+
| JSON_EXISTS('{"key1": "xxxx", "key2": [1, 2, 3]}', "$.key2") |
+-----+
|                               1 |
+-----+

SELECT JSON_EXISTS('{"key1": "xxxx", "key2": [1, 2, 3]}', "$.key3");
+-----+
| JSON_EXISTS('{"key1": "xxxx", "key2": [1, 2, 3]}', "$.key3") |
+-----+
|                               0 |
+-----+

SELECT JSON_EXISTS('{"key1": "xxxx", "key2": [1, 2, 3]}', "$.key2[1]");
+-----+
| JSON_EXISTS('{"key1": "xxxx", "key2": [1, 2, 3]}', "$.key2[1]") |
+-----+
|                               1 |
+-----+

SELECT JSON_EXISTS('{"key1": "xxxx", "key2": [1, 2, 3]}', "$.key2[10]");
+-----+
| JSON_EXISTS('{"key1": "xxxx", "key2": [1, 2, 3]}', "$.key2[10]") |
+-----+
|                               0 |
+-----+

```

## 1.1.12.9.4.14 JSON\_EXTRACT

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Syntax

```
JSON_EXTRACT(json_doc, path[, path] ...)
```

## Description

Extracts data from a JSON document. The extracted data is selected from the parts matching the path arguments. Returns all matched values; either as a single matched value, or, if the arguments could return multiple values, a result autowrapped as an array in the matching order.

Returns NULL if no paths match or if any of the arguments are NULL.

An error will occur if any path argument is not a valid path, or if the json\_doc argument is not a valid JSON document.

The path expression be a [JSONPath expression](#) as supported by MariaDB

## Examples

```
SET @json = '[1, 2, [3, 4]]';

SELECT JSON_EXTRACT(@json, '$[1]');
+-----+
| JSON_EXTRACT(@json, '$[1]') |
+-----+
| 2 |
+-----+

SELECT JSON_EXTRACT(@json, '$[2]');
+-----+
| JSON_EXTRACT(@json, '$[2]') |
+-----+
| [3, 4] |
+-----+

SELECT JSON_EXTRACT(@json, '$[2][1]');
+-----+
| JSON_EXTRACT(@json, '$[2][1]') |
+-----+
| 4 |
+-----+
```

## See Also

- [JSON video tutorial covering JSON\\_EXTRACT.](#)

## 1.1.12.9.4.15 JSON\_INSERT

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

## Syntax

```
JSON_INSERT(json_doc, path, val[, path, val] ...)
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Inserts data into a JSON document, returning the resulting document or NULL if any argument is null.

An error will occur if the JSON document is not invalid, or if any of the paths are invalid or contain a

\*  
or  
\*\*  
wildcard.

`JSON_INSERT` can only insert data while `JSON_REPLACE` can only update. `JSON_SET` can update or insert data.

## Examples

```
SET @json = '{ "A": 0, "B": [1, 2]}';  
  
SELECT JSON_INSERT(@json, '$.C', '[3, 4]');  
+-----+  
| JSON_INSERT(@json, '$.C', '[3, 4]') |  
+-----+  
| { "A": 0, "B": [1, 2], "C": "[3, 4]" } |  
+-----+
```

## See Also

- [JSON video tutorial](#) covering `JSON_INSERT`.

## 1.1.12.9.4.16 JSON\_KEYS

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_KEYS(json_doc[, path])
```

## Description

Returns the keys as a JSON array from the top-level value of a JSON object or, if the optional path argument is provided, the top-level keys from the path.

Excludes keys from nested sub-objects in the top level value. The resulting array will be empty if the selected object is empty.

Returns NULL if any of the arguments are null, a given path does not locate an object, or if the json\_doc argument is not an object.

An error will occur if JSON document is invalid, the path is invalid or if the path contains a

\*  
or  
\*\*  
wildcard.

## Examples

```
SELECT JSON_KEYS('{"A": 1, "B": {"C": 2}}');
+-----+
| JSON_KEYS('{"A": 1, "B": {"C": 2}}') |
+-----+
| ["A", "B"] |
+-----+

SELECT JSON_KEYS('{"A": 1, "B": 2, "C": {"D": 3}}', '$.C');
+-----+
| JSON_KEYS('{"A": 1, "B": 2, "C": {"D": 3}}', '$.C') |
+-----+
| ["D"] |
+-----+
```

## 1.1.12.9.4.17 JSON\_LENGTH

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

### Syntax

```
JSON_LENGTH(json_doc[, path])
```

### Description

Returns the length of a JSON document, or, if the optional path argument is given, the length of the value within the document specified by the path.

Returns NULL if any of the arguments argument are null or the path argument does not identify a value in the document.

An error will occur if the JSON document is invalid, the path is invalid or if the path contains a

- \*
- or
- \*\*
- wildcard.

Length will be determined as follow:

- A scalar's length is always 1.
- If an array, the number of elements in the array.
- If an object, the number of members in the object.

The length of nested arrays or objects are not counted.

### Examples

## 1.1.12.9.4.18 JSON\_LOOSE

MariaDB starting with 10.2.4

This function was added in MariaDB 10.2.4 .

### Syntax

```
JSON_LOOSE(json_doc)
```

### Description

Adds spaces to a JSON document to make it look more readable.

### Example

```
SET @j = '{ "A":1,"B":[2,3]}';

SELECT JSON_LOOSE(@j), @j;
+-----+-----+
| JSON_LOOSE(@j) | @j           |
+-----+-----+
| {"A": 1, "B": [2, 3]} | { "A":1,"B":[2,3]} |
+-----+-----+
```

## 1.1.12.9.4.19 JSON\_MERGE

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

### Syntax

```
JSON_MERGE(json_doc, json_doc[, json_doc] ...)
```

### Description

Merges the given JSON documents.

Returns the merged result, or NULL if any argument is NULL.

An error occurs if any of the arguments are not valid JSON documents.

JSON\_MERGE

has been deprecated since MariaDB 10.2.25 , MariaDB 10.3.16 and MariaDB 10.4.5 . JSON\_MERGE\_PATCH is an RFC 7396-compliant replacement, and JSON\_MERGE\_PRESERVE is a synonym.

### Example

```
SET @json1 = '[1, 2]';
SET @json2 = '[3, 4]';

SELECT JSON_MERGE(@json1,@json2);
+-----+
| JSON_MERGE(@json1,@json2) |
+-----+
| [1, 2, 3, 4]             |
+-----+
```

### See Also

- [JSON\\_MERGE\\_PATCH](#)
- [JSON\\_MERGE\\_PRESERVE](#)

## 1.1.12.9.4.20 JSON\_MERGE\_PATCH

MariaDB starting with 10.2.25

JSON\_MERGE\_PATCH  
was introduced in MariaDB 10.2.25 , MariaDB 10.3.16 and MariaDB 10.4.5 .

### Syntax

```
JSON_MERGE_PATCH(json_doc, json_doc[, json_doc] ...)
```

### Description

Merges the given JSON documents, returning the merged result, or NULL if any argument is NULL.

`JSON_MERGE_PATCH`  
is an RFC 7396-compliant replacement for [JSON\\_MERGE](#), which has been deprecated.

## Example

```
SET @json1 = '[1, 2]';  
SET @json2 = '[2, 3]';  
SELECT JSON_MERGE_PATCH(@json1,@json2),JSON_MERGE_PRESERVE(@json1,@json2);  
+-----+  
| JSON_MERGE_PATCH(@json1,@json2) | JSON_MERGE_PRESERVE(@json1,@json2) |  
+-----+  
| [2, 3] | [1, 2, 2, 3] |  
+-----+
```

## 1.1.12.9.4.21 JSON\_MERGE\_PRESERVE

MariaDB starting with [10.2.25](#)

`JSON_MERGE_PRESERVE`  
was introduced in [MariaDB 10.2.25](#), [MariaDB 10.3.16](#) and [MariaDB 10.4.5](#).

## Syntax

```
JSON_MERGE_PRESERVE(json_doc, json_doc[, json_doc] ...)
```

## Description

Merges the given JSON documents, returning the merged result, or NULL if any argument is NULL.

`JSON_MERGE_PRESERVE`  
was introduced in [MariaDB 10.2.25](#), [MariaDB 10.3.16](#) and [MariaDB 10.4.5](#) as a synonym for [JSON\\_MERGE](#), which has been deprecated.

## Example

```
SET @json1 = '[1, 2]';  
SET @json2 = '[2, 3]';  
SELECT JSON_MERGE_PATCH(@json1,@json2),JSON_MERGE_PRESERVE(@json1,@json2);  
+-----+  
| JSON_MERGE_PATCH(@json1,@json2) | JSON_MERGE_PRESERVE(@json1,@json2) |  
+-----+  
| [2, 3] | [1, 2, 2, 3] |  
+-----+
```

## See Also

- [JSON\\_MERGE\\_PATCH](#)

## 1.1.12.9.4.22 JSON\_NORMALIZE

MariaDB starting with [10.7.0](#)

`JSON_NORMALIZE` was added in [MariaDB 10.7.0](#).

## Syntax

```
JSON_NORMALIZE(json1, json2)
```

## Description

Recursively sorts keys and removes spaces, allowing comparison of json documents for equality.

## Examples

We may wish our application to use the database to enforce a unique constraint on the JSON contents, and we can do so using the `JSON_NORMALIZE` function in combination with a unique key.

For example, if we have a table with a JSON column:

```
CREATE TABLE t1 (
    id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
    val JSON,
    /* other columns here */
    PRIMARY KEY (id)
);
```

Add a unique constraint using `JSON_NORMALIZE` like this:

```
ALTER TABLE t1
    ADD COLUMN jnorm JSON AS (JSON_NORMALIZE(val)) VIRTUAL,
    ADD UNIQUE KEY (jnorm);
```

We can test this by first inserting a row as normal:

```
INSERT INTO t1 (val) VALUES ('{"name": "alice", "color": "blue"}');
```

And then seeing what happens with a different string which would produce the same JSON object:

```
INSERT INTO t1 (val) VALUES ('{ "color": "blue", "name": "alice" }');
ERROR 1062 (23000): Duplicate entry '{"color":"blue","name":"alice"}' for key 'jnorm'
```

## See Also

- [JSON\\_EQUALS](#)

## 1.1.12.9.4.23 JSON\_OBJECT

MariaDB starting with 10.2.3

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_OBJECT([key, value[, key, value] ...])
```

## Description

Returns a JSON object containing the given key/value pairs. The key/value list can be empty.

An error will occur if there are an odd number of arguments, or any key name is NULL.

## Example

```
SELECT JSON_OBJECT("id", 1, "name", "Monty");
+-----+
| JSON_OBJECT("id", 1, "name", "Monty") |
+-----+
| {"id": 1, "name": "Monty"}           |
+-----+
```

## See also

- [JSON\\_MAKE\\_OBJECT](#), the CONNECT storage engine function

## 1.1.12.9.4.24 JSON\_OBJECTAGG

## 1.1.12.9.4.25 JSON\_OVERLAPS

MariaDB starting with 10.9

JSON\_OVERLAPS is added in MariaDB starting from 10.9.

### Syntax

```
JSON_OVERLAPS(json_doc1, json_doc2)
```

### Description

JSON\_OVERLAPS() compares two json documents and returns true if they have at least one common key-value pair between two objects, array element common between two arrays, or array element common with scalar if one of the arguments is a scalar and other is an array. If two json documents are scalars, it returns true if they have same type and value.

If none of the above conditions are satisfied then it returns false.

### Examples

```
SELECT JSON_OVERLAPS('false', 'false');
+-----+
| JSON_OVERLAPS('false', 'false') |
+-----+
| 1 |
+-----+

SELECT JSON_OVERLAPS('true', '[{"abc": 1, "2": true, "false": false}]');
+-----+
| JSON_OVERLAPS('true', '[{"abc": 1, "2": true, "false": false}]') |
+-----+
| 1 |
+-----+

SELECT JSON_OVERLAPS('{"A": 1, "B": {"C": 2}}', '{"A": 2, "B": {"C": 2}}') AS is_overlap;
+-----+
| is_overlap |
+-----+
| 1 |
+-----+
```

Partial match is considered as no-match.

### Examples

```
SELECT JSON_OVERLAPS('[1, 2, true, false, null]', '[3, 4, [1]]) AS is_overlap;
+-----+
| is_overlap |
+-----+
| 0 |
+-----+
```

## 1.1.12.9.4.26 JSON\_QUERY

MariaDB starting with 10.2.3

JSON functions were added in [MariaDB 10.2.3](#).

### Syntax

```
JSON_QUERY(json_doc, path)
```

## Description

Given a JSON document, returns an object or array specified by the path. Returns NULL if not given a valid JSON document, or if there is no match.

## Examples

```
select json_query('{"key1":{"a":1, "b":[1,2]}}', '$.key1');
+-----+
| json_query('{"key1":{"a":1, "b":[1,2]}}', '$.key1') |
+-----+
| {"a":1, "b":[1,2]} |
+-----+

select json_query('{"key1":123, "key1": [1,2,3]}', '$.key1');
+-----+
| json_query('{"key1":123, "key1": [1,2,3]}', '$.key1') |
+-----+
| [1,2,3] |
+-----+
```

## 1.1.12.9.4.27 JSON\_QUOTE

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

## Syntax

```
JSON_QUOTE(json_value)
```

## Description

Quotes a string as a JSON value, usually for producing valid JSON string literals for inclusion in JSON documents. Wraps the string with double quote characters and escapes interior quotes and other special characters, returning a utf8mb4 string.

Returns NULL if the argument is NULL.

## Examples

```
SELECT JSON_QUOTE('A'), JSON_QUOTE("B"), JSON_QUOTE('"C"');
+-----+-----+-----+
| JSON_QUOTE('A') | JSON_QUOTE("B") | JSON_QUOTE('"C"') |
+-----+-----+-----+
| "A"           | "B"          | "\"C\""      |
+-----+-----+-----+
```

## 1.1.12.9.4.28 JSON\_REMOVE

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

## Syntax

```
JSON_REMOVE(json_doc, path[, path] ...)
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

# Description

Removes data from a JSON document returning the result, or NULL if any of the arguments are null. If the element does not exist in the document, no changes are made.

An error will occur if JSON document is invalid, the path is invalid or if the path contains a

- \*
- or
- \*\*
- wildcard.

Path arguments are evaluated from left to right, with the result from the earlier evaluation being used as the value for the next.

## Examples

```
SELECT JSON_REMOVE('{"A": 1, "B": 2, "C": {"D": 3}}', '$.C');
+-----+
| JSON_REMOVE('{"A": 1, "B": 2, "C": {"D": 3}}', '$.C') |
+-----+
| {"A": 1, "B": 2} |
+-----+  
  
SELECT JSON_REMOVE('["A", "B", ["C", "D"], "E"]', '$[1]');
+-----+
| JSON_REMOVE('["A", "B", ["C", "D"], "E"]', '$[1]') |
+-----+
| ["A", ["C", "D"], "E"] |
+-----+
```

## See Also

- [JSON video tutorial](#) covering JSON\_REMOVE.

## 1.1.12.9.4.29 JSON\_REPLACE

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_REPLACE(json_doc, path, val[, path, val] ...)
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Replaces existing values in a JSON document, returning the result, or NULL if any of the arguments are NULL.

An error will occur if the JSON document is invalid, the path is invalid or if the path contains a

- \*
- or
- \*\*
- wildcard.

Paths and values are evaluated from left to right, with the result from the earlier evaluation being used as the value for the next.

JSON\_REPLACE can only update data, while [JSON\\_INSERT](#) can only insert. [JSON\\_SET](#) can update or insert data.

## Examples

```
SELECT JSON_REPLACE('{ "A": 1, "B": [2, 3]}', '$.B[1]', 4);
+-----+
| JSON_REPLACE('{ "A": 1, "B": [2, 3]}', '$.B[1]', 4) |
+-----+
| { "A": 1, "B": [2, 4]} |
+-----+
```

## See Also

- [JSON video tutorial covering JSON\\_REPLACE.](#)

## 1.1.12.9.4.30 JSON\_SEARCH

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_SEARCH(json_doc, return_arg, search_str[, escape_char[, path] ...])
```

## Description

Returns the path to the given string within a JSON document, or NULL if any of `json_doc`, `search_str` or a path argument is NULL; if the search string is not found, or if no path exists within the document.

A warning will occur if the JSON document is not valid, any of the path arguments are not valid, if `return_arg` is neither `one` nor `all`, or if the escape character is not a constant. NULL will be returned.

`return_arg` can be one of two values:

- - 'one': Terminates after finding the first match, so will return one path string. If there is more than one match, it is undefined which is considered first.
  - 'all': Returns all matching path strings, without duplicates. Multiple strings are autowrapped as an array. The order is undefined.

## Examples

```
SET @json = '[{"A": [{"B": "1"}], {"C":"AB"}, {"D":"BC"}]';

SELECT JSON_SEARCH(@json, 'one', 'AB');
+-----+
| JSON_SEARCH(@json, 'one', 'AB') |
+-----+
| "$[2].C" |
+-----+
```

## 1.1.12.9.4.31 JSON\_SET

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_SET(json_doc, path, val[, path, val] ...)
```

## Description

Updates or inserts data into a JSON document, returning the result, or NULL if any of the arguments are NULL or the optional path fails to find an object.

An error will occur if the JSON document is invalid, the path is invalid or if the path contains a \* or **wildcard**.

JSON\_SET can update or insert data, while **JSON\_REPLACE** can only update, and **JSON\_INSERT** only insert.

## Examples

```
SELECT JSON_SET(Priv, '$.locked', 'true') FROM mysql.global_priv
```

## 1.1.12.9.4.32 JSON\_TABLE

MariaDB starting with **10.6.0**

JSON\_TABLE was added in **MariaDB 10.6.0**.

JSON\_TABLE is a table function that converts JSON data into a relational form.

## Syntax

```
JSON_TABLE(json_doc,
           context_path COLUMNS (column_list)
) [AS] alias
```

```
column_list:
    column[, column][, ...]
```

```
column:
    name FOR ORDINALITY
    | name type PATH path_str [on_empty] [on_error]
    | name type EXISTS PATH path_str
    | NESTED PATH path_str COLUMNS (column_list)
```

```
on_empty:
    {NULL | DEFAULT string | ERROR} ON EMPTY
```

```
on_error:
    {NULL | DEFAULT string | ERROR} ON ERROR
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [Column Definitions](#)
    1. [Path Columns](#)
    2. [ORDINALITY Columns](#)
    3. [EXISTS PATH Columns](#)
    4. [NESTED PATHs](#)
  2. [ON EMPTY and ON ERROR Clauses](#)
  3. [Replication](#)
  3. [See Also](#)

## Description

JSON\_TABLE can be used in contexts where a table reference can be used; in the FROM clause of a **SELECT** statement, and in multi-table **UPDATE** / **DELETE** statements.

json\_doc

is the JSON document to extract data from. In the simplest case, it is a string literal containing JSON. In more complex cases it can be an arbitrary expression returning JSON. The expression may have references to columns of other tables. However, one can only refer to tables that precede this JSON\_TABLE invocation. For RIGHT JOIN, it is assumed that its outer side precedes the inner. All tables in outer selects are also considered preceding.

context\_path

is a [JSON Path](#) expression pointing to a collection of nodes in json\_doc

that will be used as the source of rows.

The

COLUMNS

clause declares the names and types of the columns that JSON\_TABLE returns, as well as how the values of the columns are produced.

## Column Definitions

The following types of columns are supported:

### Path Columns

```
name type PATH path_str [on_empty] [on_error]
```

Locates the JSON node pointed to by

path\_str

and returns its value. The path\_str is evaluated using the current row source node as the context node.

```
set @json='
[
  {"name":"Laptop", "color":"black", "price":"1000"},
  {"name":"Jeans", "color":"blue"}
]';

select * from json_table(@json, '$[*]')
columns(
  name varchar(10) path '$.name',
  color varchar(10) path '$.color',
  price decimal(8,2) path '$.price' )
) as jt;
+-----+-----+
| name | color | price |
+-----+-----+
| Laptop | black | 1000.00 |
| Jeans | blue |     NULL |
+-----+-----+
```

The

on\_empty

and

on\_error

clauses specify the actions to be performed when the value was not found or there was an error condition. See the ON EMPTY and ON ERROR clauses section for details.

### ORDINALITY Columns

```
name FOR ORDINALITY
```

Counts the rows, starting from 1.

Example:

```
set @json='
[
  {"name":"Laptop", "color":"black"},
  {"name":"Jeans", "color":"blue"}
]';

select * from json_table(@json, '$[*]')
columns(
  id for ordinality,
  name varchar(10) path '$.name')
) as jt;
+-----+
| id   | name   |
+-----+
| 1    | Laptop |
| 2    | Jeans  |
+-----+
```

### EXISTS PATH Columns

```
name type EXISTS PATH path_str
```

Checks whether the node pointed to by

value\_path  
exists. The  
value\_path  
is evaluated using the current row source node as the context node.

```
set @json='
[
  {"name":"Laptop", "color":"black", "price":1000},
  {"name":"Jeans", "color":"blue"}
]';

select * from json_table(@json, '$[*]')
  columns(
    name varchar(10) path '$.name',
    has_price integer exists path '$.price')
) as jt;
+-----+
| name | has_price |
+-----+
| Laptop |      1 |
| Jeans  |      0 |
+-----+
```

## NESTED PATHs

NESTED PATH converts nested JSON structures into multiple rows.

```
NESTED PATH path COLUMNS (column_list)
```

It finds the sequence of JSON nodes pointed to by

path  
and uses it to produce rows. For each found node, a row is generated with column values as specified by the NESTED PATH's COLUMNS clause. If  
path  
finds no nodes, only one row is generated with all columns having NULL values.

For example, consider a JSON document that contains an array of items, and each item, in turn, is expected to have an array of its available sizes:

```
set @json='
[
  {"name":"Jeans", "sizes": [32, 34, 36]},
  {"name":"T-Shirt", "sizes":["Medium", "Large"]},
  {"name":"Cellphone"}
]';
```

NESTED PATH allows one to produce a separate row for each size each item has:

```
select * from json_table(@json, '$[*]')
  columns(
    name varchar(10) path '$.name',
    nested path '$.sizes[*]' columns (
      size varchar(32) path '$'
    )
  )
) as jt;
+-----+
| name | size |
+-----+
| Jeans | 32  |
| Jeans | 34  |
| Jeans | 36  |
| T-Shirt | Medium |
| T-Shirt | Large |
| Cellphone | NULL |
+-----+
```

NESTED PATH clauses can be nested within one another. They can also be located next to each other. In that case, the nested path clauses will produce records one at a time. The ones that are not producing records will have all columns set to NULL.

Example:

```

set @json='
[{"name":"Jeans", "sizes": [32, 34, 36], "colors":["black", "blue"]}
]';

select * from json_table(@json, '$[*]')
columns(
    name varchar(10) path '$.name',
    nested path '$.sizes[*]' columns (
        size varchar(32) path '$'
    ),
    nested path '$.colors[*]' columns (
        color varchar(32) path '$'
    )
)
) as jt;

+-----+-----+-----+
| name | size | color |
+-----+-----+-----+
| Jeans | 32 | NULL |
| Jeans | 34 | NULL |
| Jeans | 36 | NULL |
| Jeans | NULL | black |
| Jeans | NULL | blue |
+-----+-----+-----+

```

## ON EMPTY and ON ERROR Clauses

The ON EMPTY clause specifies what will be done when the element specified by the search path is missing in the JSON document.

```

on_empty:
{NULL | DEFAULT string | ERROR} ON EMPTY

```

When

```

ON EMPTY
clause is not present,
NULL ON EMPTY
is implied.

```

```

on_error:
{NULL | DEFAULT string | ERROR} ON ERROR

```

The ON ERROR clause specifies what should be done if a JSON structure error occurs when trying to extract the value pointed to by the path expression. A JSON structure error here occurs only when one attempts to convert a JSON non-scalar (array or object) into a scalar value. When the

```

ON ERROR
clause is not present,
NULL ON ERROR
is implied.

```

**Note :** A datatype conversion error (e.g. attempt to store a non-integer value into an `integer` field, or a `varchar` column being truncated) is not considered a JSON error and so will not trigger the

```

ON ERROR
behavior. It will produce warnings, in the same way as CAST(value AS datatype) would.

```

## Replication

In the current code, evaluation of JSON\_TABLE is deterministic, that is, for a given input string JSON\_TABLE will always produce the same set of rows in the same order. However, one can think of JSON documents that one can consider identical which will produce different output. In order to be future-proof and withstand changes like:

- sorting JSON object members by name (like MySQL does)
- changing the way duplicate object members are handled the function is marked as `unsafe for statement-based replication`.

## See Also

- [JSON Support \(video\)](#)

## 1.1.12.9.4.33 JSON\_TYPE

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

## Syntax

```
JSON_TYPE(json_val)
```

## Description

Returns the type of a JSON value (as a string), or NULL if the argument is null.

An error will occur if the argument is an invalid JSON value.

The following is a complete list of the possible return types:

Return type	Value	Example
ARRAY	JSON array	[1, 2, {"key": "value"}]
OBJECT	JSON object	{"key": "value"}
BOOLEAN	JSON true/false literals	true, false
DOUBLE	A number with at least one floating point decimal.	1.2
INTEGER	A number without a floating point decimal.	1
NULL	JSON null literal (this is returned as a string, not to be confused with the SQL NULL value!)	null
STRING	JSON String	"a sample string"

## Examples

```
SELECT JSON_TYPE('{"A": 1, "B": 2, "C": 3}');
+-----+
| JSON_TYPE('{"A": 1, "B": 2, "C": 3}') |
+-----+
| OBJECT |
+-----+
```

## 1.1.12.9.4.34 JSON\_UNQUOTE

MariaDB starting with 10.2.3

JSON functions were added in MariaDB 10.2.3 .

## Syntax

```
JSON_UNQUOTE(val)
```

## Description

Unquotes a JSON value, returning a string, or NULL if the argument is null.

An error will occur if the given value begins and ends with double quotes and is an invalid JSON string literal.

If the given value is not a JSON string, value is passed through unmodified.

Certain character sequences have special meanings within a string. Usually, a backslash is ignored, but the escape sequences in the table below are recognised by MariaDB, unless the [SQL Mode](#) is set to NO\_BACKSLASH\_ESCAPES SQL.

Escape sequence	Character
\"	Double quote ("")

\b	Backslash
\f	Formfeed
\n	Newline (linefeed)
\r	Carriage return
\t	Tab
\\	Backslash ()
\uXXXX	UTF-8 bytes for Unicode value XXXX

## Examples

```
SELECT JSON_UNQUOTE('"Monty"');
+-----+
| JSON_UNQUOTE('"Monty") |
+-----+
| Monty |
+-----+
```

With the default [SQL Mode](#) :

```
SELECT JSON_UNQUOTE('Si\bng\ting');
+-----+
| JSON_UNQUOTE('Si\bng\ting') |
+-----+
| Sng ing |
+-----+
```

Setting NO\_BACKSLASH\_ESCAPES:

```
SET @@sql_mode = 'NO_BACKSLASH_ESCAPES';

SELECT JSON_UNQUOTE('Si\bng\ting');
+-----+
| JSON_UNQUOTE('Si\bng\ting') |
+-----+
| Si\bng\ting |
+-----+
```

## 1.1.12.9.4.35 JSON\_VALID

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_VALID(value)
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

Indicates whether the given value is a valid JSON document or not. Returns

```
1  
if valid,  
0  
if not, and NULL if the argument is NULL.
```

From [MariaDB 10.4.3](#), the JSON\_VALID function is automatically used as a [CHECK constraint](#) for the [JSON data type alias](#) in order to ensure that a valid json document is inserted.

## Examples

```
SELECT JSON_VALID('{"id": 1, "name": "Monty"}');  
+-----+  
| JSON_VALID('{"id": 1, "name": "Monty"}') |  
+-----+  
| 1 |  
+-----+  
  
SELECT JSON_VALID('{"id": 1, "name": "Monty", "oddfield"}');  
+-----+  
| JSON_VALID('{"id": 1, "name": "Monty", "oddfield"}') |  
+-----+  
| 0 |  
+-----+
```

## See Also

- [JSON video tutorial covering JSON\\_VALID.](#)

## 1.1.12.9.4.36 JSON\_VALUE

MariaDB starting with [10.2.3](#)

JSON functions were added in [MariaDB 10.2.3](#).

## Syntax

```
JSON_VALUE(json_doc, path)
```

## Description

Given a JSON document, returns the scalar specified by the path. Returns NULL if not given a valid JSON document, or if there is no match.

## Examples

```

select json_value('{"key1":123}', '$.key1');
+-----+
| json_value('{"key1":123}', '$.key1') |
+-----+
| 123 |
+-----+

select json_value('{"key1": [1,2,3], "key1":123}', '$.key1');
+-----+
| json_value('{"key1": [1,2,3], "key1":123}', '$.key1') |
+-----+
| 123 |
+-----+

```

## 1.1.12.9.5 Spider Functions

### 1.1.12.9.5.1 SPIDER\_BG\_DIRECT\_SQL

#### Syntax

```
SPIDER_BG_DIRECT_SQL('sql', 'tmp_table_list', 'parameters')
```

#### Description

Executes the given SQL statement in the background on the remote server, as defined in the parameters listing. If the query returns a result-set, it stores the results in the given temporary table. When the given SQL statement executes successfully, this function returns the number of called UDF's. It returns

- 0  
when the given SQL statement fails.

This function is a [UDF](#) installed with the [Spider](#) storage engine.

#### Examples

```

SELECT SPIDER_BG_DIRECT_SQL('SELECT * FROM example_table', '',
    'srv "node1", port "8607"' ) AS "Direct Query";
+-----+
| Direct Query |
+-----+
|          1 |
+-----+

```

#### Parameters

`error_rw_mode`

- **Description:** Returns empty results on network error.
  - 0 : Return error on getting network error.
  - 1 : Return 0 records on getting network error.
- **Default Table Value:**
  - 0
- **DSN Parameter Name:**
  - erwm

#### See also

- [SPIDER\\_DIRECT\\_SQL](#)

## 1.1.12.9.5.2 SPIDER\_COPY\_TABLES

### Syntax

```
SPIDER_COPY_TABLES(spider_table_name,  
    source_link_id, destination_link_id_list [,parameters])
```

### Description

A [UDF](#) installed with the [Spider Storage Engine](#), this function copies table data from

```
source_link_id  
to  
destination_link_id_list  
. The service does not need to be stopped in order to copy.
```

If the Spider table is partitioned, the name must be of the format

```
table_name#P#partition_name  
. The partition name can be viewed in the  
mysql.spider_tables  
table, for example:
```

```
SELECT table_name FROM mysql.spider_tables;  
+-----+  
| table_name |  
+-----+  
| spt_a#P#pt1 |  
| spt_a#P#pt2 |  
| spt_a#P#pt3 |  
+-----+
```

#### Returns

```
1  
if the data was copied successfully, or  
0  
if copying the data failed.
```

## 1.1.12.9.5.3 SPIDER\_DIRECT\_SQL

### Syntax

```
SPIDER_DIRECT_SQL('sql', 'tmp_table_list', 'parameters')
```

### Description

A [UDF](#) installed with the [Spider Storage Engine](#), this function is used to execute the SQL string

```
sql  
on the remote server, as defined in  
parameters  
. If any resultsets are returned, they are stored in the  
tmp_table_list  
.
```

The function returns

```
1  
if the SQL executes successfully, or  
0  
if it fails.
```

### Examples

```
SELECT SPIDER_DIRECT_SQL('SELECT * FROM s', '', 'srv "node1", port "8607"');
+-----+
| SPIDER_DIRECT_SQL('SELECT * FROM s', '', 'srv "node1", port "8607"') |
+-----+
|                               1 |
+-----+
```

## See also

- [SPIDER\\_BG\\_DIRECT\\_SQL](#)

## 1.1.12.9.5.4 SPIDER\_FLUSH\_TABLE\_MON\_CACHE

### Syntax

```
SPIDER_FLUSH_TABLE_MON_CACHE()
```

### Description

A UDF installed with the [Spider Storage Engine](#), this function is used for refreshing monitoring server information. It returns a value of

1

.

### Examples

```
SELECT SPIDER_FLUSH_TABLE_MON_CACHE();
+-----+
| SPIDER_FLUSH_TABLE_MON_CACHE() |
+-----+
|                               1 |
+-----+
```

## 1.1.12.9.6 Window Functions

### 1.1.12.9.6.1 Window Functions Overview

#### Contents

1. [Introduction](#)
  1. [Syntax](#)
  2. [Description](#)
2. [Scope](#)
3. [Links](#)
4. [Examples](#)
5. [See Also](#)

MariaDB starting with [10.2](#)

Window functions were introduced in [MariaDB 10.2](#).

### Introduction

Window functions allow calculations to be performed across a set of rows related to the current row.

### Syntax

```

function (expression) OVER (
    [ PARTITION BY expression_list ]
    [ ORDER BY order_list [ frame_clause ] ] )

function:
A valid window function

expression_list:
expression | column_name [, expr_list ]

order_list:
expression | column_name [ ASC | DESC ]
[, ...]

frame_clause:
{ROWS | RANGE} {frame_border | BETWEEN frame_border AND frame_border}

frame_border:
| UNBOUNDED PRECEDING
| UNBOUNDED FOLLOWING
| CURRENT ROW
| expr PRECEDING
| expr FOLLOWING

```

## Description

In some ways, window functions are similar to [aggregate functions](#) in that they perform calculations across a set of rows. However, unlike aggregate functions, the output is not grouped into a single row.

Non-aggregate window functions include

- [CUME\\_DIST](#)
- [DENSE\\_RANK](#)
- [FIRST\\_VALUE](#)
- [LAG](#)
- [LAST\\_VALUE](#)
- [LEAD](#)
- [MEDIAN](#)
- [NTH\\_VALUE](#)
- [NTILE](#)
- [PERCENT\\_RANK](#)
- [PERCENTILE\\_CONT](#)
- [PERCENTILE\\_DISC](#)
- [RANK , ROW\\_NUMBER](#)

[Aggregate functions](#) that can also be used as window functions include

- [AVG](#)
- [BIT\\_AND](#)
- [BIT\\_OR](#)
- [BIT\\_XOR](#)
- [COUNT](#)
- [MAX](#)
- [MIN](#)
- [STD](#)
- [STDDEV](#)
- [STDDEV\\_POP](#)
- [STDDEV\\_SAMP](#)
- [SUM](#)
- [VAR\\_POP](#)
- [VAR\\_SAMP](#)
- [VARIANCE](#)

Window function queries are characterised by the OVER keyword, following which the set of rows used for the calculation is specified. By default, the set of rows used for the calculation (the "window") is the entire dataset, which can be ordered with the ORDER BY clause. The PARTITION BY clause is used to reduce the window to a particular group within the dataset.

For example, given the following data:

```

CREATE TABLE student (name CHAR(10), test CHAR(10), score TINYINT);

INSERT INTO student VALUES
('Chun', 'SQL', 75), ('Chun', 'Tuning', 73),
('Esben', 'SQL', 43), ('Esben', 'Tuning', 31),
('Kaolin', 'SQL', 56), ('Kaolin', 'Tuning', 88),
('Tatiana', 'SQL', 87), ('Tatiana', 'Tuning', 83);

```

the following two queries return the average partitioned by test and by name respectively:

```

SELECT name, test, score, AVG(score) OVER (PARTITION BY test)
  AS average_by_test FROM student;

+-----+-----+-----+-----+
| name | test | score | average_by_test |
+-----+-----+-----+-----+
| Chun | SQL | 75 | 65.2500 |
| Chun | Tuning | 73 | 68.7500 |
| Esben | SQL | 43 | 65.2500 |
| Esben | Tuning | 31 | 68.7500 |
| Kaolin | SQL | 56 | 65.2500 |
| Kaolin | Tuning | 88 | 68.7500 |
| Tatiana | SQL | 87 | 65.2500 |
| Tatiana | Tuning | 83 | 68.7500 |
+-----+-----+-----+-----+

SELECT name, test, score, AVG(score) OVER (PARTITION BY name)
  AS average_by_name FROM student;

+-----+-----+-----+-----+
| name | test | score | average_by_name |
+-----+-----+-----+-----+
| Chun | SQL | 75 | 74.0000 |
| Chun | Tuning | 73 | 74.0000 |
| Esben | SQL | 43 | 37.0000 |
| Esben | Tuning | 31 | 37.0000 |
| Kaolin | SQL | 56 | 72.0000 |
| Kaolin | Tuning | 88 | 72.0000 |
| Tatiana | SQL | 87 | 85.0000 |
| Tatiana | Tuning | 83 | 85.0000 |
+-----+-----+-----+-----+

```

It is also possible to specify which rows to include for the window function (for example, the current row and all preceding rows). See [Window Frames](#) for more details.

## Scope

Window functions were introduced in SQL:2003, and their definition was expanded in subsequent versions of the standard. The last expansion was in the latest version of the standard, SQL:2011.

Most database products support a subset of the standard, they implement some functions defined as late as in SQL:2011, and at the same time leave some parts of SQL:2008 unimplemented.

MariaDB:

- Supports ROWS and RANGE-type frames
  - All kinds of frame bounds are supported, including
 

```
RANGE PRECEDING|FOLLOWING n
frame bounds (unlike PostgreSQL or MS SQL Server)
```
  - Does not yet support DATE[TIME] datatype and arithmetic for RANGE-type frames ( [MDEV-9727](#) )
- Does not support GROUPS-type frames (it seems that no popular database supports it, either)
- Does not support frame exclusion (no other database seems to support it, either) ( [MDEV-9724](#) )
- Does not support explicit
 

```
NULLS FIRST
or
NULLS LAST
```
- Does not support nested navigation in window functions (this is
 

```
VALUE_OF(expr AT row_marker [, default_value)
syntax)
```
- The following window functions are supported:
  - "Streamable" window functions: [ROW\\_NUMBER](#) , [RANK](#) , [DENSE\\_RANK](#) ,
  - Window functions that can be streamed once the number of rows in partition is known: [PERCENT\\_RANK](#) , [CUME\\_DIST](#) , [NTILE](#)
- Aggregate functions that are currently supported as window functions are: [COUNT](#) , [SUM](#) , [AVG](#) , [BIT\\_OR](#) , [BIT\\_AND](#) , [BIT\\_XOR](#) .

- Aggregate functions with the DISTINCT specifier (e.g. COUNT( DISTINCT x) ) are not supported as window functions.

## Links

- [MDEV-6115](#) is the main jira task for window functions development. Other tasks are attached as sub-tasks
- [bb-10.2-mdev9543](#) is the feature tree for window functions. Development is ongoing, and this tree has the newest changes.
- Testcases are in mysql-test/t/win\*.test

## Examples

Given the following sample data:

```
CREATE TABLE users (
  email VARCHAR(30),
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  account_type VARCHAR(30)
);

INSERT INTO users VALUES
  ('admin@boss.org', 'Admin', 'Boss', 'admin'),
  ('bob.carlsen@foo.bar', 'Bob', 'Carlsen', 'regular'),
  ('eddie.stevens@data.org', 'Eddie', 'Stevens', 'regular'),
  ('john.smith@xyz.org', 'John', 'Smith', 'regular'),
  ('root@boss.org', 'Root', 'Chief', 'admin')
```

First, let's order the records by email alphabetically, giving each an ascending *rnum* value starting with 1. This will make use of the [ROW\\_NUMBER](#) window function:

```
SELECT row_number() OVER (ORDER BY email) AS rnum,
       email, first_name, last_name, account_type
  FROM users ORDER BY email;
```

rnum	email	first_name	last_name	account_type
1	admin@boss.org	Admin	Boss	admin
2	bob.carlsen@foo.bar	Bob	Carlsen	regular
3	eddie.stevens@data.org	Eddie	Stevens	regular
4	john.smith@xyz.org	John	Smith	regular
5	root@boss.org	Root	Chief	admin

We can generate separate sequences based on account type, using the PARTITION BY clause:

```
SELECT row_number() OVER (PARTITION BY account_type ORDER BY email) AS rnum,
       email, first_name, last_name, account_type
  FROM users ORDER BY account_type,email;
```

rnum	email	first_name	last_name	account_type
1	admin@boss.org	Admin	Boss	admin
2	root@boss.org	Root	Chief	admin
1	bob.carlsen@foo.bar	Bob	Carlsen	regular
2	eddie.stevens@data.org	Eddie	Stevens	regular
3	john.smith@xyz.org	John	Smith	regular

Given the following structure and data, we want to find the top 5 salaries from each department.

```

CREATE TABLE employee_salaries (dept VARCHAR(20), name VARCHAR(20), salary INT(11));

INSERT INTO employee_salaries VALUES
('Engineering', 'Dharma', 3500),
('Engineering', 'Binh', 3000),
('Engineering', 'Adalynn', 2800),
('Engineering', 'Samuel', 2500),
('Engineering', 'Cveta', 2200),
('Engineering', 'Ebele', 1800),
('Sales', 'Carbry', 500),
('Sales', 'Clytemnestra', 400),
('Sales', 'Juraj', 300),
('Sales', 'Kalpana', 300),
('Sales', 'Svantepolk', 250),
('Sales', 'Angelo', 200);

```

We could do this without using window functions, as follows:

```

select dept, name, salary
from employee_salaries as t1
where (select count(t2.salary)
      from employee_salaries as t2
      where t1.name != t2.name and
            t1.dept = t2.dept and
            t2.salary > t1.salary) < 5
order by dept, salary desc;

+-----+-----+-----+
| dept      | name       | salary |
+-----+-----+-----+
| Engineering | Dharma     | 3500  |
| Engineering | Binh       | 3000  |
| Engineering | Adalynn    | 2800  |
| Engineering | Samuel     | 2500  |
| Engineering | Cveta      | 2200  |
| Sales      | Carbry     | 500   |
| Sales      | Clytemnestra | 400   |
| Sales      | Juraj      | 300   |
| Sales      | Kalpana    | 300   |
| Sales      | Svantepolk | 250   |
+-----+-----+-----+

```

This has a number of disadvantages:

- if there is no index, the query could take a long time if the employee\_salary\_table is large
- Adding and maintaining indexes adds overhead, and even with indexes on `dept` and `salary`, each subquery execution adds overhead by performing a lookup through the index.

Let's try achieve the same with window functions. First, generate a rank for all employees, using the `RANK` function.

```

select rank() over (partition by dept order by salary desc) as ranking,
       dept, name, salary
  from employee_salaries
 order by dept, ranking;

+-----+-----+-----+
| ranking | dept      | name       | salary |
+-----+-----+-----+
|      1 | Engineering | Dharma     | 3500  |
|      2 | Engineering | Binh       | 3000  |
|      3 | Engineering | Adalynn    | 2800  |
|      4 | Engineering | Samuel     | 2500  |
|      5 | Engineering | Cveta      | 2200  |
|      6 | Engineering | Ebele      | 1800  |
|      1 | Sales      | Carbry     | 500   |
|      2 | Sales      | Clytemnestra | 400   |
|      3 | Sales      | Juraj      | 300   |
|      3 | Sales      | Kalpana    | 300   |
|      5 | Sales      | Svantepolk | 250   |
|      6 | Sales      | Angelo     | 200   |
+-----+-----+-----+

```

Each department has a separate sequence of ranks due to the `PARTITION BY` clause. This particular sequence of values for `rank()` is given by the `ORDER BY` clause inside the window function's `OVER` clause. Finally, to get our results in a readable format we order the data by `dept` and the newly generated `ranking` column.

Now, we need to reduce the results to find only the top 5 per department. Here is a common mistake:

```
select
rank() over (partition by dept order by salary desc) as ranking,
dept, name, salary
from employee_salaries
where ranking <= 5
order by dept, ranking;

ERROR 1054 (42S22): Unknown column 'ranking' in 'where clause'
```

Trying to filter only the first 5 values per department by putting a where clause in the statement does not work, due to the way window functions are computed. The computation of window functions happens after all WHERE, GROUP BY and HAVING clauses have been completed, right before ORDER BY, so the WHERE clause has no idea that the ranking column exists. It is only present after we have filtered and grouped all the rows.

To counteract this problem, we need to wrap our query into a derived table. We can then attach a where clause to it:

```
select * from (select rank() over (partition by dept order by salary desc) as ranking,
dept, name, salary
from employee_salaries) as salary_ranks
where (salary_ranks.ranking <= 5)
order by dept, ranking;
+-----+-----+-----+
| ranking | dept      | name       | salary   |
+-----+-----+-----+
| 1 | Engineering | Dharma    | 3500    |
| 2 | Engineering | Binh      | 3000    |
| 3 | Engineering | Adalynn   | 2800    |
| 4 | Engineering | Samuel    | 2500    |
| 5 | Engineering | Cveta     | 2200    |
| 1 | Sales      | Carbry    | 500     |
| 2 | Sales      | Clytemnestra | 400    |
| 3 | Sales      | Juraj     | 300     |
| 3 | Sales      | Kalpana   | 300     |
| 5 | Sales      | Svantepolk | 250    |
+-----+-----+-----+
```

## See Also

- [Window Frames](#)
- [Introduction to Window Functions in MariaDB Server 10.2](#)

[1.1.12.9.6.2 AVG](#)

[1.1.12.9.6.3 BIT\\_AND](#)

[1.1.12.9.6.4 BIT\\_OR](#)

[1.1.12.9.6.5 BIT\\_XOR](#)

[1.1.12.9.6.6 COUNT](#)

[1.1.12.9.6.7 COUNT DISTINCT](#)

[1.1.12.9.6.8 GROUP\\_CONCAT](#)

[1.1.12.9.6.9 JSON\\_ARRAYAGG](#)

[1.1.12.9.6.10 JSON\\_OBJECTAGG](#)

[1.1.12.9.6.11 MAX](#)

[1.1.12.9.6.12 MIN](#)

[1.1.12.9.6.13 STD](#)

[1.1.12.9.6.14 STDDEV](#)

[1.1.12.9.6.15 STDDEV\\_POP](#)

[1.1.12.9.6.16 STDDEV\\_SAMP](#)

[1.1.12.9.6.17 SUM](#)

[1.1.12.9.6.18 VARIANCE](#)

[1.1.12.9.6.19 VAR\\_POP](#)

[1.1.12.9.6.20 VAR\\_SAMP](#)

## 2 MariaDB Administration

### 2.1 Getting, Installing and Upgrading MariaDB

#### 2.1.1 Where to Download MariaDB

#### Contents

- 1. [The Latest Packages](#)
- 2. [Distributions Which Include MariaDB](#)
- 3. [Pre-Release Binaries](#)
- 4. [Getting the Source](#)
- 5. [Old Versions](#)

### The Latest Packages

Tarballs, binaries (Linux, Solaris, and Windows), and packages for some Linux distributions are available at [mariadb.com/downloads/](#) or [downloads.mariadb.org](#) .

We hope that interested [community](#) package maintainers will step forward, as others already have, to build packages for their distributions. We ask for strict adherence to your packaging system's best practices and invite you to create a [bug report](#) if our project impedes this in any way.

Instructions how to install the packages can be found [here](#) .

### Distributions Which Include MariaDB

- Most distributions already include MariaDB. See [Distributions Which Include MariaDB](#) .

### Pre-Release Binaries

Binaries from our [Buildbot](#) system (see also the [Buildbot](#) page), are available at <http://hasky.askmonty.org/archive> . They are not suitable for use in production systems but may be of use for debugging.

Once at the above URL you will need to click on the MariaDB tree you are interested in, and then the build. The build number corresponds to the `tarbuildnum` variable in Buildbot.

For example, if you were interested in the bsd9-64 build of the [MariaDB 5.5](#) tree, revision 3497, the `tarbuildnum` is listed in the "Build Properties" table of the [Buildbot build report](#) . In this case, the value is "2434".

### Getting the Source

You can find all the source code at <https://github.com/MariaDB/server>

To retrieve the code, the [Git](#) source control software offers the path of least resistance. If you are unfamiliar with git, please refer to the [git documentation](#) for an understanding of version control with git.

For instructions on creating a local branch of MariaDB, see the [Getting the MariaDB Source Code](#) page.

See the [Generic Build Instructions](#) page for general instructions on compiling MariaDB from the source. The [source](#) page has links to platform and distribution-specific information, including information on how we build the release packages.

## 2.1.2 MariaDB Binary Packages

### 2.1.2.1 Installing MariaDB RPM Files

MariaDB provides RPM packages for several RPM-based Linux distributions. MariaDB also provides YUM/DNF and ZYpp repositories for these Linux distributions. The articles here provide information and instructions on using the RPM packages and the related repositories.



#### About the MariaDB RPM Files

[Describes the contents of the RPM packages that come with each MariaDB release](#)



#### Installing MariaDB with yum/dnf

[Installing MariaDB with yum or dnf on RHEL, CentOS, Fedora, and similar distros.](#)



#### Installing MariaDB with zypper

[How to install MariaDB with zypper on SLES, OpenSUSE, and other similar Linux distributions.](#)



#### Installing MariaDB With the rpm Tool

[Downloading and installing RPM files using the rpm command](#)



#### Checking MariaDB RPM Package Signatures

[Steps to check MariaDB RPM package signatures](#)



#### Troubleshooting MariaDB Installs on Red Hat/CentOS

[Issues people have encountered when installing MariaDB on Red Hat / CentOS](#)



#### MariaDB for DirectAdmin Using RPMs

[Using DirectAdmin when installing MariaDB with YUM](#)



#### MariaDB Installation (Version 10.1.21) via RPMs on CentOS 7

[Detailed steps for installing MariaDB \(version 10.1.21\) via RPMs on CentOS 7](#)



#### Why Source RPMs (SRPMs) Aren't Packaged For Some Platforms

[Explanation for why source RPM \(SRPMs\) aren't packaged for some platforms](#)



#### Building MariaDB from a Source RPM

[How to build MariaDB from a source RPM \(SRPM\).](#)

There are [4](#) related questions .

### 2.1.2.1.1 About the MariaDB RPM Files

#### Contents

1. Available RPM Packages
  1. Available RPM Packages in MariaDB 10.4
  2. Available RPM Packages in MariaDB 10.2 and MariaDB 10.3
  3. Available RPM Packages in MariaDB 10.1
  4. Available RPM Packages in MariaDB 5.5
2. Installing RPM Packages
3. Actions Performed by RPM Packages
  1. Users and Groups Created
4. See also

## Available RPM Packages

The available RPM packages depend on the specific MariaDB release series.

## Available RPM Packages in MariaDB 10.4

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#), the following RPMs are available:

Package Name	Description
galera-4	The WSREP provider for <a href="#">Galera 4</a> .
MariaDB-backup	<a href="#">Mariabackup</a>
MariaDB-backup-debuginfo	Debuginfo for <a href="#">Mariabackup</a>
MariaDB-client	Client tools like mysql CLI, mysqldump , and others.
MariaDB-client-debuginfo	Debuginfo for client tools like mysql CLI, mysqldump , and others.
MariaDB-common	Character set files and <code>/etc/my.cnf</code>
MariaDB-common-debuginfo	Debuginfo for character set files and <code>/etc/my.cnf</code>
MariaDB-compat	Old shared client libraries, may be needed by old MariaDB or MySQL clients
MariaDB-connect-engine	The <a href="#">CONNECT</a> storage engine.
MariaDB-connect-engine-debuginfo	Debuginfo for the <a href="#">CONNECT</a> storage engine.
MariaDB-cracklib-password-check	The <a href="#">cracklib_password_check</a> password validation plugin.
MariaDB-cracklib-password-check	Debuginfo for the <a href="#">cracklib_password_check</a> password validation plugin.
MariaDB-devel	Development headers and static libraries.
MariaDB-devel-debuginfo	Debuginfo for development headers and static libraries.

MariaDB-gssapi-server	The <a href="#">gssapi</a> authentication plugin.
MariaDB-gssapi-server-debuginfo	Debuginfo for the <a href="#">gssapi</a> authentication plugin.
MariaDB-rocksdb-engine	The <a href="#">MyRocks</a> storage engine.
MariaDB-rocksdb-engine-debuginfo	Debuginfo for the <a href="#">MyRocks</a> storage engine.
MariaDB-server	The server and server tools, like <a href="#">myisamchk</a> and <a href="#">mysqlhotcopy</a> are here.
MariaDB-server-debuginfo	Debuginfo for the server and server tools, like <a href="#">myisamchk</a> and <a href="#">mysqlhotcopy</a> are here.
MariaDB-shared	Dynamic client libraries.
MariaDB-shared-debuginfo	Debuginfo for dynamic client libraries.
MariaDB-test	<a href="#">mysql-client-test</a> executable, and <b>mysql-test</b> framework with the tests.
MariaDB-test-debuginfo	Debuginfo for <a href="#">mysql-client-test</a> executable, and <b>mysql-test</b> framework with the tests.
MariaDB-tokudb-engine	The <a href="#">TokuDB</a> storage engine.
MariaDB-tokudb-engine-debuginfo	Debuginfo for the <a href="#">TokuDB</a> storage engine.

## Available RPM Packages in [MariaDB 10.2](#) and [MariaDB 10.3](#)

MariaDB starting with [10.2](#)

In [MariaDB 10.2](#) and [MariaDB 10.3](#), the following RPMs are available:

Package Name	Description
galera	The WSREP provider for <a href="#">Galera 3</a> .
MariaDB-backup	<a href="#">Mariabackup</a>
MariaDB-backup-debuginfo	Debuginfo for <a href="#">Mariabackup</a>

MariaDB-client	Client tools like mysql CLI, mysqldump , and others.
MariaDB-client-debuginfo	Debuginfo for client tools like mysql CLI, mysqldump , and others.
MariaDB-common	Character set files and /etc/my.cnf
MariaDB-common-debuginfo	Debuginfo for character set files and /etc/my.cnf
MariaDB-compat	Old shared client libraries, may be needed by old MariaDB or MySQL clients
MariaDB-connect-engine	The <a href="#">CONNECT</a> storage engine.
MariaDB-connect-engine-debuginfo	Debuginfo for the <a href="#">CONNECT</a> storage engine.
MariaDB-cracklib-password-check	The <a href="#">cracklib_password_check</a> password validation plugin.
MariaDB-cracklib-password-check	Debuginfo for the <a href="#">cracklib_password_check</a> password validation plugin.
MariaDB-devel	Development headers and static libraries.
MariaDB-devel-debuginfo	Debuginfo for development headers and static libraries.
MariaDB-gssapi-server	The <a href="#">gssapi</a> authentication plugin.
MariaDB-gssapi-server-debuginfo	Debuginfo for the <a href="#">gssapi</a> authentication plugin.
MariaDB-rocksdb-engine	The <a href="#">MyRocks</a> storage engine.
MariaDB-rocksdb-engine-debuginfo	Debuginfo for the <a href="#">MyRocks</a> storage engine.

MariaDB-server	The server and server tools, like <a href="#">myisamchk</a> and <a href="#">mysqlhotcopy</a> are here.
MariaDB-server-debuginfo	Debuginfo for the server and server tools, like <a href="#">myisamchk</a> and <a href="#">mysqlhotcopy</a> are here.
MariaDB-shared	Dynamic client libraries.
MariaDB-shared-debuginfo	Debuginfo for dynamic client libraries.
MariaDB-test	<a href="#">mysql-client-test</a> executable, and <b>mysql-test</b> framework with the tests.
MariaDB-test-debuginfo	Debuginfo for <a href="#">mysql-client-test</a> executable, and <b>mysql-test</b> framework with the tests.
MariaDB-tokudb-engine	The <a href="#">TokuDB</a> storage engine.
MariaDB-tokudb-engine-debuginfo	Debuginfo for the <a href="#">TokuDB</a> storage engine.

## Available RPM Packages in MariaDB 10.1

MariaDB starting with [10.1](#)

In [MariaDB 10.1](#), the following RPMs are available:

Package Name	Description
galera	The WSREP provider for <a href="#">Galera</a> 3.
MariaDB-backup	<a href="#">Mariabackup</a>
MariaDB-backup-debuginfo	Debuginfo for <a href="#">Mariabackup</a>
MariaDB-client	Client tools like <a href="#">mysql</a> <a href="#">CLI</a> , <a href="#">mysqldump</a> , and others.
MariaDB-client-debuginfo	Debuginfo for client tools like <a href="#">mysql</a> <a href="#">CLI</a> , <a href="#">mysqldump</a> , and others.
MariaDB-common	Character set files and <code>/etc/my.cnf</code>
MariaDB-common-debuginfo	Debuginfo for character set files and <code>/etc/my.cnf</code>
MariaDB-compat	Old shared client libraries, may be needed by old MariaDB or MySQL clients

MariaDB-connect-engine	The  CONNECT  storage engine.
MariaDB-connect-engine-debuginfo	Debuginfo for the  CONNECT  storage engine.
MariaDB-cracklib-password-check	The  cracklib_password_check  password validation plugin.
MariaDB-cracklib-password-check	Debuginfo for the  cracklib_password_check  password validation plugin.
MariaDB-devel	Development headers and static libraries.
MariaDB-devel-debuginfo	Debuginfo for development headers and static libraries.
MariaDB-gssapi-server	The  gssapi  authentication plugin.
MariaDB-gssapi-server-debuginfo	Debuginfo for the  gssapi  authentication plugin.
MariaDB-server	The server and server tools, like <a href="#">myisamchk</a> and <a href="#">mysqlhotcopy</a> are here.
MariaDB-server-debuginfo	Debuginfo for the server and server tools, like <a href="#">myisamchk</a> and <a href="#">mysqlhotcopy</a> are here.
MariaDB-shared	Dynamic client libraries.
MariaDB-shared-debuginfo	Debuginfo for dynamic client libraries.
MariaDB-test	<a href="#">mysql-client-test</a> executable, and <b>mysql-test</b> framework with the tests.
MariaDB-test-debuginfo	Debuginfo for  <a href="#">mysql-client-test</a> executable, and <b>mysql-test</b> framework with the tests.
MariaDB-tokudb-engine	The <a href="#">TokuDB</a> storage engine.
MariaDB-tokudb-engine-debuginfo	Debuginfo for the <a href="#">TokuDB</a> storage engine.

## Available RPM Packages in MariaDB 5.5

MariaDB starting with 5.5

In MariaDB 5.5 , the following RPMs are available:

Package Name	Description
MariaDB-client	Client tools like mysql CLI, mysqldump , and others.
MariaDB-client-debuginfo	Debuginfo for client tools like mysql CLI, mysqldump , and others.
MariaDB-common	Character set files and /etc/my.cnf
MariaDB-common-debuginfo	Debuginfo for character set files and /etc/my.cnf
MariaDB-compat	Old shared client libraries, may be needed by old MariaDB or MySQL clients
MariaDB-devel	Development headers and static libraries.
MariaDB-devel-debuginfo	Debuginfo for development headers and static libraries.
MariaDB-server	The server and server tools, like myisamchk and mysqlhotcopy are here.
MariaDB-server-debuginfo	Debuginfo for the server and server tools, like myisamchk and mysqlhotcopy are here.
MariaDB-shared	Dynamic client libraries.
MariaDB-shared-debuginfo	Debuginfo for dynamic client libraries.
MariaDB-test	mysql-client-test executable, and mysql-test framework with the tests.
MariaDB-test-debuginfo	Debuginfo for mysql-client-test executable, and mysql-test framework with the tests.

## Installing RPM Packages

Preferably, you should install MariaDB RPM packages using the package manager of your Linux distribution, for example

yum  
or  
zypper  
. But you can also use the lower-level  
rpm  
tool.

# Actions Performed by RPM Packages

## Users and Groups Created

When the

`MariaDB-server`  
RPM package is installed, it will create a user and group named  
`mysql`  
, if it does not already exist.

## See also

- [Installing MariaDB with yum](#)

### 2.1.2.1.2 Installing MariaDB with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant [RPM packages](#) from MariaDB's repository using

`yum`

or

`dnf`

. Starting with RHEL 8 and Fedora 22,

`yum`  
has been replaced by  
`dnf`  
, which is the next major version of  
`yum`  
. However,  
`yum`  
commands still work on many systems that use  
`dnf`

This page walks you through the simple installation steps using

`yum`

.

## Contents

1. Adding the MariaDB YUM repository
  1. Using the MariaDB Package Repository Setup Script
  2. Using the MariaDB Repository Configuration Tool
  3. Pinning the MariaDB Repository to a Specific Minor Release
2. Updating the MariaDB YUM repository to a New Major Release
  1. Updating the Major Release with the MariaDB Package Repository Setup Script
  2. Updating the Major Release with the MariaDB Repository Configuration Tool
3. Importing the MariaDB GPG Public Key
4. Installing MariaDB Packages with YUM
  1. Installing the Most Common Packages with YUM
  2. Installing MariaDB Server with YUM
  3. Installing MariaDB Galera Cluster with YUM
  4. Installing MariaDB Clients and Client Libraries with YUM
  5. Installing Mariabackup with YUM
  6. Installing Plugins with YUM
  7. Installing Debug Info Packages with YUM
    1. Installing Debug Info for the Most Common Packages with YUM
    2. Installing Debug Info for MariaDB Server with YUM
    3. Installing Debug Info for MariaDB Clients and Client Libraries with YUM
    4. Installing Debug Info for Mariabackup with YUM
    5. Installing Debug Info for Plugins with YUM
  8. Installing Older Versions from the Repository
5. After Installation

## Adding the MariaDB YUM repository

We currently have YUM repositories for the following Linux distributions:

- Red Hat Enterprise Linux (RHEL) 6
- Red Hat Enterprise Linux (RHEL) 7
- CentOS 6
- CentOS 7
- Fedora 27
- Fedora 28
- Fedora 29

## Using the MariaDB Package Repository Setup Script

If you want to install MariaDB with

```
yum  
, then you can configure  
yum  
to install from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script.
```

MariaDB Corporation provides a MariaDB Package Repository for several Linux distributions that use

```
yum  
to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, clients and utilities, client libraries, plugins, and Mariabackup. The MariaDB Package Repository setup script automatically configures your system to install packages from the MariaDB Package Repository.
```

To use the script, execute the following command:

```
curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
```

Note that this script also configures a repository for [MariaDB MaxScale](#) and a repository for MariaDB Tools, which currently only contains [Percona XtraBackup](#) and its dependencies.

See [MariaDB Package Repository Setup and Usage](#) for more information.

## Using the MariaDB Repository Configuration Tool

If you want to install MariaDB with

```
yum  
, then you can configure  
yum  
to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.
```

The MariaDB Foundation provides a MariaDB repository for several Linux distributions that use

```
yum  
to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, clients and utilities, client libraries, plugins, and Mariabackup. The MariaDB Repository Configuration Tool can easily generate the appropriate configuration file to add the repository for your distribution.
```

Once you have the appropriate repository configuration section for your distribution, add it to a file named

```
MariaDB.repo  
under  
/etc/yum.repos.d/  
:
```

For example, if you wanted to use the repository to install [MariaDB 10.3](#) on CentOS 7, then you could use the following

```
yum  
repository configuration in  
/etc/yum.repos.d/MariaDB.repo  
:  
[mariadb]  
name = MariaDB  
baseurl = http://yum.mariadb.org/10.3/centos7-amd64  
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB  
gpgcheck=1
```

The example file above includes a

```
gpgkey  
line to automatically fetch the GPG public key that is used to verify the digital signatures of the packages in our repositories. This allows the the  
yum  
,  
dnf  
, and  
rpm  
utilities to verify the integrity of the packages that they install.
```

## Pinning the MariaDB Repository to a Specific Minor Release

If you wish to pin the

```
yum  
repository to a specific minor release, or if you would like to do a  
yum downgrade  
to a specific minor release, then you can create a  
yum  
repository configuration with a  
baseurl  
option set to that specific minor release.
```

The MariaDB Foundation archives repositories of old minor releases at the following URL:

- <http://archive.mariadb.org/>

So if you can't find the repository of a specific minor release at

```
yum.mariadb.org  
, then it would be a good idea to check the archive.
```

For example, if you wanted to pin your repository to [MariaDB 10.3.14](#) on CentOS 7, then you could use the following

```
yum  
repository configuration in  
/etc/yum.repos.d/MariaDB.repo
```

```
[mariadb]
name = MariaDB-10.3.14
baseurl=http://yum.mariadb.org/10.3.14/centos7-amd64
# alternative: baseurl=http://archive.mariadb.org/mariadb-10.3.14/yum/centos7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

Note that if you change an existing repository configuration, then you need to execute the following:

```
sudo yum clean all
```

## Updating the MariaDB YUM repository to a New Major Release

MariaDB's

yum

repository can be updated to a new major release. How this is done depends on how you originally configured the repository.

### Updating the Major Release with the MariaDB Package Repository Setup Script

If you configured

yum

to install from MariaDB Corporation's MariaDB Package Repository by using the [MariaDB Package Repository setup script](#), then you can update the major release that the repository uses by running the script again.

### Updating the Major Release with the MariaDB Repository Configuration Tool

If you configured

yum

to install from MariaDB Foundation's MariaDB Repository by using the [MariaDB Repository Configuration Tool](#), then you can update the major release that the repository uses by updating the

yum

repository configuration file in-place. For example, if you wanted to change the repository from [MariaDB 10.2](#) to [MariaDB 10.3](#), and if the repository configuration file was at

/etc/yum.repos.d/MariaDB.repo

, then you could execute the following:

```
sudo sed -i 's/10.2/10.3/' /etc/yum.repos.d/MariaDB.repo
```

After that, the repository should refer to [MariaDB 10.3](#).

If the

yum

repository is pinned to a specific minor release, then the above

sed

command can result in an invalid repository configuration. In that case, the recommended options are:

- Edit the

MariaDB.repo

repository file manually.

- Or delete the

MariaDB.repo

repository file, and then install the repository of the new version with the more robust [MariaDB Package Repository setup script](#).

## Importing the MariaDB GPG Public Key

Before MariaDB can be installed, you also have to import the GPG public key that is used to verify the digital signatures of the packages in our repositories. This allows the

yum

,

dnf

and

rpm

utilities to verify the integrity of the packages that they install.

The id of our GPG public key is

0xcbcb082a1bb943db

. The short form of the id is

0x1BB943DB

. The full key fingerprint is:

```
1993 69E5 404B D5FC 7D2F E43B CBCB 082A 1BB9 43DB
```

yum

should prompt you to import the GPG public key the first time that you install a package from MariaDB's repository. However, if you like, the

rpm

utility can be used to manually import this key instead. For example:

```
sudo rpm --import https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
```

Once the GPG public key is imported, you are ready to install packages from the repository.

## Installing MariaDB Packages with YUM

After the

yum

repository is configured, you can install MariaDB by executing the

yum

command. The specific command that you would use would depend on which specific packages that you want to install.

## Installing the Most Common Packages with YUM

MariaDB starting with 10.4

In [MariaDB 10.4](#) and later, to Install the most common packages, execute the following command:

```
sudo yum install MariaDB-server galera-4 MariaDB-client MariaDB-shared MariaDB-backup MariaDB-common
```

MariaDB until 10.3

In [MariaDB 10.3](#) and before, to Install the most common packages, execute the following command:

```
sudo yum install MariaDB-server galera MariaDB-client MariaDB-shared MariaDB-backup MariaDB-common
```

## Installing MariaDB Server with YUM

To Install MariaDB Server, execute the following command:

```
sudo yum install MariaDB-server
```

## Installing MariaDB Galera Cluster with YUM

The process to install MariaDB Galera Cluster with the MariaDB

yum

repository is practically the same as installing standard MariaDB Server.

In [MariaDB 10.1](#) and later, Galera Cluster support has been included in the standard MariaDB Server packages, so you will need to install the  
MariaDB-server  
package, as you normally would.

In [MariaDB 10.4](#) and later, you also need to install the  
galera-4  
package to obtain the [Galera 4 wsrep provider library](#).

In [MariaDB 10.3](#) and before, you also need to install the  
galera  
package to obtain the [Galera 3 wsrep provider library](#).

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, to install MariaDB Galera Cluster, you could execute the following command:

```
sudo yum install MariaDB-server MariaDB-client galera-4
```

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, to install MariaDB Galera Cluster, you could execute the following command:

```
sudo yum install MariaDB-server MariaDB-client galera
```

If you haven't yet imported the MariaDB GPG public key, then

`yum`

will prompt you to import it after it downloads the packages, but before it prompts you to install them.

See [MariaDB Galera Cluster](#) for more information on MariaDB Galera Cluster.

## Installing MariaDB Clients and Client Libraries with YUM

In [MariaDB 10.2](#) and later, [MariaDB Connector/C](#) has been included as the client library. However, the package name for the client library has not been changed.

To Install the clients and client libraries, execute the following command:

```
sudo yum install MariaDB-client MariaDB-shared
```

## Installing Mariabackup with YUM

To install [Mariabackup](#), execute the following command:

```
sudo yum install MariaDB-backup
```

## Installing Plugins with YUM

Some [plugins](#) may also need to be installed.

For example, to install the

`cracklib_password_check`

password validation plugin, execute the following command:

```
sudo yum install MariaDB-cracklib-password-check
```

## Installing Debug Info Packages with YUM

MariaDB starting with [5.5.64](#)

The MariaDB

`yum`

repository first added

`debuginfo`

packages in [MariaDB 5.5.64](#) , [MariaDB 10.1.39](#) , [MariaDB 10.2.23](#) , [MariaDB 10.3.14](#) , and [MariaDB 10.4.4](#) .

The MariaDB

`yum`

repository also contains

`debuginfo`

packages. These package may be needed when [debugging a problem](#) .

## Installing Debug Info for the Most Common Packages with YUM

To install

[debuginfo](#)

for the most common packages, execute the following command:

```
sudo yum install MariaDB-server-debuginfo MariaDB-client-debuginfo MariaDB-shared-debuginfo MariaDB-backup-debuginfo MariaDB-com
```

## Installing Debug Info for MariaDB Server with YUM

To install

[debuginfo](#)

for MariaDB Server, execute the following command:

```
sudo yum install MariaDB-server-debuginfo
```

## Installing Debug Info for MariaDB Clients and Client Libraries with YUM

In [MariaDB 10.2](#) and later, [MariaDB Connector/C](#) has been included as the client library. However, the package name for the client library has not been changed.

To install

[debuginfo](#)

for the clients and client libraries, execute the following command:

```
sudo yum install MariaDB-client-debuginfo MariaDB-shared-debuginfo
```

## Installing Debug Info for Mariabackup with YUM

To install

[debuginfo](#)

for [Mariabackup](#), execute the following command:

```
sudo yum install MariaDB-backup-debuginfo
```

## Installing Debug Info for Plugins with YUM

For some [plugins](#),

[debuginfo](#)

may also need to be installed.

For example, to install

[debuginfo](#)

for the

[cracklib\\_password\\_check](#)

password validation plugin, execute the following command:

```
sudo yum install MariaDB-cracklib-password-check-debuginfo
```

## Installing Older Versions from the Repository

The MariaDB

[yum](#)

repository contains the last few versions of MariaDB. To show what versions are available, use the following command:

```
yum list --showduplicates MariaDB-server
```

In the output you will see the available versions. For example:

```
$ yum list --showduplicates MariaDB-server
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: centos.mirrors.ovh.net
 * extras: centos.mirrors.ovh.net
 * updates: centos.mirrors.ovh.net
Available Packages
MariaDB-server.x86_64    10.3.10-1.el7.centos      mariadb
MariaDB-server.x86_64    10.3.11-1.el7.centos      mariadb
MariaDB-server.x86_64    10.3.12-1.el7.centos      mariadb
mariadb-server.x86_64    1:5.5.60-1.el7_5          base
```

The MariaDB

```
yum
repository in this example contains MariaDB 10.3.10, MariaDB 10.3.11, and MariaDB 10.3.12. The CentOS base
yum
repository also contains MariaDB 5.5.60.
```

To install an older version of a package instead of the latest version we just need to specify the package name, a dash, and then the version number. And we only need to specify enough of the version number for it to be unique from the other available versions.

However, when installing an older version of a package, if

```
yum
has to install dependencies, then it will automatically choose to install the latest versions of those packages. To ensure that all MariaDB
packages are on the same version in this scenario, it is necessary to specify them all.
```

The packages that the MariaDB-server package depend on are: MariaDB-client, MariaDB-shared, and MariaDB-common. Therefore, to install [MariaDB 10.3.11](#) from this

```
yum
repository, we would do the following:
```

```
sudo yum install MariaDB-server-10.3.11 MariaDB-client-10.3.11 MariaDB-shared-10.3.11 MariaDB-backup-10.3.11 MariaDB-common-10.3
```

The rest of the install and setup process is as normal.

## After Installation

After the installation is complete, you can [start MariaDB](#).

If you are using [MariaDB Galera Cluster](#), then keep in mind that the first node will have to be [bootstrapped](#).

### 2.1.2.1.3 Installing MariaDB with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant [RPM packages](#) from MariaDB's repository using

```
zypper
```

This page walks you through the simple installation steps using

```
zypper
```

## Contents

1. Adding the MariaDB ZYpp repository
  1. Using the MariaDB Package Repository Setup Script
  2. Using the MariaDB Repository Configuration Tool
  3. Pinning the MariaDB Repository to a Specific Minor Release
2. Updating the MariaDB ZYpp repository to a New Major Release
  1. Updating the Major Release with the MariaDB Package Repository Setup Script
  2. Updating the Major Release with the MariaDB Repository Configuration Tool
3. Importing the MariaDB GPG Public Key
4. Installing MariaDB Packages with ZYpp
  1. Installing the Most Common Packages with ZYpp
  2. Installing MariaDB Server with ZYpp
  3. Installing MariaDB Galera Cluster with ZYpp
  4. Installing MariaDB Clients and Client Libraries with ZYpp
  5. Installing Mariabackup with ZYpp
  6. Installing Plugins with ZYpp
  7. Installing Debug Info Packages with ZYpp
    1. Installing Debug Info for the Most Common Packages with ZYpp
    2. Installing Debug Info for MariaDB Server with ZYpp
    3. Installing Debug Info for MariaDB Clients and Client Libraries with ZYpp
    4. Installing Debug Info for Mariabackup with ZYpp
    5. Installing Debug Info for Plugins with ZYpp
  8. Installing Older Versions from the Repository
5. After Installation

## Adding the MariaDB ZYpp repository

We currently have ZYpp repositories for the following Linux distributions:

- SUSE Linux Enterprise Server (SLES) 12
- SUSE Linux Enterprise Server (SLES) 15
- OpenSUSE 15
- OpenSUSE 42

## Using the MariaDB Package Repository Setup Script

If you want to install MariaDB with

zypper  
, then you can configure  
zypper  
to install from MariaDB Corporation's MariaDB Package Repository by using the [MariaDB Package Repository setup script](#).

MariaDB Corporation provides a MariaDB Package Repository for several Linux distributions that use

zypper  
to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, [clients and utilities](#), [client libraries](#), [plugins](#), and [Mariabackup](#). The MariaDB Package Repository setup script automatically configures your system to install packages from the MariaDB Package Repository.

To use the script, execute the following command:

```
curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
```

Note that this script also configures a repository for [MariaDB MaxScale](#) and a repository for MariaDB Tools, which currently only contains [Percona](#)

[XtraBackup](#) and its dependencies.

See [MariaDB Package Repository Setup and Usage](#) for more information.

## Using the MariaDB Repository Configuration Tool

If you want to install MariaDB with

```
zypper  
, then you can configure  
zypper  
to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool.
```

The MariaDB Foundation provides a MariaDB repository for several Linux distributions that use

```
zypper  
to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, clients and utilities, client libraries, plugins, and Mariabackup. The MariaDB Repository Configuration Tool can easily generate the appropriate commands to add the repository for your distribution.
```

For example, if you wanted to use the repository to install [MariaDB 10.3](#) on SLES 15, then you could use the following commands to add the MariaDB

```
zypper  
repository:
```

```
sudo zypper addrepo --gpgcheck --refresh https://yum.mariadb.org/10.3/sles/15/x86_64 mariadb  
sudo zypper --gpg-auto-import-keys refresh
```

## Pinning the MariaDB Repository to a Specific Minor Release

If you wish to pin the

```
zypper  
repository to a specific minor release, or if you would like to downgrade to a specific minor release, then you can create a  
zypper  
repository with the URL hard-coded to that specific minor release.
```

The MariaDB Foundation archives repositories of old minor releases at the following URL:

- <http://archive.mariadb.org/>

So if you can't find the repository of a specific minor release at

```
yum.mariadb.org  
, then it would be a good idea to check the archive.
```

For example, if you wanted to pin your repository to [MariaDB 10.3.14](#) on SLES 15, then you could use the following commands to add the MariaDB

```
zypper  
repository:
```

```
sudo zypper removerepo mariadb  
sudo zypper addrepo --gpgcheck --refresh https://yum.mariadb.org/10.3.14/sles/15/x86_64 mariadb
```

## Updating the MariaDB ZYpp repository to a New Major Release

MariaDB's

```
zypper  
repository can be updated to a new major release. How this is done depends on how you originally configured the repository.
```

## Updating the Major Release with the MariaDB Package Repository Setup Script

If you configured

```
zypper  
to install from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script, then you can update  
the major release that the repository uses by running the script again.
```

## Updating the Major Release with the MariaDB Repository Configuration Tool

If you configured

```
zypper  
to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool, then you can update the major  
release that the repository uses by removing the repository for the old version and adding the repository for the new version.
```

First, you can remove the repository for the old version by executing the following command:

```
sudo zypper removerepo mariadb
```

After that, you can add the repository for the new version. For example, if you wanted to use the repository to install [MariaDB 10.3](#) on SLES 15, then you could use the following commands to add the MariaDB

```
zypper  
repository:
```

```
sudo zypper addrepo --gpgcheck --refresh https://yum.mariadb.org/10.3/sles/15/x86_64 mariadb  
sudo zypper --gpg-auto-import-keys refresh
```

After that, the repository should refer to [MariaDB 10.3](#).

## Importing the MariaDB GPG Public Key

Before MariaDB can be installed, you also have to import the GPG public key that is used to verify the digital signatures of the packages in our repositories. This allows the the

```
zypper  
and  
rpm  
utilities to verify the integrity of the packages that they install.
```

The id of our GPG public key is

```
0xcbcb082a1bb943db  
. The short form of the id is  
0x1BB943DB  
. The full key fingerprint is:
```

```
1993 69E5 404B D5FC 7D2F E43B CBCB 082A 1BB9 43DB
```

The

```
rpm
```

utility can be used to import this key. For example:

```
sudo rpm --import https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
```

Once the GPG public key is imported, you are ready to install packages from the repository.

## Installing MariaDB Packages with ZYpp

After the

```
zypper  
repository is configured, you can install MariaDB by executing the
```

```
zypper
```

command. The specific command that you would use would depend on which specific packages that you want to install.

## Installing the Most Common Packages with ZYpp

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, to Install the most common packages, execute the following command:

```
sudo zypper install MariaDB-server galera-4 MariaDB-client MariaDB-shared MariaDB-backup MariaDB-common
```

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, to Install the most common packages, execute the following command:

```
sudo zypper install MariaDB-server galera MariaDB-client MariaDB-shared MariaDB-backup MariaDB-common
```

## Installing MariaDB Server with ZYpp

To Install MariaDB Server, execute the following command:

```
sudo zypper install MariaDB-server
```

## Installing MariaDB Galera Cluster with ZYpp

The process to install MariaDB Galera Cluster with the MariaDB

```
zypper  
repository is practically the same as installing standard MariaDB Server.
```

In [MariaDB 10.1](#) and later, Galera Cluster support has been included in the standard MariaDB Server packages, so you will need to install the  
MariaDB-server  
package, as you normally would.

In [MariaDB 10.4](#) and later, you also need to install the  
galera-4  
package to obtain the [Galera 4 wsrep provider library](#).

In [MariaDB 10.3](#) and before, you also need to install the  
galera  
package to obtain the [Galera 3 wsrep provider library](#).

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, to install MariaDB Galera Cluster, you could execute the following command:

```
sudo zypper install MariaDB-server MariaDB-client galera-4
```

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, to install MariaDB Galera Cluster, you could execute the following command:

```
sudo zypper install MariaDB-server MariaDB-client galera
```

If you haven't yet imported the MariaDB GPG public key, then

```
zypper  
will prompt you to import it after it downloads the packages, but before it prompts you to install them.
```

See [MariaDB Galera Cluster](#) for more information on MariaDB Galera Cluster.

## Installing MariaDB Clients and Client Libraries with ZYpp

In [MariaDB 10.2](#) and later, [MariaDB Connector/C](#) has been included as the client library. However, the package name for the client library has not been changed.

To Install the clients and client libraries, execute the following command:

```
sudo zypper install MariaDB-client MariaDB-shared
```

## Installing Mariabackup with ZYpp

To install [Mariabackup](#), execute the following command:

```
sudo zypper install MariaDB-backup
```

## Installing Plugins with ZYpp

Some [plugins](#) may also need to be installed.

For example, to install the

```
cracklib_password_check
```

password validation plugin, execute the following command:

```
sudo zypper install MariaDB-cracklib-password-check
```

## Installing Debug Info Packages with ZYpp

MariaDB starting with [5.5.64](#)

#### The MariaDB

```
zypper  
repository first added
```

```
debuginfo
```

packages in [MariaDB 5.5.64](#) , [MariaDB 10.1.39](#) , [MariaDB 10.2.23](#) , [MariaDB 10.3.14](#) , and [MariaDB 10.4.4](#) .

#### The MariaDB

```
zypper  
repository also contains
```

```
debuginfo
```

packages. These package may be needed when [debugging a problem](#) .

### Installing Debug Info for the Most Common Packages with ZYpp

To install

```
debuginfo
```

for the most common packages, execute the following command:

```
sudo zypper install MariaDB-server-debuginfo MariaDB-client-debuginfo MariaDB-shared-debuginfo MariaDB-backup-debuginfo MariaDB-
```

### Installing Debug Info for MariaDB Server with ZYpp

To install

```
debuginfo
```

for MariaDB Server, execute the following command:

```
sudo zypper install MariaDB-server-debuginfo
```

### Installing Debug Info for MariaDB Clients and Client Libraries with ZYpp

In [MariaDB 10.2](#) and later, [MariaDB Connector/C](#) has been included as the client library. However, the package name for the client library has not been changed.

To install

```
debuginfo
```

for the clients and client libraries, execute the following command:

```
sudo zypper install MariaDB-client-debuginfo MariaDB-shared-debuginfo
```

### Installing Debug Info for Mariabackup with ZYpp

To install

```
debuginfo
```

for [Mariabackup](#) , execute the following command:

```
sudo zypper install MariaDB-backup-debuginfo
```

### Installing Debug Info for Plugins with ZYpp

For some [plugins](#) ,

```
debuginfo
```

may also need to be installed.

For example, to install

```
debuginfo
```

for the

```
cracklib_password_check
```

password validation plugin, execute the following command:

```
sudo zypper install MariaDB-cracklib-password-check-debuginfo
```

## Installing Older Versions from the Repository

The MariaDB

```
zypper
```

repository contains the last few versions of MariaDB. To show what versions are available, use the following command:

```
zypper search --details MariaDB-server
```

In the output you will see the available versions.

To install an older version of a package instead of the latest version we just need to specify the package name, a dash, and then the version number. And we only need to specify enough of the version number for it to be unique from the other available versions.

However, when installing an older version of a package, if

```
zypper
```

has to install dependencies, then it will automatically choose to install the latest versions of those packages. To ensure that all MariaDB packages are on the same version in this scenario, it is necessary to specify them all.

The packages that the MariaDB-server package depend on are: MariaDB-client, MariaDB-shared, and MariaDB-common. Therefore, to install [MariaDB 10.3.14](#) from this

```
zypper
```

repository, we would do the following:

```
sudo zypper install MariaDB-server-10.3.14 MariaDB-client-10.3.14 MariaDB-shared-10.3.14 MariaDB-backup-10.3.14 MariaDB-common-1
```

The rest of the install and setup process is as normal.

## After Installation

After the installation is complete, you can [start MariaDB](#).

If you are using [MariaDB Galera Cluster](#), then keep in mind that the first node will have to be [bootstrapped](#).

### 2.1.2.1.4 Installing MariaDB With the rpm Tool

This article describes how to download the RPM files and install them using the

```
rpm
```

command.

It is highly recommended to [Install MariaDB with yum](#) where possible.

Navigate to <http://downloads.mariadb.org> and choose the desired database version and then select the RPMs that match your Linux distribution and architecture.

Clicking those links takes you to a local mirror. Choose the rpms link and download the desired packages. The packages will be similar to the following:

```
MariaDB-client-5.2.5-99.el5.x86_64.rpm  
MariaDB-debuginfo-5.2.5-99.el5.x86_64.rpm  
MariaDB-devel-5.2.5-99.el5.x86_64.rpm  
MariaDB-server-5.2.5-99.el5.x86_64.rpm  
MariaDB-shared-5.2.5-99.el5.x86_64.rpm  
MariaDB-test-5.2.5-99.el5.x86_64.rpm
```

For a standard server installation you will need to download at least the *client*, *shared*, and *server* RPM files. See [About the MariaDB RPM Files](#) for more information about what is included in each RPM package.

After downloading the MariaDB RPM files, you might want to check their signatures. See [Checking MariaDB RPM Package Signatures](#) for more information about checking signatures.

```
rpm --checksig $(find . -name '*.rpm')
```

Prior to installing MariaDB, be aware that it will conflict with an existing installation of MySQL. To check whether MySQL is already installed, issue the command:

```
rpm -qa 'mysql*'
```

If necessary, you can remove found MySQL packages before installing MariaDB.

To install MariaDB, use the command:

```
rpm -ivh MariaDB-*
```

You should see output such as the following:

```
Preparing...                                           ###### [100%]
1:MariaDB-shared                                     ##### [ 14%]
2:MariaDB-client                                     ##### [ 29%]
3:MariaDB-client                                     ##### [ 43%]
4:MariaDB-debuginfo                                  ##### [ 57%]
5:MariaDB-devel                                       ##### [ 71%]
6:MariaDB-server                                     ##### [ 86%]
```

PLEASE REMEMBER TO SET A PASSWORD FOR THE MariaDB root USER !

To do so, start the server, then issue the following commands:

```
/usr/bin/mysqladmin -u root password 'new-password'
/usr/bin/mysqladmin -u root -h hostname password 'new-password'
```

Alternatively you can run:

```
/usr/bin/mysql_secure_installation
```

which will also give you the option of removing the test databases and anonymous user created by default. This is strongly recommended for production servers.

See the MySQL manual for more instructions.

Please report any problems with the /usr/bin/mysqlbug script!

The latest information about MariaDB is available at <http://www.askmonty.org/>.

You can find additional information about the MySQL part at:

<http://dev.mysql.com>

Support MariaDB development by buying support/new features from

Monty Program Ab. You can contact us about this at [sales@askmonty.org](mailto:sales@askmonty.org).

Alternatively consider joining our community based development effort:

[http://askmonty.org/wiki/index.php/MariaDB#How\\_can\\_I\\_participate\\_in\\_the\\_development\\_of\\_MariaDB](http://askmonty.org/wiki/index.php/MariaDB#How_can_I_participate_in_the_development_of_MariaDB)

Starting MySQL....[ OK ]

Giving mysqld 2 seconds to start

```
7:MariaDB-test                                     ##### [100%]
```

Be sure to follow the instructions given in the preceding output and create a password for the root user either by using mysqladmin or by running the /usr/bin/mysql\_secure\_installation script.

Installing the MariaDB RPM files installs the MySQL tools in the

```
/usr/bin
directory. You can confirm that MariaDB has been installed by using the MySQL client program. Issuing the command
mysql
should give you the MariaDB cursor.
```

## See Also

- [Installing MariaDB with yum](#)
- [Troubleshooting MariaDB Installs on RedHat/CentOS](#)
- [Checking MariaDB RPM Package Signatures](#)

## 2.1.2.1.5 Checking MariaDB RPM Package Signatures

MariaDB RPM packages since [MariaDB 5.1.55](#) are signed.

The key we use has an id of

1BB943DB

and the key fingerprint is:

```
1993 69E5 404B D5FC 7D2F E43B CBCB 082A 1BB9 43DB
```

To check the signature you first need to import the public part of the key like so:

```
gpg --keyserver hkp://pgp.mit.edu --recv-keys 1BB943DB
```

Next you need to let pgp know about the key like so:

```
gpg --export --armour 1BB943DB > mariadb-signing-key.asc
sudo rpm --import mariadb-signing-key.asc
```

You can check to see if the key was imported with:

```
rpm -qa gpg-pubkey*
```

Once the key is imported, you can check the signature of the MariaDB RPM files by running the something like the following in your download directory:

```
rpm --checksig $(find . -name '*.rpm')
```

The output of the above will look something like this (make sure gpg shows up on each OK line):

```
me@desktop:~$ rpm --checksig $(find . -name '*.rpm')
./kvm-rpm-centos5-amd64/rpms/MariaDB-test-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-server-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-client-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-shared-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-devel-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/rpms/MariaDB-debuginfo-5.1.55-98.el5.x86_64.rpm: (sha1) dsa sha1 md5 gpg OK
./kvm-rpm-centos5-amd64/srpms/MariaDB-5.1.55-98.el5.src.rpm: (sha1) dsa sha1 md5 gpg OK
```

## See Also

- [Installing MariaDB RPM Files](#)
- [Troubleshooting MariaDB Installs on RedHat/CentOS](#)

## 2.1.2.1.6 Troubleshooting MariaDB Installs on Red Hat/CentOS

The following article is about different issues people have encountered when installing MariaDB on Red Hat / CentOS.

It is highly recommended to [install with yum](#) where possible.

In Red Hat / CentOS it is also possible to install a [RPM](#) or a [tar ball](#). The RPM is the preferred version, except if you want to install many versions of MariaDB or install MariaDB in a non standard location.

## Replacing MySQL

If you removed an MySQL RPM to install MariaDB, note that the MySQL RPM on uninstall renames /etc/my.cnf to /etc/my.cnf.rpmsave.

After installing MariaDB you should do the following to restore your configuration options:

```
mv /etc/my.cnf.rpmsave /etc/my.cnf
```

## Unsupported configuration options

If you are using any of the following options in your /etc/my.cnf or other my.cnf file you should remove them. This is also true for MySQL 5.1 or newer:

```
skip-bdb
```

## See also

- [Installing with yum \(recommended\)](#)
- [Installing MariaDB RPM Files](#)
- [Checking MariaDB RPM Package Signatures](#)

## 2.1.2.1.7 MariaDB for DirectAdmin Using RPMs

If you are using DirectAdmin and you encounter any issues with [Installing MariaDB with YUM](#), then the directions below may help. The process is very straightforward.

**Note:** Installing with YUM is preferable to installing the MariaDB RPM packages manually, so only do this if you are having issues such as:

```
Starting httpd:  
httpd:  
Syntax error on line 18 of /etc/httpd/conf/httpd.conf:  
Syntax error on line 1 of /etc/httpd/conf/extrahttpd-phpmodules.conf:  
Cannot load /usr/lib/apache/libphp5.so into server:  
libmysqlclient.so.18: cannot open shared object file: No such file or directory
```

Or:

```
Starting httpd:  
httpd:  
Syntax error on line 18 of /etc/httpd/conf/httpd.conf:  
Syntax error on line 1 of /etc/httpd/conf/extrahttpd-phpmodules.conf:  
Cannot load /usr/lib/apache/libphp5.so into server:  
/usr/lib/apache/libphp5.so: undefined symbol: client_errors
```

## RPM Installation

To install the RPMs, there is a quick and easy guide to [Installing MariaDB with the RPM Tool](#). Follow the instructions there.

## Necessary Edits

We do not want DirectAdmin's custombuild to remove/overwrite our MariaDB installation whenever an update is performed. To rectify this, disable automatic MySQL installation.

Edit

```
/usr/local/directadmin/custombuild/options.conf
```

Change:

```
mysql_inst=yes
```

To:

```
mysql_inst=no
```

**Note:** When MariaDB is installed manually (i.e. not using YUM), updates are not automatic. You will need to update the RPMs yourself.

## 2.1.2.1.8 MariaDB Installation (Version 10.1.21) via RPMs on CentOS 7

Here are the detailed steps for installing MariaDB (version 10.1.21) via RPMs on CentOS 7.

The RPM's needed for the installation are all available on the MariaDB website and are given below:

- jemalloc-3.6.0-1.el7.x86\_64.rpm
- MariaDB-10.1.21-centos7-x86\_64-client.rpm
- MariaDB-10.1.21-centos7-x86\_64-compat.rpm
- galera-25.3.19-1.rhel7.el7.centos.x86\_64.rpm
- jemalloc-devel-3.6.0-1.el7.x86\_64.rpm
- MariaDB-10.1.21-centos7-x86\_64-common.rpm
- MariaDB-10.1.21-centos7-x86\_64-server.rpm

Step by step installation:

- 1) First install all of the dependencies needed. Its easy to do this via YUM packages: yum install rsync nmap lsof perl-DBI nc
- 2) rpm -ivh jemalloc-3.6.0-1.el7.x86\_64.rpm
- 3) rpm -ivh jemalloc-devel-3.6.0-1.el7.x86\_64.rpm
- 4) rpm -ivh MariaDB-10.1.21-centos7-x86\_64-common.rpm MariaDB-10.1.21-centos7-x86\_64-compat.rpm MariaDB-10.1.21-centos7-x86\_64-client.rpm galera-25.3.19-1.rhel7.el7.centos.x86\_64.rpm MariaDB-10.1.21-centos7-x86\_64-server.rpm

While installing MariaDB-10.1.21-centos7-x86\_64-common.rpm there might be a conflict with older MariaDB packages. we need to remove them and install the original rpm again.

Here is the error message for dependencies:

```
# rpm -ivh MariaDB-10.1.21-centos7-x86_64-common.rpm
warning: MariaDB-10.1.21-centos7-x86_64-common.rpm: Header V4 DSA/SHA1 Signature, key ID 1bb943db: NOKEY
error: Failed dependencies:
        mariadb-libs < 1:10.1.21-1.el7.centos conflicts with MariaDB-common-10.1.21-1.el7.centos.x86_64
```

Solution: search for this package:

```
# rpm -qa | grep mariadb-libs
mariadb-libs-5.5.52-1.el7.x86_64
```

Remove this package:

```
# rpm -ev --nodeps mariadb-libs-5.5.52-1.el7.x86_64
Preparing packages...
mariadb-libs-1:5.5.52-1.el7.x86_64
```

While installing the Galera package there might be a conflict in installation for a dependency package. Here is the error message:

```
[root@centos-2 /]# rpm -ivh galera-25.3.19-1.rhel7.el7.centos.x86_64.rpm
error: Failed dependencies:
        libboost_program_options.so.1.53.0()(64bit) is needed by galera-25.3.19-1.rhel7.el7.centos.x86_64

The dependencies for Galera package is: libboost_program_options.so.1.53.0
```

Solution:

```
yum install boost-devel.x86_64
```

Another warning message while installing Galera package is as shown below:

```
warning: galera-25.3.19-1.rhel7.el7.centos.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID 1bb943db: NOKEY
```

The solution for this is to import the key:

```
#rpm --import http://yum.mariadb.org/RPM-GPG-KEY-MariaDB
```

After step 4, the installation will be completed. The last step will be to run [mysql\\_secure\\_installation](#) to secure the production server by dis allowing remote login for root, creating root password and removing the test database.

- 5) mysql\_secure\_installation

## 2.1.2.1.9 Why Source RPMs (SRPMs) Aren't Packaged For Some Platforms

MariaDB source RPMs (SRPMs) are not packaged on all platforms for which MariaDB RPMs are packaged.

The reason is that MariaDB's build process relies heavily on

[cmake](#)

for a lot of things. In this specific case, MariaDB's build process relies on [CMake CPack Package Generators](#) to build RPMs. The specific package generator that it uses to build RPMs is called

[CPackRPM](#)

Support for source RPMs in

[CPackRPM](#)

became usable with MariaDB's build system starting from around [cmake 3.10](#). This means that we do not produce source RPMs on platforms where the installed

[cmake](#)

version is older than that.

See also [Building MariaDB from a Source RPM](#).

## 2.1.2.1.10 Building MariaDB from a Source RPM

For some distributions you can build MariaDB from a source RPM. (See also [Why Source RPMs \(SRPMs\) Aren't Packaged For Some Platforms](#)).

You can build it as follows:

### using dnf

On RHEL8 you might need to start with:

```
sudo dnf config-manager --set-enabled codeready-builder-beta-for-rhel-8-x86_64-rpms
```

Then, on all dnf distributions:

```
sudo dnf install rpm-build perl-generators
dnf download --source MariaDB
sudo dnf builddep MariaDB-*.src.rpm
rpmbuild --rebuild MariaDB-*.src.rpm
```

### using yum

```
sudo yum install rpm-build yum-utils
yumdownloader --source MariaDB
sudo yum-builddep MariaDB-*.src.rpm
rpmbuild --rebuild MariaDB-*.src.rpm
```

### using zypper

```
sudo zypper in rpm-build
sudo zypper si MariaDB
sudo rpmbuild -bb /usr/src/packages/SPECS/MariaDB.spec
```

Or (to avoid building as root):

```
sudo zypper in rpm-build
sudo zypper si -d MariaDB
zypper --pkg-cache-dir=`pwd` si --download-only MariaDB
rpmbuild --rebuild mariadb/srpms/MariaDB-*.src.rpm
```

## 2.1.2.2 Installing MariaDB .deb Files

## Contents

- 1. [Installing MariaDB with APT](#)
  - 1. [Adding the MariaDB APT repository](#)
    - 1. [Using the MariaDB Package Repository Setup Script](#)
  - 2. [Using the MariaDB Repository Configuration Tool](#)
    - 1. [Executing add-apt-repository](#)
    - 2. [Creating a Source List File](#)
    - 3. [Using Ubuntu Software Center](#)
    - 4. [Using Synaptic Package Manager](#)
  - 3. [Pinning the MariaDB Repository to a Specific Minor Release](#)
- 2. [Updating the MariaDB APT repository to a New Major Release](#)
  - 1. [Updating the Major Release with the MariaDB Package Repository Setup Script](#)
  - 2. [Updating the Major Release with the MariaDB Repository Configuration Tool](#)
    - 1. [Updating a Repository with add-apt-repository](#)
    - 2. [Updating a Source List File](#)
- 3. [Importing the MariaDB GPG Public Key](#)
- 4. [Installing MariaDB Packages with APT](#)
  - 1. [Installing the Most Common Packages with APT](#)
  - 2. [Installing MariaDB Server with APT](#)
  - 3. [Installing MariaDB Galera Cluster with APT](#)
  - 4. [Installing MariaDB Clients and Client Libraries with APT](#)
  - 5. [Installing Mariabackup with APT](#)
  - 6. [Installing Plugins with APT](#)
  - 7. [Installing Older Versions from the Repository](#)
- 2. [Installing MariaDB with dpkg](#)
- 3. [After Installation](#)
- 4. [Installation Issues](#)
  - 1. [Upgrading mariadb-server and mariadb-client packages](#)
  - 2. [Version Mismatch Between MariaDB and Ubuntu/Debian Repositories](#)
    - 1. [Pinning the MariaDB Repository](#)
    - 2. [Specifying Specific Versions to Install](#)
    - 3. [Holding Packages](#)
- 5. [Available DEB Packages](#)
  - 1. [Available DEB Packages in MariaDB 10.4](#)
  - 2. [Available DEB Packages in MariaDB 10.2 and MariaDB 10.3](#)
  - 3. [Available DEB Packages in MariaDB 10.1](#)
  - 4. [Available DEB Packages in MariaDB 5.5](#)
- 6. [Actions Performed by DEB Packages](#)
  - 1. [Users and Groups Created](#)
- 7. [See Also](#)

## Installing MariaDB with APT

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant

.deb

packages from MariaDB's repository using

apt

`aptitude`

, [Ubuntu Software Center](#) , [Synaptic Package Manager](#) , or another package manager.

This page walks you through the simple installation steps using

`apt`

## Adding the MariaDB APT repository

We currently have APT repositories for the following Linux distributions:

- Debian 9 (Jessie)
- Debian 10 (Buster)
- Debian Unstable (Sid)
- Ubuntu 18.04 LTS (Bionic)
- Ubuntu 20.04 LTS (Focal)
- Ubuntu 20.10 (Groovy)
- Ubuntu 21.04 (Hirsute)

## Using the MariaDB Package Repository Setup Script

If you want to install MariaDB with

`apt`

, then you can configure

`apt`

to install from MariaDB Corporation's MariaDB Package Repository by using the [MariaDB Package Repository setup script](#) .

MariaDB Corporation provides a MariaDB Package Repository for several Linux distributions that use

`apt`

to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, [clients and utilities](#) , [client libraries](#) , [plugins](#) , and [Mariabackup](#) . The MariaDB Package Repository setup script automatically configures your system to install packages from the MariaDB Package Repository.

To use the script, execute the following command:

```
curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
```

Note that this script also configures a repository for [MariaDB MaxScale](#) and a repository for MariaDB Tools, which currently only contains [Percona XtraBackup](#) and its dependencies.

See [MariaDB Package Repository Setup and Usage](#) for more information.

## Using the MariaDB Repository Configuration Tool

If you want to install MariaDB with

`apt`

, then you can configure

`apt`

to install from MariaDB Foundation's MariaDB Repository by using the [MariaDB Repository Configuration Tool](#) .

The MariaDB Foundation provides a MariaDB repository for several Linux distributions that use

`apt-get`

to manage packages. This repository contains software packages related to MariaDB Server, including the server itself, [clients and utilities](#) , [client libraries](#) , [plugins](#) , and [Mariabackup](#) . The MariaDB Repository Configuration Tool can easily generate the appropriate commands to add the repository for your distribution.

There are several ways to add the repository.

Executing add-apt-repository

One way to add an

`apt`

repository is by using the

`add-apt-repository`

command. This command will add the repository configuration to  
`/etc/apt/sources.list`

For example, if you wanted to use the repository to install [MariaDB 10.3](#) on Ubuntu 18.04 LTS (Bionic), then you could use the following commands to add

the MariaDB

apt  
repository:

```
sudo apt-get install software-properties-common
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el] http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.3/ubuntu bionic main'
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

## Creating a Source List File

Another way to add an

apt  
repository is by creating a [source list](#) file in  
`/etc/apt/sources.list.d/`

For example, if you wanted to use the repository to install [MariaDB 10.3](#) on Ubuntu 18.04 LTS (Bionic), then you could create the

MariaDB.list  
file in  
`/etc/apt/sources.list.d/`  
with the following contents to add the MariaDB  
apt  
repository:

```
# MariaDB 10.3 repository list - created 2019-01-27 09:50 UTC
# http://downloads.mariadb.org/mariadb/repositories/
deb [arch=amd64,arm64,ppc64el] http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.3/ubuntu bionic main
deb-src http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.3/ubuntu bionic main
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

## Using Ubuntu Software Center

Another way to add an

apt  
repository is by using [Ubuntu Software Center](#).

You can do this by going to the **Software Sources** window. This window can be opened either by navigating to **Edit > Software Sources** or by navigating to **System > Administration > Software Sources**.

Once the **Software Sources** window is open, go to the **Other Software** tab, and click the **Add** button. At that point, you can input the repository information provided by the [MariaDB Repository Configuration Tool](#).

See [here](#) for more information.

## Using Synaptic Package Manager

Another way to add an

apt  
repository is by using [Synaptic Package Manager](#).

You can do this by going to the **Software Sources** window. This window can be opened either by navigating to **System > Administrator > Software Sources** or by navigating to **Settings > Repositories**.

Once the **Software Sources** window is open, go to the **Other Software** tab, and click the **Add** button. At that point, you can input the repository information provided by the [MariaDB Repository Configuration Tool](#).

See [here](#) for more information.

## Pinning the MariaDB Repository to a Specific Minor Release

If you wish to pin the

apt  
repository to a specific minor release, or if you would like to downgrade to a specific minor release, then you can create a  
apt  
repository with the URL hard-coded to that specific minor release.

The MariaDB Foundation archives repositories of old minor releases at the following URL:

- <http://archive.mariadb.org/>

Archives are only of the distros and architectures supported at the time of release. For example [MariaDB 10.0.38](#) exists for Ubuntu

```
precise
,
trusty
,
xenial
,
wily
, and
yakkety
```

obtained looking in <https://archive.mariadb.org/mariadb-10.0.38/repo/ubuntu/dists>.

For example, if you wanted to pin your repository to [MariaDB 10.5.9](#) on Ubuntu 20.04 LTS (Focal), then you would have to first remove any existing MariaDB repository source list file from

```
/etc/apt/sources.list.d/
. And then you could use the following commands to add the MariaDB
apt-get
repository:
```

```
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el,s390x] http://archive.mariadb.org/mariadb-10.5.9/repo/ubuntu/ focal main'
```

Ensure you have the [signing key installed](#).

Ubuntu Xenial and older will need:

```
sudo apt-get install -y apt-transport-https
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

## Updating the MariaDB APT repository to a New Major Release

MariaDB's

```
apt
repository can be updated to a new major release. How this is done depends on how you originally configured the repository.
```

### Updating the Major Release with the MariaDB Package Repository Setup Script

If you configured

```
apt
to install from MariaDB Corporation's MariaDB Package Repository by using the MariaDB Package Repository setup script, then you can update the major release that the repository uses by running the script again.
```

### Updating the Major Release with the MariaDB Repository Configuration Tool

If you configured

```
apt
to install from MariaDB Foundation's MariaDB Repository by using the MariaDB Repository Configuration Tool, then you can update the major release in various ways, depending on how you originally added the repository.
```

Updating a Repository with add-apt-repository

If you added the

```
apt
repository by using the
add-apt-repository
```

command, then you can update the major release that the repository uses by using the the

```
add-apt-repository
```

command again.

First, look for the repository string for the old version in

```
/etc/apt/sources.list
```

And then, you can remove the repository for the old version by executing the

```
add-apt-repository
```

command and providing the

```
--remove
```

option. For example, if you wanted to remove a [MariaDB 10.2](#) repository, then you could do so by executing something like the following:

```
sudo add-apt-repository --remove 'deb [arch=amd64,arm64,ppc64el] http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.2/ubuntu b'.
```

After that, you can add the repository for the new version with the

```
add-apt-repository
```

command. For example, if you wanted to use the repository to install [MariaDB 10.3](#) on Ubuntu 18.04 LTS (Bionic), then you could use the following commands to add the MariaDB

```
apt
```

```
repository:
```

```
sudo apt-get install software-properties-common  
sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el] http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.3/ubuntu bionic mai'.
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

After that, the repository should refer to [MariaDB 10.3](#).

Updating a Source List File

If you added the

```
apt
```

```
repository by creating a source list file in
```

```
/etc/apt/sources.list.d/
```

, then you can update the major release that the repository uses by updating the source list file in-place. For example, if you wanted to change the repository from [MariaDB 10.2](#) to [MariaDB 10.3](#), and if the source list file was at

```
/etc/apt/sources.list.d/MariaDB.list  
, then you could execute the following:
```

```
sudo sed -i 's/10.2/10.3/' /etc/apt/sources.list.d/MariaDB.list
```

And then you would have to update the package cache by executing the following command:

```
sudo apt update
```

After that, the repository should refer to [MariaDB 10.3](#).

## Importing the MariaDB GPG Public Key

Before MariaDB can be installed, you also have to import the GPG public key that is used to verify the digital signatures of the packages in our repositories. This allows the

```
apt
```

```
utility to verify the integrity of the packages that it installs.
```

- Prior to Debian 9 (Stretch), and Debian Unstable (Sid), and Ubuntu 16.04 LTS (Xenial), the id of our GPG public key is 0xcbcb082a1bb943db. The full key fingerprint is:

```
1993 69E5 404B D5FC 7D2F E43B CBCB 082A 1BB9 43DB
```

The

```
apt-key
```

utility can be used to import this key. For example:

```
sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xcbcb082a1bb943db
```

- Starting with Debian 9 (Stretch) and Ubuntu 16.04 LTS (Xenial), the id of our GPG public key is 0xF1656F24C74CD1D8 . The full key fingerprint is:

```
177F 4010 FE56 CA33 3630 0305 F165 6F24 C74C D1D8
```

The

[apt-key](#)

utility can be used to import this key. For example:

```
sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8
```

Starting with Debian 9 (Stretch), the

[dirmngr](#)

package needs to be installed before the GPG public key can be imported. To install it, execute:

```
sudo apt install dirmngr
```

If you are unsure which GPG public key you need, then it is perfectly safe to import both keys.

The command used to import the GPG public key is the same on both Debian and Ubuntu. For example:

```
$ sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xcbcb082a1bb943db
Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --secret-keyring /tmp/tmp.ASyOPV87XC --trustdb-name /etc
gpg: requesting key 1BB943DB from hkp server keyserver.ubuntu.com
gpg: key 1BB943DB: "MariaDB Package Signing Key <package-signing-key@mariadb.org>" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:           imported: 1
```

Once the GPG public key is imported, you are ready to install packages from the repository.

## Installing MariaDB Packages with APT

After the

[apt](#)

repository is configured, you can install MariaDB by executing the

[apt-get](#)

command. The specific command that you would use would depend on which specific packages that you want to install.

### Installing the Most Common Packages with APT

To Install the most common packages, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

MariaDB starting with 10.4

In [MariaDB 10.4](#) and later, to Install the most common packages, execute the following command:

```
sudo apt-get install mariadb-server galera-4 mariadb-client libmariadb3 mariadb-backup mariadb-common
```

MariaDB 10.2 - 10.3

In [MariaDB 10.2](#) and [MariaDB 10.3](#) , to Install the most common packages, execute the following command:

```
sudo apt-get install mariadb-server galera mariadb-client libmariadb3 mariadb-backup mariadb-common
```

MariaDB until 10.1

In [MariaDB 10.1](#) and before, to Install the most common packages, execute the following command:

```
sudo apt-get install mariadb-server galera mariadb-client libmysqlclient18 mariadb-backup mariadb-common
```

## Installing MariaDB Server with APT

To Install MariaDB Server, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

Then, execute the following command:

```
sudo apt-get install mariadb-server
```

## Installing MariaDB Galera Cluster with APT

The process to install MariaDB Galera Cluster with the MariaDB

```
apt-get  
repository
```

is practically the same as installing standard MariaDB Server.

In [MariaDB 10.1](#) and later, Galera Cluster support has been included in the standard MariaDB Server packages, so you will need to install the  
`mariadb-server`  
package, as you normally would.

In [MariaDB 10.4](#) and later, you also need to install the

```
galera-4  
package
```

to obtain the [Galera 4 wsrep provider library](#).

In [MariaDB 10.3](#) and before, you also need to install the

```
galera-3  
package
```

to obtain the [Galera 3 wsrep provider library](#).

To install MariaDB Galera Cluster, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

### MariaDB starting with 10.4

In [MariaDB 10.4](#) and later, to install MariaDB Galera Cluster, you could execute the following command:

```
sudo apt-get install mariadb-server mariadb-client galera-4
```

### MariaDB until 10.3

In [MariaDB 10.3](#) and before, to install MariaDB Galera Cluster, you could execute the following command:

```
sudo apt-get install mariadb-server mariadb-client galera-3
```

MariaDB Galera Cluster also has a separate package that can be installed on arbitrator nodes. In [MariaDB 10.4](#) and later, the package is called

```
galera-arbitrator-4
```

In [MariaDB 10.3](#) and before, the package is called

```
galera-arbitrator-3
```

. This package should be installed on whatever node you want to serve as the arbitrator. It can either run on a separate server that is not acting as a cluster node, which is the recommended configuration, or it can run on a server that is also acting as an existing cluster node.

### MariaDB starting with 10.4

In [MariaDB 10.4](#) and later, to install the arbitrator package, you could execute the following command:

```
sudo apt-get install galera-arbitrator-4
```

### MariaDB until 10.3

In [MariaDB 10.3](#) and before, to install the arbitrator package, you could execute the following command:

```
sudo apt-get install galera-arbitrator-3
```

See [MariaDB Galera Cluster](#) for more information on MariaDB Galera Cluster.

## Installing MariaDB Clients and Client Libraries with APT

In [MariaDB 10.2](#) and later, [MariaDB Connector/C](#) has been included as the client library.

To Install the clients and client libraries, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

Then, in [MariaDB 10.2](#) and later, execute the following command:

```
sudo apt-get install mariadb-client libmariadb3
```

Or in [MariaDB 10.1](#) and before, execute the following command:

```
sudo apt-get install mariadb-client libmysqlclient18
```

## Installing Mariabackup with APT

To install [Mariabackup](#), first you would have to update the package cache by executing the following command:

```
sudo apt update
```

Then, execute the following command:

```
sudo apt-get install mariadb-backup
```

## Installing Plugins with APT

Some [plugins](#) may also need to be installed.

For example, to install the

```
cracklib_password_check
```

password validation plugin, first you would have to update the package cache by executing the following command:

```
sudo apt update
```

Then, execute the following command:

```
sudo apt-get install mariadb-cracklib-password-check
```

## Installing Older Versions from the Repository

The MariaDB

```
apt
```

repository contains the last few versions of MariaDB. To show what versions are available, use the

```
apt-cache
```

command:

```
sudo apt-cache showpkg mariadb-server
```

In the output you will see the available versions.

To install an older version of a package instead of the latest version we just need to specify the package name, an equal sign, and then the version number.

However, when installing an older version of a package, if

```
apt-get
```

has to install dependencies, then it will automatically choose to install the latest versions of those packages. To ensure that all MariaDB packages are on the same version in this scenario, it is necessary to specify them all. Therefore, to install [MariaDB 10.3.14](#) from this

```
apt
```

repository, we would do the following:

```
sudo apt-get install mariadb-server=10.3.14-1 mariadb-client=10.3.14-1 libmariadb3=10.3.14-1 mariadb-backup=10.3.14-1 mariadb-co
```

The rest of the install and setup process is as normal.

## Installing MariaDB with dpkg

While it is not recommended, it is possible to download and install the

```
.deb  
packages manually. However, it is generally recommended to use a package manager like  
apt-get
```

A tarball that contains the

```
.deb  
packages can be downloaded from the following URL:
```

- <https://downloads.mariadb.com>

For example, to install the [MariaDB 10.4.8](#)

```
.deb  
packages on Ubuntu 18.04 LTS (Bionic), you could execute the following:
```

```
sudo apt-get update  
sudo apt-get install libdbi-perl libdbd-mysql-perl psmisc libaio1 socat  
wget https://downloads.mariadb.com/MariaDB/mariadb-10.4.8/repo/ubuntu/mariadb-10.4.8-ubuntu-bionic-amd64-debs.tar  
tar -xvf mariadb-10.4.8-ubuntu-bionic-amd64-debs.tar  
cd mariadb-10.4.8-ubuntu-bionic-amd64-debs/  
sudo dpkg --install ./mariadb-common*.deb \  
./mysql-common*.deb \  
./mariadb-client*.deb \  
./libmariadb3*.deb \  
./libmysqlclient18*.deb  
sudo dpkg --install ./mariadb-server*.deb \  
./mariadb-backup*.deb \  
./galera-4*.deb
```

## After Installation

After the installation is complete, you can [start MariaDB](#).

If you are using [MariaDB Galera Cluster](#), then keep in mind that the first node will have to be [bootstrapped](#).

## Installation Issues

MariaDB starting with 5.5

### Upgrading

mariadb-server  
and  
mariadb-client  
packages

As noted in [MDEV-4266](#), the

```
mariadb-server  
and  
mariadb-client  
packages have a minor upgrade issue if you use '  
apt-get install mariadb-server  
' or '  
apt-get install mariadb-client  
' to upgrade them instead of the more common '  
apt-get upgrade  
'. This is because those two packages depend on  
mariadb-server-5.5  
and  
mariadb-client-5.5  
with no specific version of those packages. For example, if you have the  
mariadb-server
```

package installed, version 5.5.29 and you install version 5.5.30 of that package it will not automatically upgrade the mariadb-server-5.5 package to version 5.5.30 like you would expect because the 5.5.29 version of that package satisfies the dependency.

The

```
mariadb-server  
and  
mariadb-client  
packages are virtual packages, they only exist to require the installation of the  
mariadb-server-5.5  
and  
mariadb-client-5.5  
packages, respectively. MariaDB will function normally with a, for example, version 5.5.30 version of the  
mariadb-server  
package and a version 5.5.29 version of the  
mariadb-server-5.5  
package. No data is at risk. However, expected behavior is for '  
apt-get install mariadb-server  
' to upgrade everything to the latest version (if a new version is available), so this is definitely a bug.
```

A fix is planned for this bug in a future version of MariaDB. In the mean time, when upgrading MariaDB, use '

```
apt-get upgrade  
' or '  
apt-get install mariadb-server-5.5  
'
```

## MariaDB 5.1 - 5.5

### Version Mismatch Between MariaDB and Ubuntu/Debian Repositories

As mentioned [here](#) (and in [MDEV-4080](#) and [MDEV-3882](#)) sometimes APT will refuse to install MariaDB. Or, if MariaDB is already installed, suggest the removal of MariaDB to apply an upgrade to the

```
mysql-common  
or  
libmysqlclient  
packages. This happens whenever the version number of those two packages is higher in the distribution repositories than the versions in the MariaDB repositories. Most MariaDB packages have different names than their MySQL counterparts, but in order for upgrades from MySQL to MariaDB to be successful in APT, those two packages must be named the same. Because they have the same names, APT just checks the version numbers and tries to install what it considers to be the most recent.
```

It is rare for the version numbers of

```
mysql-common  
or  
libmysqlclient  
to be higher in the official Ubuntu or Debian repositories than they are in the MariaDB repositories, but it has happened. Whenever it has it has been because of critical bug fix releases for bugs that existed in the version of MySQL in the distribution repositories but which had already been fixed in the version of MariaDB in the MariaDB repositories.
```

If a situation as described above exists when you try to install MariaDB you will get an error like this:

```
The following packages have unmet dependencies:  
mariadb-server : Depends: mariadb-server-5.5 but it is not going to be installed  
E: Unable to correct problems, you have held broken packages.
```

There are three primary ways of fixing this issue:

1. [Pinning the MariaDB Repository](#)
2. [Specifying Specific Versions to Install](#)
3. [Holding Packages](#)

Click on the links above to jump to directions for each (or simply scroll down).

#### Pinning the MariaDB Repository

It is possible to *pin* the MariaDB repository used so the packages it provides will always have an higher priority over the ones from the system repositories.

This is done by creating a file with the '

```
.pref  
' extension under'  
/etc/apt/preferences.d/  
' with the following contents:
```

```
Package: *  
Pin: origin <mirror-domain>  
Pin-Priority: 1000
```

```
Replace '  
<mirror-domain>  
' with the domain name of the MariaDB mirror you use. For example '  
ftp.osuosl.org  
'
```

## Specifying Specific Versions to Install

A way to fix this is to specify the exact version of the two packages that you want to install. To do this, first determine the full version numbers of the affected packages. An easy way to do so is with

```
'apt-cache show'  
:  
  
apt-cache show mysql-common | grep Version  
apt-cache show libmysqlclient18 | grep Version
```

For each of the above you will be given a list of versions. The ones in the MariaDB repositories will have "mariadb" in the version strings and are the ones you want. With the version numbers in hand you will be able to install MariaDB by explicitly specifying the version numbers like so:

```
apt-get install mariadb-server-5.5 mariadb-client-5.5 \  
libmysqlclient18=<version-number> \  
mysql-common=<version-number>
```

Replace the two instances of

```
<version-number>  
in the example above with the actual version number of MariaDB that you want to install.
```

Even after having installed these specific packages versions, running '

```
apt-get dist-upgrade  
' will still try to upgrade the packages to the highest version available.
```

## Holding Packages

After MariaDB is installed, and as long as the version number issue exists, an '

```
apt-get dist-upgrade  
' will try to remove MariaDB in order to install the "upgraded"  
libmysqlclient  
and  
mysql-common  
packages. To prevent this from happening you can hold them so that apt doesn't try to upgrade them. To do so, open a terminal, become root with '  
sudo -s  
' , and then enter the following:
```

```
echo libmysqlclient18 hold | dpkg --set-selections  
echo mysql-common hold | dpkg --set-selections
```

The holds will prevent you from upgrading MariaDB, so when you want to remove the holds, open a terminal, become root with '

```
sudo -s  
' , and then enter the following:
```

```
echo libmysqlclient18 install | dpkg --set-selections  
echo mysql-common install | dpkg --set-selections
```

You will then be able to upgrade MariaDB as normal (e.g. with '

```
sudo apt-get update; sudo apt-get upgrade  
').
```

# Available DEB Packages

The available DEB packages depend on the specific MariaDB release series.

## Available DEB Packages in MariaDB 10.4

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) , the following DEBs are available:

Package Name	Description
--------------	-------------

galera-4	The WSREP provider for <a href="#">Galera 4</a> .
libmariadb3	Dynamic client libraries.
libmariadb-dev	Development headers and static libraries.
libmariadbclient18	Virtual package to satisfy external depends
libmysqlclient18	Virtual package to satisfy external depends
mariadb-backup	<a href="#">Mariabackup</a>
mariadb-client	Client tools like <code>mysql</code> CLI, <code>mysqldump</code> , and others.
mariadb-client-core	Core client tools
mariadb-common	Character set files and <code>/etc/my.cnf</code>
mariadb-plugin-connect	The <code>CONNECT</code> storage engine.
mariadb-plugin-cracklib-password-check	The <code>cracklib_password_check</code> password validation plugin.
mariadb-plugin-gssapi-client	The client-side component of the <code>gssapi</code> authentication plugin.
mariadb-plugin-gssapi-server	The server-side component of the <code>gssapi</code> authentication plugin.
mariadb-plugin-rocksdb	The <a href="#">MyRocks</a> storage engine.
mariadb-plugin-spider	The <code>SPIDER</code> storage engine.
mariadb-plugin-tokudb	The <a href="#">TokuDB</a> storage engine.

mariadb-server	The server and server tools, like <code>myisamchk</code> and <code>mysqlhotcopy</code> are here.
mariadb-server-core	The core server.
mariadb-test	<code>mysql-client-test</code> executable, and <b>mysql-test</b> framework with the tests.
mariadb-test-data	MariaDB database regression test suite - data files

## Available DEB Packages in MariaDB 10.2 and MariaDB 10.3

MariaDB starting with 10.2

In MariaDB 10.2 and MariaDB 10.3 , the following DEBs are available:

Package Name	Description
galera	The WSREP provider for Galera 3.
libmariadb3	Dynamic client libraries.
libmariadb-dev	Development headers and static libraries.
libmariadbclient18	Virtual package to satisfy external depends
libmysqlclient18	Virtual package to satisfy external depends
mariadb-backup	<b>Mariabackup</b>
mariadb-client	Client tools like <code>mysql</code> CLI, <code>mysqldump</code> , and others.
mariadb-client-core	Core client tools
mariadb-common	Character set files and <code>/etc/my.cnf</code>
mariadb-plugin-connect	The <code>CONNECT</code> storage engine.
mariadb-plugin-cracklib-password-check	The <code>cracklib_password_check</code> password validation plugin.

<code>mariadb-plugin-gssapi-client</code>	The client-side component of the <code>gssapi</code> authentication plugin.
<code>mariadb-plugin-gssapi-server</code>	The server-side component of the <code>gssapi</code> authentication plugin.
<code>mariadb-plugin-rocksdb</code>	The <code>MyRocks</code> storage engine.
<code>mariadb-plugin-spider</code>	The <code>SPIDER</code> storage engine.
<code>mariadb-plugin-tokudb</code>	The <code>TokuDB</code> storage engine.
<code>mariadb-server</code>	The server and server tools, like <code>myisamchk</code> and <code>mysqlhotcopy</code> are here.
<code>mariadb-server-core</code>	The core server.
<code>mariadb-test</code>	<code>mysql-client-test</code> executable, and <code>mysql-test</code> framework with the tests.
<code>mariadb-test-data</code>	MariaDB database regression test suite - data files

## Available DEB Packages in MariaDB 10.1

MariaDB starting with [10.1](#)

In [MariaDB 10.1](#), the following DEBs are available:

Package Name	Description
<code>galera</code>	The WSREP provider for <a href="#">Galera 3</a> .
<code>libmysqlclient18</code>	Dynamic client libraries.
<code>mariadb-backup</code>	<a href="#">Mariabackup</a>
<code>mariadb-client</code>	Client tools like <code>mysql</code> CLI, <code>mysqldump</code> , and others.
<code>mariadb-client-core</code>	Core client tools
<code>mariadb-common</code>	Character set files and <code>/etc/my.cnf</code>

mariadb-plugin-connect	The  CONNECT  storage engine.
mariadb-plugin-cracklib-password-check	The  cracklib_password_check  password validation plugin.
mariadb-plugin-gssapi-client	The client-side component of the  gssapi  authentication plugin.
mariadb-plugin-gssapi-server	The server-side component of the  gssapi  authentication plugin.
mariadb-plugin-spider	The  SPIDER  storage engine.
mariadb-plugin-tokudb	The TokuDB storage engine.
mariadb-server	The server and server tools, like myisamchk and mysqlhotcopy are here.
mariadb-server-core	The core server.
mariadb-test	mysql-client-test executable, and mysql-test framework with the tests.
mariadb-test-data	MariaDB database regression test suite - data files

## Available DEB Packages in MariaDB 5.5

MariaDB starting with 5.5

In MariaDB 5.5 , the following DEBs are available:

Package Name	Description
libmysqlclient18	Dynamic client libraries.
mariadb-client	Client tools like  mysql CLI, mysqldump , and others.
mariadb-client-core	Core client tools
mariadb-common	Character set files and  /etc/my.cnf

mariadb-server	The server and server tools, like <code>myisamchk</code> and <code>mysqlhotcopy</code> are here.
mariadb-server-core	The core server.
mariadb-test	<code>mysql-client-test</code> executable, and <b>mysql-test</b> framework with the tests.
mariadb-test-data	MariaDB database regression test suite - data files

## Actions Performed by DEB Packages

### Users and Groups Created

When the

`mariadb-server`  
DEB package is installed, it will create a user and group named  
`mysql`, if they do not already exist.

### See Also

- [Differences in MariaDB in Debian](#)
- [Installing MariaDB .deb Files with Ansible](#)

## 2.1.2.3 Installing MariaDB MSI Packages on Windows

MSI packages are available for both x86 (32 bit) and x64 (64 bit) processor architectures. We'll use screenshots from an x64 installation below (the 32 bit installer is very similar).

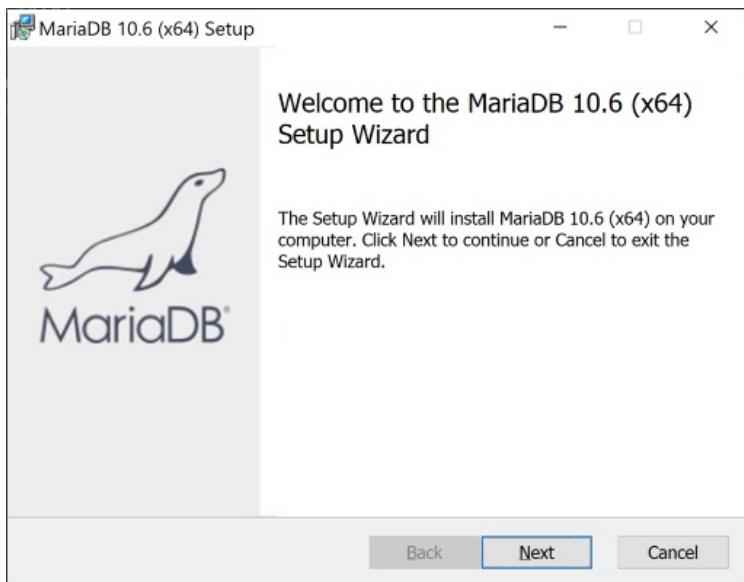
### Contents

1. Installation UI
  1. Welcome
  2. License Agreement
  3. Custom Setup
  4. Database Authentication/Security Related Properties
  5. Other Database Properties
  6. Ready to Install
  7. End
2. New Entries in Start Menu
3. Uninstall UI
4. Silent Installation
  1. Properties
  2. Features
  3. Silent Installation Examples
  4. Silent Uninstall
5. Installation Logs
6. Running 32 and 64 Bit Distributions on the Same Machine

### Installation UI

This is the typical mode of installation. To start the installer, just click on the `mariadb-<major>.<minor>.<patch>.msi`

### Welcome

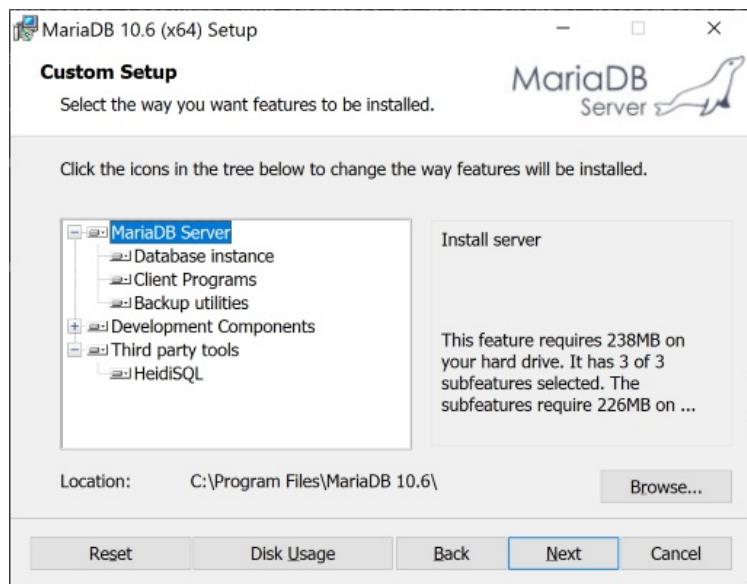


## License Agreement



Click on "I accept the terms"

## Custom Setup



Here, you can choose what features to install. By default, all features are installed with the exception of the debug symbols. If the "Database instance" feature is selected, the installer will create a database instance, by default running as a service. In this case the installer will present additional dialogs to control various database properties. Note that you do not necessarily have to create an instance at this stage. For example, if you already have MySQL or

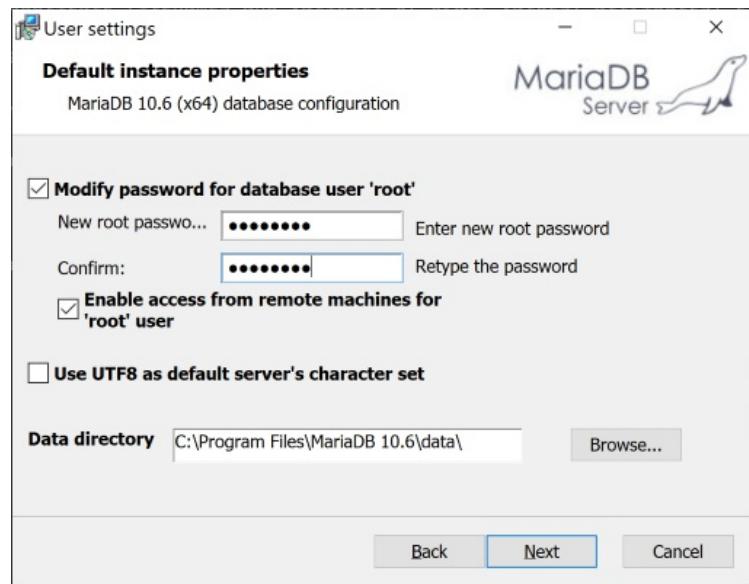
MariaDB databases running as services, you can just upgrade them during the installation. Also, you can create additional database instances after the installation, with the

`mysql_install_db.exe`

utility.

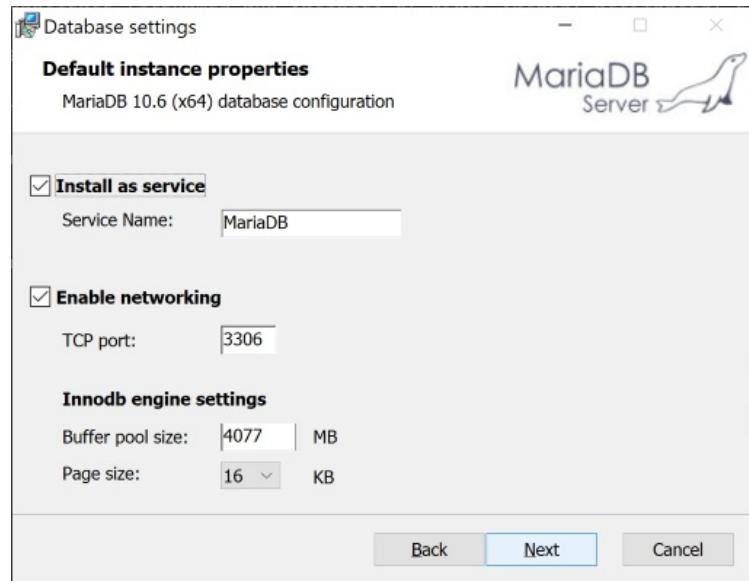
**NOTE :** By default, if you install a database instance, the data directory will be in the "data" folder under the installation root. To change the data directory location, select "Database instance" in the feature tree, and use the "Browse" button to point to another place.

## Database Authentication/Security Related Properties



This dialog is shown if you selected the "Database instance" feature. Here, you can set the password for the "root" database user and specify whether root can access database from remote machines. The "Create anonymous account" setting allows for anonymous (non-authenticated) users. It is off by default and it is not recommended to change this setting.

## Other Database Properties



- Install as service

Defines whether the database should be run as a service. If it should be run as a service, then it also defines the service name. It is recommended to run your database instance as a service as it greatly simplifies database management. In [MariaDB 10.4](#) and later, the default service name used by the MSI installer is "MariaDB". In 10.3 and before, the default service name used by the MSI installer is "MySQL". Note that the default service name for the

`--install`

and

--install-manual

options for  
mysqld.exe  
is "MySQL" in all versions of MariaDB.

- Enable Networking

Whether to enable TCP/IP (recommended) and which port MariaDB should listen to. If security is a concern, you can change the [bind-address](#) parameter post-installation to bind to only local addresses. If the "Enable networking" checkbox is deselected, the database will use named pipes for communication.

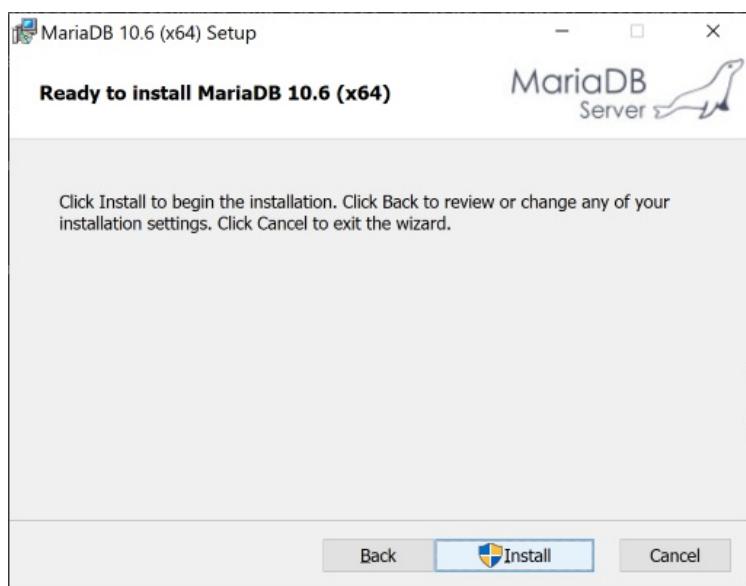
- Optimize for Transactions

If this checkbox is selected, the default storage engine is set to InnoDB (or XtraDB) and the

`sql_mode`  
parameter is set to "  
`NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES`

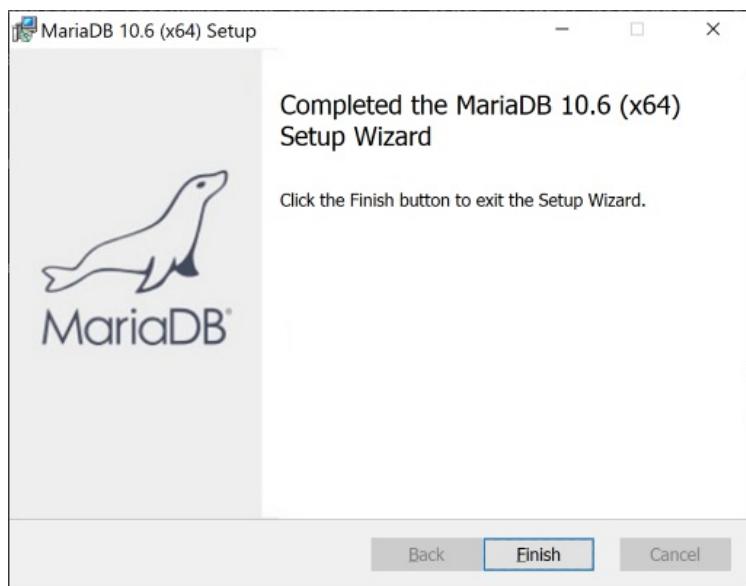
". You can also define the InnoDB/Xtradb buffer pool size. The default buffer pool size is 12.5% of RAM and depending on your requirements you can give InnoDB more (up to 70-80% RAM). 32 bit versions of MariaDB have restrictions on maximum buffer pool size, which is approximately 1GB, due to virtual address space limitations for 32bit processes.

## Ready to Install



At this point, all installation settings are collected. Click on the "Install" button.

## End

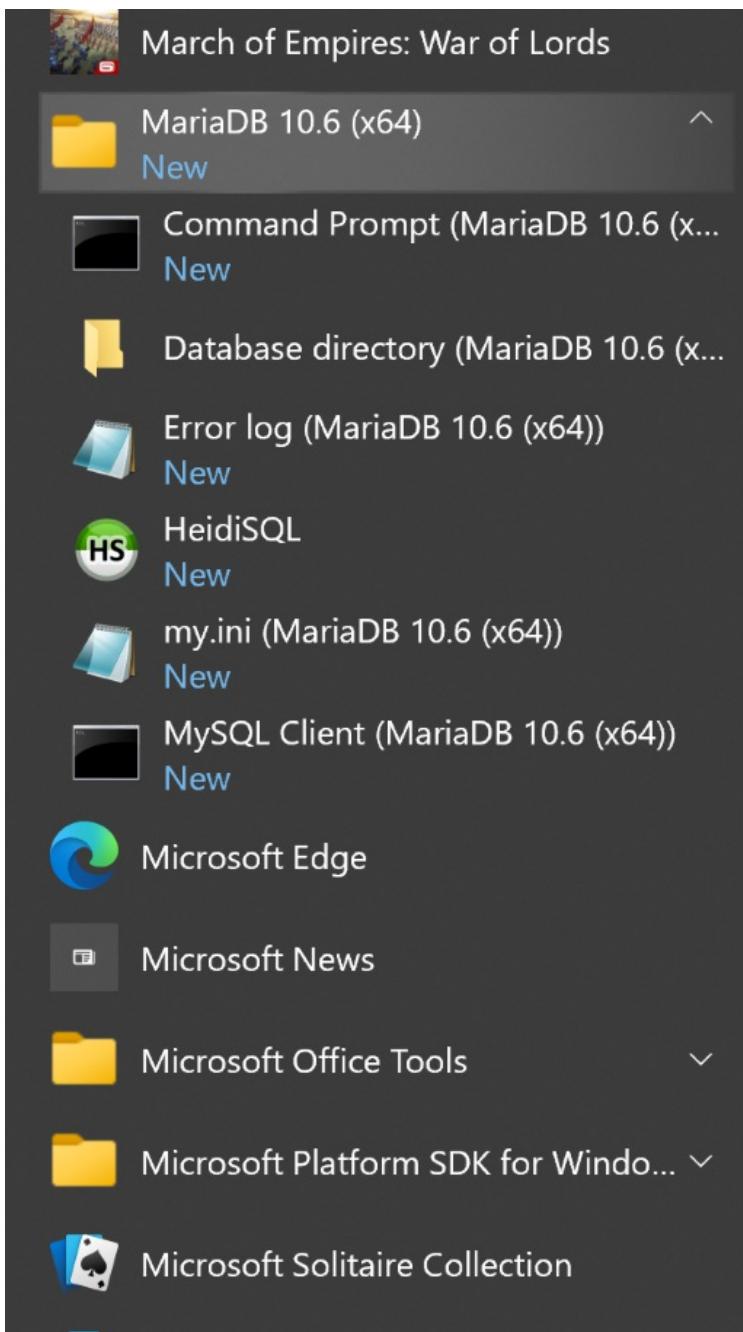


Installation is finished now. If you have upgradable instances of MariaDB/MySQL, running as services, this dialog will present a "Do you want to upgrade existing instances" checkbox (if selected, it launches the Upgrade Wizard post-installation).

If you installed a database instance as service, the service will be running already.

## New Entries in Start Menu

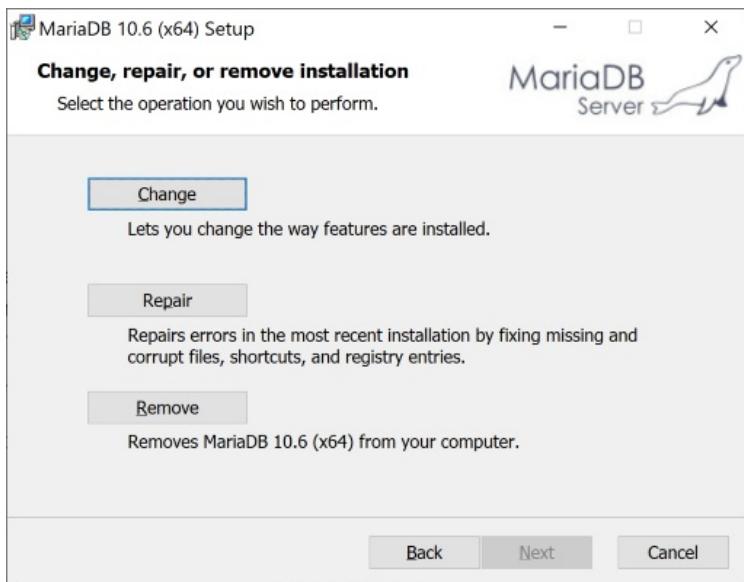
Installation will add some entries in the Start Menu:



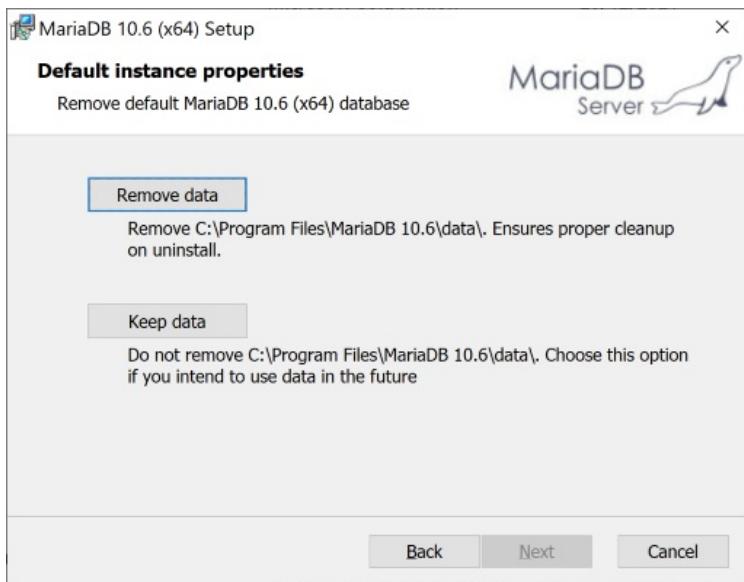
- MariaDB Client - Starts command line client mysql.exe
- Command Prompt - Starts a command prompt. Environment is set such that "bin" directory of the installation is included into PATH environment variable, i.e you can use this command prompt to issue MariaDB commands (mysqldadmin, mysql etc...)
- Database directory - Opens the data directory in Explorer.
- Error log - Opens the database error log in Notepad.
- my.ini - Opens the database configuration file my.ini in Notepad.
- Upgrade Wizard - Starts the Wizard to upgrade an existing MariaDB/MySQL database instance to this MariaDB version.

## Uninstall UI

In the Explorer applet "Programs and Features" (or "Add/Remove programs" on older Windows), find the entry for MariaDB, choose Uninstall/Change and click on the "Remove" button in the dialog below.



If you installed a database instance, you will need to decide if you want to remove or keep the data in the database directory.



## Silent Installation

The MSI installer supports silent installations as well. In its simplest form silent installation with all defaults can be performed from an elevated command prompt like this:

```
msiexec /i path-to-package.msi /qn
```

**Note:** the installation is silent due to msiexec.exe's /qn switch (no user interface), if you omit the switch, the installation will have the full UI.

## Properties

Silent installations also support installation properties (a property would correspond for example to checked/unchecked state of a checkbox in the UI, user password, etc). With properties the command line to install the MSI package would look like this:

```
msiexec /i path-to-package.msi [PROPERTY_1=VALUE_1 ... PROPERTY_N=VALUE_N] /qn
```

The MSI installer package requires property names to be all capitals and contain only English letters. By convention, for a boolean property, an empty value means "false" and a non-empty is "true".

MariaDB installation supports the following properties:

Property name	Default value	Description
INSTALLDIR	%ProgramFiles%\MariaDB<version>\	Installation root

PORT	3306	--port parameter for the server
ALLOWREMOTEROOTACCESS		Allow remote access for root user
BUFFERPOOLSIZEx	RAM/8	Bufferpoolsize for innodb
CLEANUPDATA	1	Remove the data directory (uninstall only)
DATADIR	INSTALLDIR\data	Location of the data directory
DEFAULTUSER		Allow anonymous users
PASSWORD		Password of the root user
SERVICENAME		Name of the Windows service. A service is not created if this value is empty.
SKIPNETWORKING		Skip networking
STDCONFIG	1	Corresponds to "optimize for transactions" in the GUI, default engine innodb, strict sql mode
UTF8		if set, adds character-set-server=utf8 to my.ini file (added in <a href="#">MariaDB 5.5.29</a> )
PAGESIZE	16K	page size for innodb

## Features

*Feature* is a Windows installer term for a unit of installation. Features can be selected and deselected in the UI in the feature tree in the "Custom Setup" dialog.

Silent installation supports adding features with the special property

```
ADDLOCAL=Feature_1,...,Feature_N
```

and removing features with

```
REMOVE=Feature_1,..., Feature_N
```

Features in the MariaDB installer:

Feature id	Installed by default?	Description
DBInstance	yes	Install database instance
Client	yes	Command line client programs
MYSQLSERVER	yes	Install server
SharedLibraries	yes	Install client shared library
DEVEL	yes	install C/C++ header files and client libraries
HeidiSQL	yes	Installs HeidiSQL

## Silent Installation Examples

All examples here require running as administrator (and elevated command line in Vista and later)

- Install default features, database instance as service, non-default datadir and port

```
msiexec /i path-to-package.msi SERVICENAME=MySQL DATADIR=C:\mariadb5.2\data PORT=3307 /qn
```

- Install service, add debug symbols, do not add development components (client libraries and headers)

```
msiexec /i path-to-package.msi SERVICENAME=MySQL ADDLOCAL=DEBUGSYMBOLS REMOVE=DEVEL /qn
```

## Silent Uninstall

To uninstall silently, use the

```
REMOVE=ALL
```

property with msiexec:

```
msiexec /i path-to-package.msi REMOVE=ALL /qn
```

To keep the data directory during an uninstall, you will need to pass an additional parameter:

```
msiexec /i path-to-package.msi REMOVE=ALL CLEANUPDATA="" /qn
```

# Installation Logs

If you encounter a bug in the installer, the installer logs should be used for diagnosis. Please attach verbose logs to the bug reports you create. To create a verbose installer log, start the installer from the command line with the

```
/1*v  
switch, like so:
```

```
msiexec.exe /i path-to-package.msi /1*v path-to-logfile.txt
```

## Running 32 and 64 Bit Distributions on the Same Machine

It is possible to install 32 and 64 bit packages on the same Windows x64.

Apart from testing, an example where this feature can be useful is a development scenario, where users want to run a 64 bit server and develop both 32 and 64 bit client components. In this case the full 64 bit package can be installed, including a database instance plus development-related features (headers and libraries) from the 32 bit package.

## 2.1.2.4 Installing MariaDB Server PKG packages on macOS

### Contents

1. [Installer behavior](#)
2. [launchd behavior](#)
3. [Uninstalling MariaDB Server](#)
4. [See Also](#)

MariaDB starting with 10.2.5

Starting with MariaDB 10.2.5, a .pkg installer for macOS is available from <http://mariadb.com/downloads>

The .pkg installer installs MariaDB Server and, optionally, a launchd service definition to allow the OS to automatically start and manage MariaDB Server.

## Installer behavior

The installer installs MariaDB Server in a subdirectory of /usr/local/mariadb.

You can place /usr/local/mariadb/server/bin in your PATH to always have access to executables distributed in the MariaDB Server package. The basedir of the MariaDB Server installation will depend on the version installed; for 10.2.6, it will be /usr/local/mariadb/mariadb-10.2.6-osx10.12-x86\_64, to which a symlink /usr/local/mariadb/server will be created.

The MariaDB Server binaries distributed in this package do not read from .cnf files in /etc/. Instead, put .cnf files in /usr/local/mariadb/etc/.

If there's an existing service listening on port 3306, the installer will create /usr/local/mariadb/etc/my.cnf with

```
skip-networking  
. You can edit this file to set an alternate port for MariaDB Server to listen on, or you can remove  
skip-networking  
if you have disabled the daemon that had bound port 3306.
```

MariaDB Server will place its socket file at /usr/local/mariadb/data/mariadb.sock and the client tools included in the package will by default look for a socket file at that location.

After installation, your MariaDB Server instance will have a "root" user account and a user account named after the username of the user who ran the installer. Both of these accounts use the [UNIX\\_SOCKET Authentication Plugin](#), which means no password is required to log in.

To log in to MariaDB as root, you will need to become root on the OS; the preferred mechanism for this is to use

```
sudo  
, like this:
```

```
sudo /usr/local/mariadb/server/bin/mariadb
```

If you want to be able to connect to MariaDB Server using TCP, you will need to create a new user and give it a password.

The daemon's executable is

```
mariadb  
, not  
mysqld  
. You should take this into account if you have scripts, tools, or workflows that look for processes named "mysqld". You can also invoke  
mysqld  
to start the server; edit /Library/LaunchDaemons/com.mariadb.server.plist if you want to do that permanently.
```

## launchd behavior

The installer can install a launchd service configuration. This is optional, but it is enabled by default.

The service definition causes launchd to automatically start MariaDB Server after installation and when the OS boots.

The launchd service configuration file installed by the installer is located at /Library/LaunchDaemons/com.mariadb.server.plist. The service definition file uses command-line arguments to

```
mariadb  
to control some of its behavior. These will override changes you make in option (.cnf) files.
```

To stop MariaDB Server using launchd, execute

```
sudo launchctl stop com.mariadb.server
```

To start MariaDB Server using launchd, execute

```
sudo launchctl start com.mariadb.server
```

## Uninstalling MariaDB Server

To remove MariaDB Server, follow these steps:

First, stop MariaDB Server and remove the files associated with launchd support, if you installed it (you'll need to do this as root or under sudo):

- ```
launchctl stop com.mariadb.server
```
- ```
launchctl unload /Library/LaunchDaemons/com.mariadb.server.plist
```
- ```
rm /Library/LaunchDaemons/com.mariadb.server.plist /usr/local/mariadb/server
```
- ```
pkgutil --forget com.mariadb.mariadb-server-launchd
```

Now, remove files associated with the MariaDB Server package:

- ```
rm -r /usr/local/mariadb/server/ /usr/local/mariadb/server
```
- If you also want to remove config files, remove  

```
/usr/local/mariadb/etc/
```
- If you want to remove the MariaDB Server data directory, remove  

```
/usr/local/mariadb/data/
```

You can also consult the output of

```
pkgutil --files com.mariadb.mariadb-server-files  
to see the specific list of files installed by the package.
```

After removing the files you wish to remove, tell pkgutil to forget the MariaDB Server package:

- ```
pkgutil --forget com.mariadb.mariadb-server-files
```

## See Also

- [MariaDB Server on macOS: Does it even make sense to try? \(video\)](#)

### 2.1.2.5 Installing MariaDB Binary Tarballs

#### Contents

1. [Ensure You Use the Correct my.cnf Files](#)
2. [Installing MariaDB as root in /usr/local/mysql](#)
3. [Installing MariaDB as Not root in Any Directory](#)
4. [Auto Start of mysqld](#)
5. [Post Installation](#)

MariaDB Binary tarballs are named following the pattern: mariadb-VERSION-OS.tar.gz. Be sure to [download](#) the correct version for your machine.

**Note:** Some binary tarballs are marked '(GLIBC\_2.14)' or '(requires GLIBC\_2.14+)'. These binaries are built the same as the others, but on a newer build host, and they require GLIBC 2.14 or higher. Use the other binaries for machines with older versions of GLIBC installed. Run

```
ldd --version  
to see which version is running on your distribution.
```

Others are marked 'systemd', which are for systems with

```
systemd  
and GLIBC 2.19 or higher.
```

To install the [binaries](#), unpack the distribution into the directory of your choice and run the

```
mysql_install_db
```

script.

In the example below we install MariaDB in the

```
/usr/local/mysql  
directory (this is the default location for MariaDB for many platforms). However any other directory should work too.
```

We install the binary with a symlink to the original name. This is done so that you can easily change MariaDB versions just by moving the symlink to point to another directory.

**NOTE:** For MariaDB 5.1.32 *only* the line "

```
./scripts/mysql_install_db --user=mysql  
" should be changed to "  
./bin/mysql_install_db --user=mysql  
"
```

## Ensure You Use the Correct my.cnf Files

MariaDB searches for the configuration files '

```
/etc/my.cnf  
' (on some systems '  
/etc/mysql/my.cnf  
) and '  
~/.my.cnf  
' If you have an old  
my.cnf  
file (maybe from a system installation of MariaDB or MySQL) you need to take care that you don't accidentally use the old one with your new  
binary .tar installation.
```

The normal solution for this is to ignore the

```
.my.cnf  
file in  
/etc  
when you use the programs in the tar file.
```

This is done by [creating your own .my.cnf file](#) in your home directory and telling

```
mysql_install_db  
, mysqld_safe and possibly mysql (the command-line client utility) to only use this one with the option '  
--defaults-file=~/.my.cnf  
' Note that this has to be first option for the above commands!
```

## Installing MariaDB as root in /usr/local/mysql

If you have root access to the system, you probably want to install MariaDB under the user and group 'mysql' (to keep compatibility with MySQL installations):

```
groupadd mysql  
useradd -g mysql mysql  
cd /usr/local  
tar -zxvf /path-to/mariadb-VERSION-OS.tar.gz  
ln -s mariadb-VERSION-OS mysql  
cd mysql  
./scripts/mysql_install_db --user=mysql  
chown -R root .  
chown -R mysql data
```

The symlinking with

```
ln -s  
is recommended as it makes it easy to install many MariaDB version at the same time (for easy testing, upgrading, downgrading etc).
```

If you are installing MariaDB to replace MySQL, then you can leave out the call to

```
mysql_install_db  
. Instead shut down MySQL. MariaDB should find the path to the data directory from your old  
/etc/my.cnf  
file (path may vary depending on your system).
```

To start mysqld you should now do:

```
./bin/mysqld_safe --user=mysql &  
or  
.bin/mysqld_safe --defaults-file=~/my.cnf --user=mysql &
```

To test connection, modify your \$PATH so you can invoke client such as [mysql](#), [mysqldump](#), etc.

```
export PATH=$PATH:/usr/local/mysql/bin/
```

You may want to modify your .bashrc or .bash\_profile to make it permanent.

## Installing MariaDB as Not root in Any Directory

Below, change /usr/local to the directory of your choice.

```
cd /usr/local  
gunzip < /path-to/mariadb-VERSION-OS.tar.gz | tar xf -  
ln -s mariadb-VERSION-OS mysql  
cd mysql  
.scripts/mysql_install_db --defaults-file=~/my.cnf
```

If you have problems with the above gunzip command line, you can instead, if you have gnu tar, do:

```
tar xfz /path-to/mariadb-VERSION-OS.tar.gz
```

To start mysqld you should now do:

```
./bin/mysqld_safe --defaults-file=~/my.cnf &
```

## Auto Start of mysqld

You can get mysqld (the MariaDB server) to autostart by copying the file

```
mysql.server  
file to the right place.
```

```
cp support-files/mysql.server /etc/init.d/mysql.server
```

The exact place depends on your system. The

```
mysql.server  
file contains instructions of how to use and fine tune it.
```

For systemd installation the mariadb.service file will need to be copied from the support-files/systemd folder to the /usr/lib/systemd/system/ folder.

```
cp support-files/systemd/mariadb.service /usr/lib/systemd/system/mariadb.service
```

Note that by default the /usr/ directory is write protected by systemd though, so when having the data directory in /usr/local/mysql/data as per the instructions above you also need to make that directory writable. You can do so by adding an extra service include file:

```
mkdir /etc/systemd/system/mariadb.service.d/  
  
cat > /etc/systemd/system/mariadb.service.d/datadir.conf <<EOF  
[Service]  
ReadWritePaths=/usr/local/mysql/data  
EOF  
  
systemctl daemon-reload
```

After this you can start and stop the service using

```
systemctl start mariadb.service
```

and

```
systemctl stop mariadb.service
```

respectively.

Please refer to the [systemd](#) page for further information.

## Post Installation

After this, remember to set proper passwords for all accounts accessible from untrusted sources, to avoid exposing the host to security risks!

Also consider using the [mysql.server](#) to [start MariaDB automatically](#) when your system boots.

On systems using systemd you can instead enable automatic startup during system boot with

```
systemctl enable mariadb.service
```

instead.

Our MariaDB binaries are similar to the Generic binaries available for the MySQL binary distribution. So for more options on using these binaries, the MySQL 5.5 manual entry on [installing generic binaries](#) can be consulted.

For details on the exact steps used to build the binaries, see the [compiling MariaDB section](#) of the KB.

## 2.1.2.6 Installing MariaDB Server on macOS Using Homebrew

### Contents

1. [Upgrading MariaDB](#)
2. [Building MariaDB Server from source](#)
3. [Other resources](#)

MariaDB Server is available for installation on macOS (formerly Mac OS X) via the [Homebrew](#) package manager.

MariaDB Server is available as a Homebrew "bottle", a pre-compiled package. This means you can install it without having to build from source yourself. This saves time.

After installing Homebrew, MariaDB Server can be installed with this command:

```
brew install mariadb
```

After installation, start MariaDB Server:

```
mysql.server start
```

To auto-start MariaDB Server, use Homebrew's services functionality, which configures auto-start with the launchctl utility from [launchd](#) :

```
brew services start mariadb
```

After MariaDB Server is started, you can log in as your user:

```
mysql
```

Or log in as root:

```
sudo mysql -u root
```

## Upgrading MariaDB

First you may need to update your brew installation:

```
brew update
```

Then, to upgrade MariaDB Server:

```
brew upgrade mariadb
```

## Building MariaDB Server from source

In addition to the "bottled" MariaDB Server package available from Homebrew, you can use Homebrew to build MariaDB from source. This is useful if you want to use a different version of the server or enable some different capabilities that are not included in the bottle package.

Two components not included in the bottle package (as of MariaDB Server 10.1.19) are the CONNECT and OQGRAPH engines, because they have non-standard dependencies. To build MariaDB Server with these engines, you must first install

```
boost  
and  
judy
```

. As of December 2016, judy is in the Homebrew "boneyard", but the old formula still works on macOS Sierra. Follow these steps to install the dependencies and build the server:

```
brew install boost homebrew/boneyard/judy  
brew install mariadb --build-from-source
```

You can also use Homebrew to build and install a pre-release version of MariaDB Server (for example MariaDB Server 10.2, when the highest GA version is MariaDB Server 10.1). Use this command to build and install a "development" version of MariaDB Server:

```
brew install mariadb --devel
```

## Other resources

- [mariadb.rb on github](#)
- [Terin Stock \(terinjokes\)](#) who is the packager for Homebrew
- [MariaDB Server on macOS: Does it even make sense to try?](#) (video)

## 2.1.2.7 Installing MariaDB Windows ZIP Packages

Getting started with ZIP packages on Windows is very straightforward - this distribution includes pre-built database files that can be used right after unpacking the ZIP.

You can run mysqld.exe from the command prompt like this:

```
cd <directory-where-zip-was-unpacked>  
bin\mysqld --console
```

However, running MariaDB from the command line in a user session like this covers only the simplest testing scenarios.

For any serious purpose, use [mysql\\_install\\_db.exe](#) to create database instances - it is more secure (disables anonymous user by default, allows setting root password during database creation), and more convenient (can create Windows service, supports user-specified data directory).

Note : Starting with [MariaDB 10.4.3](#), a pre-built data directory is no longer provided, and users need to run [mysql\\_install\\_db.exe](#) to create a data directory.

For older MariaDB releases, you can follow the instructions in [MySQL documentation on ZIP distributions](#) - this describes the manual process for registering a service and securing the database.

## 2.1.2.8

### 2.1.2.8.1 Get, Build and Test Latest MariaDB the Lazy Way

The intention of this documentation is show all the steps of getting, building and testing the latest MariaDB server (10.5 at time of writing) from GitHub. Each stage links to the full documentation for that step if you need to find out more.

#### Install all tools needed to build MariaDB

##### OpenSuse

```
sudo zypper install git gcc gcc-c++ make bison ncurses ncurses-devel zlib-devel libevent-devel cmake openssl
```

##### Debian

```
apt install -y build-essential bison  
apt build-dep mariadb-server
```

## Set Up git

Fetch and checkout the MariaDB source to a subdirectory of the current directory

```
git clone https://github.com/MariaDB/server.git mariadb  
cd mariadb  
git checkout 10.5
```

## Build It

The following command builds a server the same way that is used for building releases. Use

```
cmake . -DCMAKE_BUILD_TYPE=Debug  
to build for debugging.
```

```
cmake . -DBUILD_CONFIG=mysql_release && make -j8
```

## Check the Server (If You Want To)

```
cd mysql-test  
mtr --parallel=8 --force
```

## Install the Default Databases

```
./scripts/mariadb-install-db --srcdir=.
```

(Older MariaDB version use [mysql\\_install\\_db](#) )

## Install the Server (If needed)

You can also [run and test mariadb directly from the build directory](#) , in which case you can skip the rest of the steps below.

```
make install
```

## Start the Server

Start the server in it's own terminal window for testing. Note that the directory depends on your system!

```
/usr/sbin/mysqld
```

## 2.1.2.8.2 MariaDB Source Code

### Checking out the Source with Git

The MariaDB source is hosted on GitHub: <https://github.com/MariaDB/server>

If you want a source tarball for a specific released MariaDB version, you can find it at <http://downloads.mariadb.org> .

At any given time, developers will be working on their own branches locally or on GitHub, with the main MariaDB branches receiving pushes less often.

The main MariaDB branches are:

- [10.6](#)
- [10.5](#)
- [10.4](#)
- [10.3](#)
- [10.2](#)

Source repositories for the MariaDB Connectors are:

- [MariaDB Connector/C](#)
- [MariaDB Connector/J](#)
- [MariaDB Connector/Node.js](#)
- [MariaDB Connector/ODBC](#)
- [MariaDB Connector/Python](#)

See the [Using git](#) page for instructions on how to use git to check out the source code and switch between the various branches.

## 2.1.2.8.3 Build Environment Setup for Linux

### Contents

1. [Required Tools](#)
2. [See Also](#)

## Required Tools

The following is a list of tools that are required for building MariaDB on Linux and Mac OS X. Most, if not all, of these will exist as packages in your distribution's package repositories, so check there first. See [Building MariaDB on Ubuntu](#), [Building MariaDB on CentOS](#), and [Building MariaDB on Gentoo](#) pages for specific requirements for those platforms.

- [git](#)
- [gunzip](#)
- [GNU tar](#)
- [gcc/g++ 4.8.5 or later, recommend above 9 or clang/clang++](#)
- [GNU make 3.75 or later or Ninja](#)
- [bison \(3.0\)](#)
- [libncurses](#)
- [zlib-dev](#)
- [libevent-dev](#)
- [cmake above 2.8.7 though preferably above 3.3](#)
- [gnutls or openssl](#)
- [jemalloc \(optional\)](#)
- [valgrind \(only needed if running mysql-test-run --valgrind \)](#)
- [libcurl \(only needed if you want to use the S3 storage engine \)](#)

You can install these programs individually through your package manager.

In addition, some package managers support the use a build dependency command. When using this command, the package manager retrieves a list of build dependencies and install them for you, making it much easier to get started on the compile. The actual option varies, depending on the distribution you use.

On Ubuntu and Debian you can use the

```
build-dep  
command.
```

```
# apt build-dep mariadb-server
```

Fedora uses the  
`builddep`  
command with DNF.

```
# dnf builddep mariadb-server
```

CentOS has a separate utility

```
yum-builddep  
, which is part of the  
yum-utils  
package. This works like the DNF  
builddep  
command.
```

```
# yum install yum-utils  
# yum-builddep mariadb-server
```

With openSUSE and SUSE, you can use the source-install command.

```
# zypper source-install -d mariadb
```

Each of these commands works off of the release of MariaDB provided in the official software repositories of the given distribution. In some instances and especially in older versions of Linux, MariaDB may not be available in the official repositories. In these cases you can use the MariaDB repositories as an alternative.

Bear in mind, the release of MariaDB provided by your distribution may not be the same as the version you are trying to install. Additionally, the package managers don't always retrieve all of the packages you need to compile MariaDB. There may be some missed or unlisted in the process. When this is the case, CMake fails during checks with an error message telling you what's missing.

Note: On Debian-based distributions, you may receive a "You must put some 'source' URLs in your sources.list" error. To avoid this, ensure that /etc/apt/sources.list contains the source repositories.

For example, for Debian buster:

```
deb http://ftp.debian.org/debian buster main contrib  
deb http://security.debian.org buster/updates main contrib  
deb-src http://ftp.debian.org/debian buster main contrib  
deb-src http://security.debian.org buster/updates main contrib
```

Refer to the documentation for your Linux distribution for how to do this on your system.

After editing the sources.list, do:

```
sudo apt update
```

...and then the above mentioned

```
build-dep  
command.
```

Note: On openSUSE the source package repository may be disabled. The following command will enable it:

```
sudo zypper mr -er repo-source
```

After enabling it, you will be able to run the zypper command to install the build dependencies.

You should now have your build environment set up and can proceed to [Getting the MariaDB Source Code](#) and then using the [Generic Build Instructions](#) to build Mariadb (or following the steps for your Linux distribution or [Creating a MariaDB Binary Tarball](#) ).

## See Also

- [Installing Galera from source](#)

### 2.1.2.8.4 Generic Build Instructions

#### Contents

1. [Using cmake](#)
2. [Using BUILD Scripts](#)
3. [Starting MariaDB for the First Time](#)
4. [Testing MariaDB](#)
5. [Increasing Version Number or Tagging a Version](#)
6. [Non-ascii Symbols](#)
7. [Post-install Tasks](#)

The instructions on this page will help you compile [MariaDB](#) from source. Links to more complete instructions for specific platforms can be found on the [source](#) page.

First, [get a copy of the MariaDB source](#).

Next, [prepare your system to be able to compile the source](#).

If you don't want to run MariaDB as yourself, then you should create a

```
mysql  
user. The example below uses this user.
```

## Using cmake

[MariaDB 5.5](#) and above is compiled using [cmake](#).

It is recommended to create a build directory **beside** your source directory

```
mkdir build-mariadb  
cd build-mariadb
```

**NOTE** If you have built MariaDB in the past and have recently updated the repository, you should perform a complete cleanup of old artifacts (such as cmake configured files). In the base repository run:

```
git clean -xffd && git submodule foreach --recursive git clean -xffd
```

You can configure your build simply by running [cmake](#) without any special options, like

```
cmake .. /server
```

where

server  
is where you installed MariaDB. If you are building in the source directory, just omit  
.. /server

If you want it to be configured exactly as a normal MariaDB server release is built, use

```
cmake .. /server -DBUILD_CONFIG=mysql_release
```

This will configure the build to generate binary tarballs similar to release tarballs from downloads.mariadb.org. Unfortunately this doesn't work on old platforms, like OpenSuse Leap 15.0, because MariaDB binary tarballs are built to minimize external dependencies, and that needs static libraries that might not be provided by the platform by default, and would need to be installed manually.

To do a build suitable for debugging use:

```
cmake .. /server -DCMAKE_BUILD_TYPE=Debug
```

By default, MariaDB is compiled with the

-Werror  
flag, which causes compiling to abort if there is a compiler warning. You can disable that by configuring with  
-DMYSQL\_MAINTAINER\_MODE=OFF

```
cmake .. /server -DCMAKE_BUILD_TYPE=Debug -DMYSQL_MAINTAINER_MODE=OFF .
```

All *cmake* configuration options for MariaDB can be displayed with:

```
cmake .. /server -LH
```

To build and install MariaDB after running *cmake* use

```
cmake --build .  
sudo cmake --install .
```

If the commands above fail, you can enable more compilation information by doing:

```
cmake --build . --verbose
```

If you want to generate a binary tarball, run

```
cpack
```

## Using BUILD Scripts

There are also

BUILD

scripts for the most common systems for those that doesn't want to dig into *cmake* options. These are optimized for in source builds.

The scripts are of type 'compile-#cpu#-how\_to\_build'. Some common scripts-are

Script	Description
compile-pentium64	Compile an optimized binary optimized for 64 bit pentium (works also for amd64)
compile-pentium-debug	Compile a debug binary optimized for 64 bit pentium
compile-pentium-valgrind-max	Compile a debug binary that can be used with <a href="#">valgrind</a> to find wrong memory accesses and memory leaks. Should be used if one want's to run the <code>mysql-test-run</code> test suite with the <code>--valgrind</code> option

Some common suffixes used for the scripts:

Suffix	Description
--------	-------------

32	Compile for 32 bit cpu's
64	Compile for 64 bit cpu's
-max	Enable (almost) all features and plugins that MariaDB supports
-gprof	binary is compiled with profiling (gcc --pg)
-gcov	binary is compiled with code coverage (gcc -fprofile-arcs -ftest-coverage)
-valgrind	The binary is compiled for debugging and optimized to be used with <a href="#">valgrind</a> .
-debug	The binary is compiled with all symbols (gcc -g) and the <a href="#">DBUG</a> log system is enabled.

All

BUILD

scripts support the following options:

Suffix	Description
-h, --help	Show this help message.
-n, --just-print	Don't actually run any commands; just print them.
-c, --just-configure	Stop after running configure. Combined with --just-print shows configure options.
--extra-configs=xxx	Add this to configure options
--extra-flags=xxx	Add this C and CXX flags
--extra-cflags=xxx	Add this to C flags
--extra-cxxflags=xxx	Add this to CXX flags
--verbose	Print out full compile lines
--with-debug=full	Build with full debug(no optimizations, keep call stack).

A typical compilation used by a developer would be:

```
shell> ./BUILD/compile-pentium64-debug
```

This configures the source for debugging and runs make. The server binary will be

```
sql/mariadb  
or  
sql/mysql
```

## Starting MariaDB for the First Time

After installing MariaDB (using

```
sudo make install  
)
```

, but prior to starting MariaDB for the first time, one should:

1. ensure the directory where you installed MariaDB is owned by the mysql user (if the user doesn't exist, you'll need to create it)
2. run the

```
mysql_install_db  
script to generate the needed system tables
```

Here is an example:

```
# The following assumes that the 'mysql' user exists and that we installed MariaDB  
# in /usr/local/mysql  
chown -R mysql /usr/local/mysql/  
cd /usr/local/mysql/  
scripts/mysql_install_db --user=mysql  
/usr/local/mysql/bin/mysqld_safe --user=mysql &
```

## Testing MariaDB

If you want to test your compiled MariaDB, you can do either of:

Run unit tests:

```
cmake --build . --target test
```

Or run mtr tests:

```
mysql-test/mysql-test-run --force
```

Each of the above are run from the build directory. There is no need to '

```
make install  
/  
cmake --install .  
' MariaDB prior to running them.
```

**NOTE:** If you are doing more extensive testing or debugging of MariaDB (like with real application data and workloads) you may want to start and run MariaDB directly from the source directory instead of installing it with '

```
sudo make install  
' If so, see Running MariaDB from the Source Directory.
```

## Increasing Version Number or Tagging a Version

If you have made code changes and want to increase the version number or tag our version with a specific tag you can do this by editing the

```
VERSION  
file. Tags are shown when running the '  
mysqld --version  
' command.
```

## Non-ascii Symbols

MariaDB builds with

```
readline  
; using an alternative such as  
Editline  
may result in problems with non-ascii symbols.
```

## Post-install Tasks

- [Installing system tables \(mysql\\_install\\_db\)](#)
- [Starting and Stopping MariaDB Automatically](#)

### 2.1.2.8.5 Compiling MariaDB with Extra Modules/Options

#### 2.1.2.8.5.1 Compiling MariaDB with TCMalloc

[TCMalloc](#) is a malloc replacement library optimized for multi-threaded usage. It also features a built-in heap debugger and profiler.

To build [MariaDB 5.5](#) with

```
TCMalloc  
, you need to use the following command
```

```
cmake -DCMAKE_EXE_LINKER_FLAGS='`ltcmalloc` -DWITH_SAFEMALLOC=OFF'
```

Many other malloc replacement libraries (as well as heap debuggers and profilers) can be used with MariaDB in a similar fashion.

You can also start a standard MariaDB server with

TCmalloc  
with:

```
/usr/sbin/mysqld_safe --malloc-lib=tcmalloc
```

## See Also

- [Debugging a running server on Linux](#)

## 2.1.2.8.5.2 Compiling MariaDB with Vanilla XtraDB

Sometimes, one needs to have MariaDB compiled with Vanilla XtraDB. This page describes the process to do this. The process is rather crude, as my goal was just a once-off compile for testing (that is, not packaging or shipping) purposes.

The process is applicable to [MariaDB 5.3.4](#) and XtraDB from Percona Server 5.1.61.

```
wget http://s.petrunia.net/scratch/make-vanilla-xtradb-work-with-mariadb.diff
bzr branch /path/to/mariadb-5.3 5.3-vanilla-xtradb-r3
cd 5.3-vanilla-xtradb-r3/storage/
tar czf innodb_plugin.tgz innodb_plugin/
rm -rf innodb_plugin
tar czf xtradb.tgz xtradb/
rm -rf xtradb
cd ../../
tar zxvf ~/Percona-Server-5.1.61.tar.gz
cp -r Percona-Server-5.1.61/storage/innodb_plugin 5.3-vanilla-xtradb-r3/storage/
patch -p1 -d 5.3-vanilla-xtradb-r3/storage/innodb_plugin/ < make-vanilla-xtradb-work-with-mariadb.diff
cd 5.3-vanilla-xtradb-r3/
BUILD/autorun.sh
./configure --with-plugin-innodb_plugin
make
```

## 2.1.2.8.5.3 Specifying Which Plugins to Build

By default all plugins are enabled and built as dynamic

.so  
(or  
.dll  
) modules. If a plugin does not support dynamic builds, it is not built at all.

### MariaDB 5.5 - 10.0

To specify that a specific plugin should be enabled and built statically into the server executable, use the

-DWITH\_xxx=1  
cmake option (where  
xxx  
is the plugin name). Of course, static linking only works if the plugin itself supports it — some plugins can only be built as dynamic modules.  
A plugin will automatically be skipped by cmake if it cannot be built; for example, if some required libraries are missing or if you want to statically link a plugin that only supports dynamic linking.

If you want to avoid building some specific plugin, specify the

-DWITHOUT\_xxx=1  
cmake option.

### MariaDB starting with 10.1

Use

PLUGIN\_xxx  
cmake variables. They can be set on the command line with  
-DPLUGIN\_xxx=  
  
value

or in the cmake gui. Supported values are

Value	Effect
-------	--------

<b>NO</b>	the plugin will be not compiled at all
<b>STATIC</b>	the plugin will be compiled statically, if supported. Otherwise it will be not compiled.
<b>DYNAMIC</b>	the plugin will be compiled dynamically, if supported. Otherwise it will be not compiled. This is the default behavior.
<b>AUTO</b>	the plugin will be compiled statically, if supported. Otherwise it will be compiled dynamically.
<b>YES</b>	same as <b>AUTO</b> , but if plugin prerequisites (for example, specific libraries) are missing, it will not be skipped, it will abort cmake with an error.

Note that unlike autotools, cmake tries to configure and build incrementally. You can modify one configuration option and cmake will only rebuild the part of the tree affected by it. For example, when you do

```
cmake -DWITH_EMBEDDED_SERVER=1
```

in the already-built tree, it will make libmysqld to be built, but no other configuration options will be changed or reset to their default values.

In particular this means that if you have run, for example

MariaDB 5.5 - 10.0

```
cmake -DWITHOUT_OQGRAPH=1
```

MariaDB starting with 10.1

```
cmake -DPLUGIN_OQGRAPH=NO
```

and later you want to restore the default behavior (with OQGraph being built) in the same build tree, you would need to run

MariaDB 5.5 - 10.0

```
cmake -UWITHOUT_OQGRAPH
```

MariaDB starting with 10.1

```
cmake -DPLUGIN_OQGRAPH=DYNAMIC
```

Alternatively, you might simply delete the

CMakeCache.txt

file — this is the file where cmake stores current build configuration — and rebuild everything from scratch.

## 2.1.2.8.6 Creating the MariaDB Source Tarball

How to create a source tar.gz file.

First [Setup your build environment](#).

Then use automake/configure/make to generate the tar file:

```
BUILD/autorun.sh
./configure --with-plugin-xtradb
make dist
```

This creates a source file with a name like

```
mysql-5.3.2-MariaDB-beta.tar.gz
```

See also [the generic build instructions](#).

## 2.1.2.8.7 Creating the MariaDB Binary Tarball

How to generate binary tar.gz files.

- [Setup your build environment](#).
- [Build binaries](#) with your preferred options/plugins.

### MariaDB until 5.3

- Use

```
make_binary_distribution  
to generate a binary tar file:
```

```
cd mariadb-source  
../scripts/make_binary_distribution
```

This creates a source file with a name like

```
mariadb-5.3.2-MariaDB-beta-linux-x86_64.tar.gz  
in your current directory.
```

The other option is to use the bakery scripts. In this case you don't have to compile MariaDB source first.

```
cd $PACKAGING_WORK  
bakery/preheat.sh  
cd bakery_{number}  
bakery/tarbake51.sh last:1 $MARIA_WORK  
bakery/autobake51-bintar.sh mariadb-{version_num}-maria-beta-ourdelta{number}.tar.gz
```

### MariaDB starting with 5.5

If the binaries are already built, you can generate a binary tarball with

```
make package
```

## 2.1.2.8.8 Build Environment Setup for Mac

### XCode

- Install Xcode from Apple (free registration required): <https://developer.apple.com/xcode/> or from your Mac OS X installation disk (macports needs XCode >= 3.1, so if you do not have that version or greater you will need to download the latest version, which is 900+ MB)

You can install the necessary dependencies using either MacPorts or Homebrew.

### Using MacPorts

- Download and install the MacPorts dmg image from <http://www.macports.org>
- After installing, update it from the terminal:  

```
sudo port -v selfupdate
```

```
sudo port install cmake jemalloc Judy openssl boost gnutls
```

### Using Homebrew

- Download and install Homebrew from <https://brew.sh>

```
brew install cmake jemalloc trailldb/Judy Judy openssl boost gnutls
```

Your Mac should now have everything it needs to get, compile, and otherwise work with the MariaDB source code. The next step is to actually get a copy of the code. For help with this see the [Getting the MariaDB Source Code](#) page.

When building with Mac, you'll need

```
-DOPENSSL_ROOT_DIR=/usr/local/openssl  
passed as a  
cmake  
argument to build against openssl correctly.
```

## 2.1.2.8.9 Building MariaDB From a Source RPM

## 2.1.2.8.10 Building MariaDB on CentOS

### Contents

1. [Installing Build Dependencies](#)
2. [Building MariaDB](#)
3. [Creating MariaDB-compat package](#)
4. [Additional Dependencies](#)
5. [More about CMake and CPackRPM](#)

In the event that you are using the Linux-based operating system CentOS or any of its derivatives, you can optionally compile MariaDB from source code. This is useful in cases where you want use a more advanced release than the one that's available in the official repositories, or when you want to enable certain feature that are not otherwise accessible.

### Installing Build Dependencies

Before you start building MariaDB, you first need to install the build dependencies required to run the compile. CentOS provides a tool for installing build dependencies. The

```
yum-builddep
```

utility reads a package and generates a list of the packages required to build from source, then calls YUM to install them for you. In the event that this utility is not available on your system, you can install it through the

```
yum-utils
```

package. Once you have it, install the MariaDB build dependencies.

```
# yum-builddep mariadb-server
```

Running this command installs many of the build dependencies, but it doesn't install all of them. Not all the required packages are noted and it's run against the official CentOS package of MariaDB, not necessarily the version that you want to install. Use YUM to install the remaining packages.

```
# yum install git \
  gcc \
  gcc-c++ \
  bison \
  libxml2-devel \
  libevent-devel \
  rpm-build
```

In addition to these, you also need to install

```
gnutls
or
openssl
, depending on the TLS implementation you want to use.
```

For more information on dependencies, see [Linux Build Environment](#).

### Building MariaDB

Once you have the base dependencies installed, you can retrieve the source code and start building MariaDB. The source code is available on GitHub. Use the

```
--branch
option to specify the particular version of MariaDB you want to build.
```

```
$ git clone --branch 10.3 https://github.com/MariaDB/server.git
```

With the source repository cloned onto your system, you can start building MariaDB. Run CMake to read MariaDB for the build,

```
$ cmake -DRPM=centos7 server/
```

Once CMake readies the relevant Makefile for your system, use Make to build MariaDB.

```
$ make package
```

This generates an RPM file, which you can then install on your system or copy over to install on other CentOS hosts.

### Creating MariaDB-compat package

MariaDB-compat package contains libraries from older MariaDB releases. They cannot be built from the current source tree, so cpack creates them by repackaging old MariaDB-shared packages. If you want to have -compat package created, you need to download MariaDB-shared-5.3 and MariaDB-

shared-10.1 rpm packages for your architecture (any minor version will do) and put them *one level above* the source tree you're building. CMake will pick them up and create a MariaDB-compat package. CMake reports it as

```
$ ls ../*.rpm
./MariaDB-shared-10.1.17-centos7-x86_64.rpm
./MariaDB-shared-5.3.12-122.el5.x86_64.rpm
$ cmake -DRPM=centos7 .
...
Using ../MariaDB-shared-5.3.12-122.el5.x86_64.rpm to build MariaDB-compat
Using ../MariaDB-shared-10.1.17-centos7-x86_64.rpm to build MariaDB-compat
```

## Additional Dependencies

In the event that you miss a package while installing build dependencies, CMake may continue to fail after you install the necessary packages. If this happens to you, delete the CMake cache then run the above command again:

```
$ rm CMakeCache.txt
```

When CMake runs through the tests again, it should now find the packages it needs, instead of the cache telling it they're unavailable.

## More about CMake and CPackRPM

See also [building RPM packages from source](#)

### 2.1.2.8.11 Building MariaDB on Fedora

In the event that you are using the Linux-based operating system Fedora or any of its derivatives and would like to compile MariaDB from source code, you can do so using the MariaDB build in the official repositories.

## Installing Build Dependencies

MariaDB requires a number of packages to compile from source. Fortunately, you can use the package in the Fedora repository to retrieve most of the relevant build dependencies through DNF.

```
# dnf builddep mariadb-server
```

Running DNF in this way pulls the build dependencies for the release of MariaDB compiled by your version of Fedora. These may not be all the dependencies you need to build the particular version of MariaDB you want to use, but it will retrieve most of them.

You'll also need to install Git to retrieve the source repository:

```
# dnf install git
```

## Building MariaDB

Once you have the base dependencies installed, you can retrieve the source code and start building MariaDB. The source code is available on GitHub. Use the

```
--branch  
option to specify the particular version of MariaDB you want to build.
```

```
$ git clone --branch 10.3 https://github.com/MariaDB/server.git mariadb-server
```

With the source repository cloned onto your system, you can start building MariaDB. Change into the new `mariadb-server/` directory and run CMake to prepare the build.

```
$ mkdir mariadb-build
$ cd mariadb-build
$ cmake -DRPM=fedora ../mariadb-server
```

Once CMake readies the relevant Makefile for your system, use Make to build MariaDB.

```
$ make package
```

### 2.1.2.8.12 Building MariaDB on Debian

## Contents

1. [Installing Build Dependencies](#)
2. [Building MariaDB](#)
  1. [After Building](#)

In the event that you are using the Linux-based operating system Debian or any of its direct derivatives and would like to compile MariaDB from source code, you can do so using the MariaDB source repository for the release that interests you. For Ubuntu and its derivatives, see [Building on Ubuntu](#).

Before you begin, install the

software-properties-common  
and  
devscripts  
packages:

```
$ sudo apt-get install -y software-properties-common \
  devscripts
```

## Installing Build Dependencies

MariaDB requires a number of packages to compile from source. Fortunately, you can use the MariaDB repositories to retrieve the necessary code for the version you want. Use the [Repository Configuration](#) tool to determine how to set up the MariaDB repository for your release of Debian, the version of MariaDB that you want to install, and the mirror that you want to use.

First add the authentication key for the repository, then add the repository.

```
$ sudo apt-key adv --recv-keys \
  --keyserver hkp://keyserver.ubuntu.com:80 \
  0xF1656F24C74CD1D8
$ sudo add-apt-repository 'deb [arch=amd64] http://nyc2.mirrors.digitalocean.com/mariadb/repo/10.3/debian stretch main'
```

The second command added text to the

/etc/apt/sources.list

file. One of these lines is the repository containing binary packages for MariaDB, the other contains the source packages. The line for the source packages is commented out by default. This can be scripted:

```
sed -e '/^# deb-src.*mariadb/s/^# //' -i /etc/apt/sources.list
```

Alternately, open the file using your preferred text editor and uncomment the source repository.

```
$ sudo vim /etc/apt/sources.list
...
deb [arch=amd64] http://nyc2.mirrors.digitalocean.com/mariadb/repo/10.3/debian stretch main
deb-src [arch=amd64] http://nyc2.mirrors.digitalocean.com/mariadb/repo/10.3/debian stretch main
```

Once the repository is set up, you can use

apt-get

to retrieve the build dependencies. MariaDB packages supplied by Ubuntu and packages supplied by the MariaDB repository have the same base name of

mariadb-server

. You need to specify the specific version you want to retrieve.

```
$ sudo apt-get update
$ sudo apt-get build-dep -y mariadb-server-10.3
```

## Building MariaDB

Once you have the base dependencies installed, you can retrieve the source code and start building MariaDB. The source code is available on GitHub. Use the

--branch

option to specify the particular version of MariaDB you want to build.

```
$ git clone --branch 10.3 https://github.com/MariaDB/server.git
```

The source code includes scripts to install the remaining build dependencies. For Ubuntu, they're located in the

debian/

directory. Navigate into the repository and run the

autobake-deb.sh

script. Then use

```
$ export DEB_BUILD_OPTIONS=parallel=$(nproc)
$ cd server/
$ ./debian/autobake-deb.sh
```

## After Building

After building the packages, it is a good idea to put them in a repository. See the [Creating a Debian Repository](#) page for instructions.

### 2.1.2.8.13 Building MariaDB on FreeBSD

#### Contents

- 1. [Using Ports](#)
  - 1. [Building MariaDB from Ports](#)
- 2. [Using Poudriere](#)
  - 1. [Building MariaDB](#)
  - 2. [Using Poudriere Repositories](#)

It is relatively straightforward to build MariaDB from source on FreeBSD. When working with an individual host, you can use Ports to compile particular or multiple versions of MariaDB. When working with multiple hosts, you can use Poudriere to build MariaDB once, then serve it as a package to multiple FreeBSD hosts.

## Using Ports

The FreeBSD Ports Collection provides a series of Makefiles that you can use to retrieve source code, configure builds, install dependencies and compile software. This allows you to use more advanced releases than what is normally available through the package managers as well as enable any additional features that interest you.

In the event that you have not used Ports before on your system, you need to first fetch and extract the Ports tree. This downloads the Ports tree from FreeBSD and extracts it onto your system, placing the various Makefiles, patches and so on in the

```
/usr/ports/  
directory.
```

```
# portsnap fetch extract
```

In the event that you have used Ports before on this system, run Portsnap again to download and install any updates to the Ports tree.

```
# portsnap fetch update
```

This ensures that you are using the most up to date release of the Ports tree that is available on your system.

## Building MariaDB from Ports

Once Portsnap has installed or updated your Ports tree, you can change into the relevant directory and install MariaDB. Ports for MariaDB are located in the

```
/usr/ports/databases/  
directory.
```

```
$ ls /usr/ports/databases | grep mariadb

mariadb-connector-c
mariadb-connector-odbc
mariadb100-client
mariadb100-server
mariadb101-client
mariadb101-server
mariadb102-client
mariadb102-server
mariadb103-client
mariadb103-server
mariadb55-client
mariadb55-server
```

Note that FreeBSD treats the Server and Client as separate packages. The Client is a dependency of the Server, so you only need to build the Server to get both. It also provides a number of different versions. You can search the available ports from [Fresh Ports](#). Decide what version of MariaDB you want to install, then change into the relevant directory. Once in the directory, run Make to build MariaDB.

```
# cd /usr/ports/databases/mariadb103-server  
# make
```

In addition to downloading and building MariaDB, Ports also downloads and build any libraries on which MariaDB depends. Each port it builds will take you to a GUI window where you can select various build options. In the case of MariaDB, this includes various storage engines and specific features you need in your build.

Once you finish building the ports, install MariaDB on your system and clean the directory to free up disk space.

```
# make install clean
```

This installs FreeBSD on your server. You can now enable, configure and start the service as you normally would after installing MariaDB from a package.

## Using Poudriere

Poudriere is a utility for building FreeBSD packages. It allows you to build MariaDB from a FreeBSD Jail, then serve it as a binary package to other FreeBSD hosts. You may find this is particularly useful when building to deploy multiple MariaDB servers on FreeBSD, such as with Galera Cluster or similar deployments.

### Building MariaDB

Once you've configured your host to use Jails and Poudriere, initialize a jail to use in building packages and a jail for managing ports.

```
# poudriere jail -c -j package-builder -v 11.2-RELEASE  
# poudriere ports -c -p local-ports
```

This creates two jails,

package-builder  
and  
local-ports

, which you can then use to build MariaDB. Create a text file to define the packages you want to build. Poudriere will build these packages as well as their dependencies. MariaDB is located at

databases/mariadb103-server  
. Adjust the path to match the version you want to install.

```
$ vi mariadb-package-builder.txt  
  
databases/mariadb103-server
```

Use the

options  
command to initialize the build options for the packages you want Poudriere to compile.

```
# poudriere options -j package-builder -p local-ports -z mariadb-builder -f mariadb-package-builder.txt
```

Lastly, use the

bulk  
command to compile the packages.

```
# poudriere bulk -j package-builder -p local-ports -z mariadb-builder -f mariadb-package-builder.txt
```

## Using Poudriere Repositories

In order to use Poudriere, you need to set up and configure a web server, such as Nginx or Apache to serve the directory that Poudriere built. For instance, in the case of the above example, you would map to the

package-builder  
jail:  
/usr/local/poudriere/data/packages/package-builder/  
. You may find it useful to map this directory to a sub-domain, for instance  
https

*pkg.example.com*

or something similar.

Lastly, you need to configure the FreeBSD hosts to use the Poudriere repository you just created. On each host, disable the FreeBSD official repositories and enable the Poudriere repository as an alternative.

```
# vi /usr/local/etc/pkg/repos/FreeBSD.conf

FreeBSD: {
    enabled: no
}
```

Then add the URL for your Poudriere repository to configuration file:

```
# vi /usr/local/etc/pkg/repos/mariadb.conf

custom: {
    url: "https://pkg.example.com",
    enabled: yes
}
```

You can then install MariaDB from Poudriere using the package manager.

```
# pkg install mariadb103-server
```

## 2.1.2.8.14 Building MariaDB on Gentoo

MariaDB is in Gentoo, so the steps to build it are:

1. Synchronize your tree with

```
emerge --sync
```

2. Build MariaDB using

```
emerge mariadb
```

## 2.1.2.8.15 Building MariaDB on Solaris and OpenSolaris

The following two articles should help you get your Solaris machine prepared to build MariaDB (just ignore the parts about installing buildbot):

- [Buildbot Setup for Solaris Sparc](#)
- [Buildbot Setup for Solaris x86](#)

## Notes

- The BUILD dir contains various scripts for compiling MariaDB on Solaris. The  
BUILD/compile-solaris-amd64  
and  
BUILD/compile-solaris-amd64-debug  
are probably the most useful.
- The scripts do not play nice with non-bash shells such as the Korn Shell (ksh). So if your /bin/sh is pointing at ksh or ksh93, you'll want to change it so that it points at bash.

## 2.1.2.8.16 Building MariaDB on Ubuntu

### Contents

1. [Installing Build Dependencies](#)
2. [Building MariaDB](#)
  1. [Further Dependencies](#)
  2. [After Building](#)

In the event that you are using the Linux-based operating system Ubuntu or any of its derivatives and would like to compile MariaDB from source code, you can do so using the MariaDB source repository for the release that interests you.

Before you begin, install the

```
software-properties-common
,
devscripts
and
equivs
packages.
```

```
$ sudo apt-get install software-properties-common \
  devscripts \
  equivs
```

## Installing Build Dependencies

MariaDB requires a number of packages to compile from source. Fortunately, you can use the MariaDB repositories to retrieve the necessary code for the version you want. Use the [Repository Configuration](#) tool to determine how to set up the MariaDB repository for your release of Ubuntu, the version of MariaDB that you want to install, and the mirror that you want to use.

First add the authentication key for the repository, then add the repository.

```
$ sudo apt-key adv --recv-keys \
  --keyserver hkp://keyserver.ubuntu.com:80 \
  0xF1656F24C74CD1D8
$ sudo add-apt-repository --update --yes --enable-source \
  'deb [arch=amd64] http://nyc2.mirrors.digitalocean.com/mariadb/repo/10.3/ubuntu \'$(lsb_release -sc)\' main'
```

Once the repository is set up, you can use

`apt-get`

to retrieve the build dependencies. MariaDB packages supplied by Ubuntu and packages supplied by the MariaDB repository have the same base name of

`mariadb-server`

. You need to specify the specific version you want to retrieve.

```
$ sudo apt-get build-dep mariadb-10.3
```

## Building MariaDB

Once you have the base dependencies installed, you can retrieve the source code and start building MariaDB. The source code is available on GitHub. Use the

`--branch`

option to specify the particular version of MariaDB you want to build.

```
$ git clone --branch 10.3 https://github.com/MariaDB/server.git
```

The source code includes scripts to install the remaining build dependencies. For Ubuntu, they're located in the

`debian/`

directory. Navigate into the repository and run the

`autobake-deb.sh`

script. Then use

```
$ cd server/
$ ./debian/autobake-deb.sh
```

## After Building

After building the packages, it is a good idea to put them in a repository. See the [Creating a Debian Repository](#) page for instructions.

### 2.1.2.8.17 Building MariaDB on Windows

## Contents

- 1. Build Requirements
- 2. Building Windows Binaries
- 3. Build Variations
  - 1. Debug Builds
  - 2. 32bit and 64 bit Builds
    - 1. Build 64 bit binary
    - 2. Build 32 bit binary
  - 3. IDE Builds
- 4. Building the ZIP Package
- 5. Building the MSI Package
- 6. Including HeidiSQL in the MSI Installer
- 7. Code Signing Support for MariaDB Release Processing
- 8. Building Packages for MariaDB Releases
- 9. Running Tests
  - 1. Running a Test Under Debugger

## Build Requirements

To build MariaDB you need the following:

- **Visual C++** : We currently support Visual Studio 2019 and 2022. Generally we try to support the two most recent VS versions, but build ourselves using the last one. Community editions will work fine; we only use them in our builds. While installing Visual Studio, make sure to [add "Desktop Development with C++"](#).
- **CMake** : We recommend the latest release. Older releases might not support your version of Visual Studio. Visual Studio 2019 requires cmake 3.14 at least.
- **Git** : Required to build newer versions from the source tree.
  - NOTE: run

```
git config --global core.autocrlf input
```

after the installation, otherwise some mtr tests will fail

In the "Adjusting your PATH" dialog, choose "Use Git from Windows command prompt", otherwise wrong (mingw64) git and perl will be in your PATH

- **Bison from GnuWin32** : Bison creates parts of the SQL parser. Choose "Complete package except sources" when downloading.
  - NOTE: **Do not** install this into your default path with spaces (e.g. under  
C:\Program Files\GnuWin32  
); the build will break due to [this bison bug](#). Instead, install into  
C:\GnuWin32
  - Add  
C:\GnuWin32\bin  
to your system  
PATH  
after installation.
- **Strawberry perl** : Used to run the test suite. [ActiveState Perl](#) is another Win32 Perl distribution and should work as well (but it is not as well tested).  
NOTE: Cygwin or mingw Perl versions **will not work** for testing. Use Windows native Perl, please.
- Optional: If you intend to build the MSI packages, install [[ <https://wixtoolset.org/releases/>] Windows Installer XML] . If you build MSI with 10.4, also modify your Visual Studio installation, add "Redistributable MSMs" (see [MDEV-22555](#) )
- **Gnu Diff** , needed if you run mysql-test-run.pl tests.

Verify that bison.exe, bsr.exe or git.exe, cmake.exe and perl.exe can be found in the PATH environment variable with "

```
where bison
",
where git
",
where perl
" etc. from the command line prompt.
```

## Building Windows Binaries

The above instructions assume [MariaDB 10.2](#) or higher.

Branch the MariaDB bsr repository, or unpack the source archive. On the command prompt, switch to your source directory, then execute:

```
mkdir bld  
cd bld  
cmake ..  
cmake --build . --config RelWithDebInfo
```

The above example builds a release configured for 64 bit systems in a subdirectory named

```
bld  
."  
cmake ...  
" is the configuration step,  
cmake --build . --config Relwithdebinfo  
" is the build step.
```

## Build Variations

### Debug Builds

Building Debug version is done with:

```
cmake --build . --config Debug
```

### 32bit and 64 bit Builds

#### Build 64 bit binary

Visual Studio 2019-2022 cmake generator will use host architecture by default, that is, with the steps above, cmake will build x64 binaries on x64 machine.

#### Build 32 bit binary

pass -A Win32 parameter for CMake, like this

```
cmake .. -A Win32
```

Historical note: With Visual Studio 2017 and earlier, one had to pass the name of 32bit generator ,e.g cmake .. -G "Visual Studio 15 2017"

For a complete list of available generators, call "cmake" without any parameters.

### IDE Builds

Instead of calling "

```
cmake --build  
" as above, open  
MySQL.sln  
. When Visual Studio starts, choose Build/Compile.
```

## Building the ZIP Package

```
cmake --build . --config relwithdebinfo --target package
```

This is how it is "done by the book", standard cmake target.

MariaDB however uses non-standard target "win\_package" for the packaging for its releases, it generates 2 ZIPs, a slim one with executables, and another one with debuginfo (.PDB files). The debuginfo is important to be able to debug released binaries, and to analyze crashes.

```
cmake --build . --config relwithdebinfo --target win_package
```

## Building the MSI Package

```
cmake --build . --config relwithdebinfo  
cmake --build . --config relwithdebinfo --target MSI
```

## Including HeidiSQL in the MSI Installer

Starting with MariaDB 5.2.7 , it is possible to build an installer which includes 3rd party products, as described in [MWL#200](#) . Currently only HeidiSQL support is implemented; it is also included in the official builds. Use the CMake parameter

-DWITH\_THIRD\_PARTY=HeidiSQL  
to include it in the installer.

## Code Signing Support for MariaDB Release Processing

MariaDB builds optionally support authenticode code signing with an optional parameter

SIGNCODE  
. Use  
cmake -DSIGNCODE=1  
during the configuration step to sign the binaries in the  
ZIP  
and  
MSI  
packages.

**Important:** for

SIGNCODE=1  
to work, the user that runs the build needs to install a valid authenticode digital certificate into their certificate store, otherwise the packaging step will fail.

## Building Packages for MariaDB Releases

The full script to create the release in an out-of-source build with Visual Studio with signed binaries might look like:

```
mkdir bld
cd bld
cmake .. -DSIGNCODE=1 -DWITH_THIRD_PARTY=HeidiSQL
cmake --build . --config relwithdebinfo --target win_package
cmake --build . --config relwithdebinfo --target MSI
```

This command sequence will produce a ZIP package (e.g mariadb-5.2.6-win32.zip) and MSI package (e.g mariadb-5.2.6-win32.msi) in the  
bld  
directory.

## Running Tests

- **Important: Do not use Cygwin bash, MinGW bash, Git bash, WSL bash, or any other bash when running the test suite. You will then very likely use the wrong version of Perl too (a "Unix-flavoured" one on Windows), and spend a lot of time trying to figure out why this version of Perl does not work for the test suite. Use native perl, in cmd.exe , or powershell instead,**
- Switch mysql-test subdirectory of the build directory

```
cd C:\server\bld\mysql-test
```

- Run the test suite

```
perl mysql-test-run.pl --suite=main --parallel=auto
```

## Running a Test Under Debugger

Assuming VS is installed on the machine

```
perl mysql-test-run.pl <test_name> --vsjitdebugger
```

If vsjitdebugger does not start, you can edit AeDebug registry key as mentioned in

[https://docs.microsoft.com/en-us/visualstudio/debugger/debug-using-the-just-in-time-debugger?view=vs-2019#jit\\_errors](https://docs.microsoft.com/en-us/visualstudio/debugger/debug-using-the-just-in-time-debugger?view=vs-2019#jit_errors)

Alternatively:

```
perl mysql-test-run.pl <test_name> --devenv
```

(devenv.exe needs to be in PATH)

or, if you prefer WinDBG

```
perl mysql-test-run.pl <test_name> --windbg
```

## 2.1.2.8.18 Installing MariaDB Server on macOS Using Homebrew

### Contents

1. [Upgrading MariaDB](#)
2. [Building MariaDB Server from source](#)
3. [Other resources](#)

MariaDB Server is available for installation on macOS (formerly Mac OS X) via the [Homebrew](#) package manager.

MariaDB Server is available as a Homebrew "bottle", a pre-compiled package. This means you can install it without having to build from source yourself. This saves time.

After installing Homebrew, MariaDB Server can be installed with this command:

```
brew install mariadb
```

After installation, start MariaDB Server:

```
mysql.server start
```

To auto-start MariaDB Server, use Homebrew's services functionality, which configures auto-start with the launchctl utility from [launchd](#) :

```
brew services start mariadb
```

After MariaDB Server is started, you can log in as your user:

```
mysql
```

Or log in as root:

```
sudo mysql -u root
```

## Upgrading MariaDB

First you may need to update your brew installation:

```
brew update
```

Then, to upgrade MariaDB Server:

```
brew upgrade mariadb
```

## Building MariaDB Server from source

In addition to the "bottled" MariaDB Server package available from Homebrew, you can use Homebrew to build MariaDB from source. This is useful if you want to use a different version of the server or enable some different capabilities that are not included in the bottle package.

Two components not included in the bottle package (as of MariaDB Server 10.1.19) are the CONNECT and OQGRAPH engines, because they have non-standard dependencies. To build MariaDB Server with these engines, you must first install

```
boost  
and  
judy
```

. As of December 2016, judy is in the Homebrew "boneyard", but the old formula still works on macOS Sierra. Follow these steps to install the dependencies and build the server:

```
brew install boost homebrew/boneyard/judy  
brew install mariadb --build-from-source
```

You can also use Homebrew to build and install a pre-release version of MariaDB Server (for example MariaDB Server 10.2, when the highest GA version is MariaDB Server 10.1). Use this command to build and install a "development" version of MariaDB Server:

```
brew install mariadb --devel
```

## Other resources

- mariadb.rb on [github](#)
- Terin Stock ([terinjokes](#)) who is the packager for Homebrew
- MariaDB Server on macOS: [Does it even make sense to try?](#) (video)

## 2.1.2.8.19 Creating a Debian Repository

Below are instructions for creating your own Debian repository. The instructions are based on <http://www.debian.org/doc/manuals/repository-howto/repository-howto.en.html>

```
REPO_DIR={pick some location}
mkdir $REPO_DIR
mkdir $REPO_DIR/binary
mkdir $REPO_DIR/source
cp *.deb *.ddeb $REPO_DIR/binary
cd $REPO_DIR
dpkg-scanpackages binary /dev/null | gzip -9c > binary/Packages.gz
dpkg-scansources source /dev/null | gzip -9c > source/Sources.gz
```

## Using the Debian repository you just created

One needs to add a new file to the

```
/etc/apt/sources.list.d/
directory. For instance a new file called
mariadb.list
```

```
# sergey's MariaDB repository
#
deb file:///home/psergey/testrepo binary/
deb-src file:///home/psergey/testrepo source/
```

after which one can run

```
apt-get update # Let apt learn about the new repository
apt-get install mariadb-server
```

and collect bugs :-).

"apt-get install" will spray output of scripts and servers all over /var/log. It is also possible to set DEBIAN\_SCRIPT\_DEBUG=1 to get some (not all) of it to stdout.

## Cleaning up after failed installation

Run

```
dpkg --get-selections | grep mariadb
dpkg --get-selections | grep mysql
```

to see what is installed, and then

```
dpkg --purge <packages>
```

until the former produces empty output. Note: after some failures, /etc/mysql and /var/lib/mysql are not cleaned and still need to be removed manually.

## 2.1.2.8.20 Building MariaDB From Source Using musl-based GNU/Linux

### Instructions on compiling MariaDB on musl-based operating systems (Alpine)

#### Contents

1. [Instructions on compiling MariaDB on musl-based operating systems \(Alpine\)](#)
2. [Using cmake](#)

The instructions on this page will help you compile [MariaDB](#) from source. Links to more complete instructions for specific platforms can be found on the

[source](#) page.

- First, [get a copy of the MariaDB source](#) .
- Next, [prepare your system to be able to compile the source](#) .

## Using cmake

MariaDB 10.1 and above is compiled using `cmake` . You can configure your build simply by running `cmake` using special option, i.e.

```
cmake . -DWITHOUT_TOKUDB=1
```

To build and install MariaDB after running `cmake` use

```
make  
sudo make install
```

Note that building with MariaDB this way will disable tokuDB, till tokuDB becomes fully supported on musl.

### 2.1.2.8.21 Compiling MariaDB for Debugging

#### Contents

1. [Compiling MariaDB for Debugging Using the CMAKE\\_BUILD\\_TYPE Option](#)
2. [Compiling MariaDB for Debugging Using Build Scripts](#)
  1. [Example of Compiling MariaDB for Debugging Using Build Scripts](#)
  3. [Building Optimized Build With Debug Symbols](#)
  4. [Doing a Debug Build on Debian/Ubuntu](#)
    1. [Temporarily Installing your Debug Build](#)
    2. [Reinstalling your Release Build](#)
  5. [Different Compilation Options](#)
    1. [Changing DBUG\\_ASSERT to Print to Error Log](#)
  6. [See Also](#)

Compiling MariaDB with full debug information includes all code symbols and also new code to do internal testing of structures and allow one to trace MariaDB execution. A full debug binary will be notably slower than a normal binary (30%).

## Compiling MariaDB for Debugging Using the CMAKE\_BUILD\_TYPE Option

On Unix systems, you can build a debug build by executing

```
cmake  
and by setting the  
CMAKE_BUILD_TYPE  
option to  
Debug  
. For example:
```

```
cmake -DCMAKE_BUILD_TYPE=Debug .
```

## Compiling MariaDB for Debugging Using Build Scripts

The other option is to use the scripts in the BUILD directory that will compile MariaDB with most common debug options and plugins:

```
./BUILD/compile-pentium64-debug-max
```

or alternatively if you want to use the [Valgrind](#) memory checking tool with the [MariaDB test system](#) :

```
./BUILD/compile-pentium64-valgrind-max
```

There are separate build scripts for different configurations in the BUILD directory.

You can find a list of the needed packages/libraries for building on Linux [here](#).

## Example of Compiling MariaDB for Debugging Using Build Scripts

- Scripts containing "debug" in the name are for developers that want to build, develop and test MariaDB.
- Scripts containing "valgrind" in the name are for running mysqld under [http://valgrind.org/valgrind]. Compiling for valgrind means that we are using another implementation of MEM\_ROOT to allow valgrind to better detect memory overruns. In addition, some memory areas are marked as used/not used to remove some false positives.
- Scripts containing "max" in the name include all normal plugins.

Here is an example of how to compile MariaDB for debugging in your home directory with [MariaDB 5.2.9](#) as an example:

```
cd ~  
mkdir mariadb  
cd mariadb  
tar xvf mariadb-5.2.9.tar.gz  
ln -s mariadb-5.2.9 current  
cd current  
.BUILD/compile-pentium64-debug-max
```

The last command will produce a debug version of

```
sql/mysqld  
. If you have a system other than 64 bit Intel/AMD on Linux you can use a different  
BUILD/...-debug-max  
file. If this fails, you can try with:
```

```
cmake --build . --config Debug  
make -j4
```

## Building Optimized Build With Debug Symbols

To build MariaDB with symbols, to get better stack traces and to be able to debug the binary with

```
gdb  
, you need to supply the  
-g3  
option to the  
gcc  
compiler.
```

Just compiling with

```
-g3  
will make the binary much bigger but the slowdown of the server should be negligible.
```

One way to do this is to edit the script

```
BUILD/compile-pentium64-max
```

and add the -g3 last on the line with

```
extra_flags  
, like this:
```

```
extra_flags="$pentium64_cflags $fast_cflags -g3"
```

After that you can compile MariaDB with debugging symbols by just execution the above script.

## Doing a Debug Build on Debian/Ubuntu

To build a "mysqld" binary with debugging enabled that uses the same parameters as the ones used in Debian/Ubuntu binary packages, you must do as follows (you must have a deb-src line of one of the MariaDB repositories on your /etc/apt/sources.list in order to do that):

```
apt-get build-dep mariadb-10.2  
apt-get install cmake libaio1 libaio-dev  
apt-get source mariadb-10.2  
cd mariadb-10.2*  
.debian/rules configure  
.BUILD/compile-pentium64-debug-all
```

Then you will have your "debugging enabled" mysqld binary in the sql/ directory.

This binary can directly replace the one provided by the binary package that is placed on "/usr/bin/mysqld".

## Temporarily Installing your Debug Build

The commands shown below replace the release

```
mysqld  
binary with the debug  
mysqld  
binary that you compiled. Most importantly, they replace the binary in a way which makes it trivial to revert back to the original release  
mysqld  
binary.
```

First, [stop MariaDB](#).

Then, use the

```
mv  
utility to rename the release  
mysqld  
binary:
```

```
sudo mv /usr/sbin/mysqld /usr/sbin/mysqld-orig
```

Note: Do not use the

```
cp  
utility because that will change the file modification timestamp.
```

Then, install the debug

```
mysqld  
binary from your source tree:
```

```
sudo install ~/mariadb-10.3.14/sql/mysqld /usr/sbin/mysqld-debug
```

Then, link the

```
mysqld  
path to the path of your debug  
mysqld  
binary:
```

```
sudo ln -s /usr/sbin/mysqld-debug /usr/sbin/mysqld
```

Then, [start MariaDB](#).

Be sure to replace

```
/usr/sbin/mysqld  
with the path to your  
mysqld  
binary and to also replace
```

~

```
/mariadb-10.3.14/sql/mysqld  
with the path to your debug  
mysqld  
binary.
```

## Reinstalling your Release Build

If you want to restore your original

```
mysqld  
binary, you can do it with the following process::
```

First, [stop MariaDB](#).

Then, execute the following command to delete the symbolic link:

```
sudo rm /usr/sbin/mysqld
```

Then, execute the following command to move the original

```
mysqld  
release binary back into place:
```

```
sudo mv /usr/sbin/mysqld-orig /usr/sbin/mysqld
```

Then, [start MariaDB](#).

Be sure to replace

```
/usr/sbin/mysqld  
with the path to your  
mysqld  
binary
```

Notice that the debug

```
mysqld  
binary located at  
/usr/sbin/mysqld-debug  
was not deleted. Only the symbolic link to this file was deleted. The debug  
mysqld  
binary is still present if it is needed again in the future.
```

## Different Compilation Options

### Changing DBUG\_ASSERT to Print to Error Log

A debug binary has lots of code checks and asserts, that are not checked in production. This is done to get more performance when running in production. In some cases, when one is trying to find a hard-to-repeat bug, it could be beneficial to have these checks in production builds too.

Compiling with

```
-DDBUG_ASSERT_AS_PRINTF  
will change DBUG_ASSERT() to print any failed check to the error log.
```

```
cmake . -DDBUG_ASSERT_AS_PRINTF
```

Enabling the above option should not have any notable impact on performance (probably < 1% slowdown). This is achieved by grouping asserts in MariaDB server code into two groups:

- Fast checks, using  
`DBUG_ASSERT()`  
: These are converted to printing to error log.
- Slow checks, using  
`DBUG_SLOW_ASSERT()`  
. These will always be removed in production builds.

## See Also

- [Build environment setup for Linux](#)
- [Debugging MariaDB with a debugger](#)
- [Creating a trace file](#)
- [Using ASAN with MariaDB](#)

### 2.1.2.8.22 Cross-compiling MariaDB

#### Buildroot

[Buildroot](#) is a way to cross compile MariaDB and other packages into a root filesystem. In the menuconfig you need to enable "Enable C++ Support" first under "Toolchain". After C++ is enabled MariaDB is an option under "Target Packages" ->"Libraries" -> "Databases" -> "mysql support" -> "mysql variant" -> "mariadb". Also enable the "mariadb server" below the "mysql support" option.

#### Details

To cross-compile with cmake you will need a toolchain file. See, for example, [here](#). Besides cmake specific variables it should have, at least

```
SET(STACK_DIRECTION -1)  
SET(HAVE_IB_GCC_ATOMIC_BUILTINS 1)
```

with appropriate values for your target architecture. Normally these MariaDB configuration settings are detected by running a small test program, and it cannot be done when cross-compiling.

Note that during the build few helper tools are compiled and then immediately used to generate more source files for this very build. When cross-compiling these tools naturally should be built for the host architecture, not for the target architecture. Do it like this (assuming you're in the mariadb source tree):

```
$ mkdir host
$ cd host
$ cmake ..
$ make import_executables
$ cd ..
```

Now the helpers are built and you can cross-compile:

```
$ mkdir target
$ cd target
$ cmake .. -DCMAKE_TOOLCHAIN_FILE=/path/to/toolchain/file.cmake -DIMPORT_EXECUTABLES=../host/import_executables.cmake
$ make
```

Here you invoke cmake, specifying the path to your toolchain file and the path to the

`import_executables.cmake`

that you have just built on the previous step. Of course, you can also specify any other cmake parameters that could be necessary for this build, for example, enable or disable specific storage engines.

See also <https://lists.launchpad.net/maria-discuss/msg02911.html>

## 2.1.2.8.23 MariaDB Source Configuration Options

### Contents

#### 1. See Also

All CMake configuration options for MariaDB can be displayed with:

```
cmake . -LH
```

## See Also

- [Get the Code, Build it, Test it](#) (mariadb.org)
- [Generic Build Instructions](#)

## 2.1.2.8.24 Building RPM Packages From Source

To generate RPM packages from the build, supply the

`-DRPM=yes`

flag to CMake.

The value

`yes`

(or whatever) will build generic RPM packages. It is also possible to use one of these special values that will slightly modify what kind of layout the resulting RPM packages will have:

- `fedora`
- `centos7/rhel7`
- `centos8/rhel8`
- `sles`

What these do are controlled in the following CMake files:

- `cmake/cpack_rpm.cmake`
- `cmake/build_configurations/mysql_release.cmake`
- `cmake/mariadb_connector_c.cmake`

## See Also

- [About the MariaDB RPM Files](#)
- [Building MariaDB on CentOS](#)
- [Installing MariaDB RPM Files](#)
- [MariaDB RPM Packages](#)

## 2.1.2.8.25 Compile and Using MariaDB with

# AddressSanitizer (ASAN)

## Contents

1. [What is AddressSanitizer \(ASAN\)](#)
2. [How to Compile MariaDB for ASAN](#)
3. [Running an ASAN Build](#)
4. [Other Things](#)
  - 1. [Using Valgrind](#)
5. [See Also](#)

## What is AddressSanitizer (ASAN)

[AddressSanitizer \(aka ASan\)](#) is a memory error detector for C/C++. It finds a lot of the same things as [valgrind](#), but with a lot less overhead.

- Use after free (dangling pointer dereference)
- Heap buffer overflow
- Stack buffer overflow
- Global buffer overflow
- Use after return
- Use after scope
- Initialization order bugs
- Memory leaks

To use ASAN you need a gcc version that supports ASAN. gcc 4.8.5 and up are known to work.

## How to Compile MariaDB for ASAN

ASAN is supported in [MariaDB 10.1](#) and up.

You can use one of the two following build commands:

```
cmake . -DWITH_ASAN=ON
```

or from [MariaDB 10.2](#) and up:

```
./BUILD/compile-pentium64-asan-max
```

## Running an ASAN Build

To run mysqld with instrumentation you have to set the

```
ASAN_OPTIONS  
environment variable before starting  
mysqld  
. Either in your shell or in your mysqld\_safe script.
```

```
export ASAN_OPTIONS=abort_on_error=1:leak_check_at_exit=0
```

The above command will abort mysqld if any errors are found, which is good for debugging. If you set `abort_on_error=0` all errors are logged to your error log file (`mysqld.err`).

## Other Things

A side effect of using ASAN builds is that you will not get any core file if your server crashes.

## Using Valgrind

The [MariaDB test system](#) can use [Valgrind](#) for finding memory leaks and wrong memory accesses. Valgrind is an instrumentation framework for building dynamic analysis tools. If Valgrind is installed on your system, you can simply use `mysql-test-run --valgrind` to run the test under Valgrind.

## See Also

- [Compiling MariaDB for debugging](#)

## 2.1.2.9 Distributions Which Include MariaDB

## Contents

1. Linux Distributions
2. BSD Distributions
3. macOS

The following is a partial list of distributions which include MariaDB in their package repositories. For these you can use the distribution's management system to install MariaDB.

The term "default" in the list below refers to the distribution's default relational or MySQL-type database.

## Linux Distributions

- [Alpine Linux](#) — Defaults to MariaDB. [MariaDB 10.5](#) has been available 3.13.5
- [4mLinux](#) — Defaults to MariaDB. [MariaDB 10.3](#) has been available since 26.0
- [ALT Linux](#) — [MariaDB 5.5](#) included in 7.0.0, MariaDB is default from 8.1, which includes [MariaDB 10.1](#), 9.1 includes [MariaDB 10.4](#)
- [Arch Linux](#) — Features [MariaDB 10.5](#), and replaced MySQL as a default
- [Asianux](#) — [MariaDB 5.5](#) replaced MySQL in 7.
- [Austrumi](#) — Defaulted to [MariaDB 5.3](#) in 2.4.8, 3.5.8 includes [MariaDB 10.1](#)
- [BlackArch](#) — Defaulted to [MariaDB 5.5](#) in 2014.07.01, 2020.06.01 includes [MariaDB 10.4](#)
- [BlueOnyx](#) — 5209 defaults to [MariaDB 5.5](#), 5210R to [MariaDB 10.3](#)
- [BlueStar](#) — 4.8.4 defaults to [MariaDB 10.1](#), 5.4.2 to [MariaDB 10.5](#)
- [CentOS](#) — [MariaDB 5.5](#) replaced MySQL in CentOS 7
- [The Chakra Project](#) — MariaDB replaced MySQL as default in 2013.05. 2016.02 includes [MariaDB 10.1](#)
- [Debian](#) — Debian 8 "Jessie" includes [MariaDB 10.0](#), Debian 9 "Stretch" [MariaDB 10.1](#), Debian 10 "Buster" [MariaDB 10.3](#), Debian 11 "Bullseye" [MariaDB 10.5](#) as default.
- [Elastix](#) — Defaulted to [MariaDB 5.5](#) in 4.0.76
- [Exherbo](#) — Includes [MariaDB 10.4](#)
- [Fedora](#) — [MariaDB 5.5](#) became the default relational database in Fedora 19. Fedora 30 includes [MariaDB 10.3](#), Fedora 34 includes [MariaDB 10.5](#).
- [Funtoo](#) — Includes [MariaDB 5.5](#)
- [Gentoo Linux](#)
- [Guix](#) — 11.2.0 includes [MariaDB 10.1](#)
- [Hanthana](#) — 19.1 defaulted to [MariaDB 5.5](#), 21 includes [MariaDB 10.0](#), 28.1.2 includes [MariaDB 10.2](#), 30 includes [MariaDB 10.3](#)
- [KaOS](#) — Defaulted to [MariaDB 5.5](#) in 2014.12, 2019.04 includes [MariaDB 10.3](#)
- [Kali Linux](#) — 2017.3 Included [MariaDB 10.1](#), 2021.1 includes [MariaDB 10.5](#)
- [GNU/Linux KDu](#) — [MariaDB 5.5](#) replaced MySQL as a default in 2.0 Final.
- [Korora](#) — Defaulted to MariaDB in 19.1, 26 includes [MariaDB 10.1](#)
- [Linux from Scratch](#) — 10.1-BLFS defaults to [MariaDB 10.5](#)
- [Lunar](#) — 1.7.0 includes [MariaDB 5.5](#), Moonbase includes [MariaDB 10.5](#)
- [Mageia](#) — [MariaDB 5.5](#) replaced MySQL as default in version 3, [MariaDB 10.3](#) from version 7.1, [MariaDB 10.5](#) from 8.
- [Manjaro Linux](#) — Defaulted to [MariaDB 5.5](#) in 0.8.11, 16.10.3 includes [MariaDB 10.1](#).
- [NixOS](#) — 14.0.4.630 included [MariaDB 10.0](#), 18.09 includes [MariaDB 10.2](#), Stable includes [MariaDB 10.5](#)
- [Network Security Toolkit](#) — 20-5663 defaulted to [MariaDB 5.5](#), 32-11992 includes [MariaDB 10.4](#)
- [NuTyX](#) — 14.11 included [MariaDB 10.0](#), defaulted to [MariaDB 10.1](#) in 8.2.1, 20.12.1 includes [MariaDB 10.5](#)
- [OpenELEC](#)
- [OpenEuler](#) — 21.9 includes [10.5](#), 20.03 LTS includes [MariaDB 10.3](#), as does 21.03
- [Open Mandriva](#) — Defaulted to [MariaDB 10.0](#) in 2014.2, includes [MariaDB 10.5](#) in 4.2
- [openSUSE](#) — [MariaDB 5.5](#) became the default relational database in [openSUSE 12.3](#), and [MariaDB 10.4](#) the default since 15.2
- [Oracle Linux](#) — 7.3 includes [MariaDB 5.5](#). 8.0 includes [MariaDB 10.3](#)
- [Paldo](#) — Defaults to [MariaDB 10.5](#) in Stable
- [Parabola GNU/Linux](#) — Includes [MariaDB 10.1](#) since 3.7
- [Parrot Security](#) — Based on Debian's testing branch (stretch), Parrot Security switched from MySQL to [MariaDB 10.0](#) in 3.1, 4.7 includes [MariaDB 10.3](#)
- [Parted Magic](#) — Defaulted to [MariaDB 5.5](#) in 2015\_11\_13
- [PCLinuxOS](#) — Replaced MySQL with [MariaDB 10.1](#) in 2017.03
- [Pisi Linux](#) — Defaulted to [MariaDB 10.0](#) in 1.1
- [Plamo](#) — Defaulted to [MariaDB 5.5](#) in 5.3.1, 7.3 includes [MariaDB 10.2](#)
- [PoliArch](#) — Defaulted to [MariaDB 5.5](#) in 13.1, 15.1 includes [MariaDB 10.0](#)
- [Red Hat Enterprise Linux](#) — [MariaDB 5.5](#) was included as the default "MySQL" database since RHEL 7, RHEL 8 includes [MariaDB 10.3](#)
- [ROSA](#) — Defaulted to [MariaDB 10.0](#) in R4 and [MariaDB 10.1](#) in R9.
- [Sabayon](#) — Included [MariaDB 10.0](#) in 14.12, includes [MariaDB 10.3](#) since 19.03
- [Scientific Linux](#) — Defaulted to [MariaDB 5.5](#) in 7.3
- [Slackware](#) — [MariaDB 5.5](#) replaced MySQL as default in 14.1. 14.2 includes [MariaDB 10.0](#), current includes [MariaDB 10.5](#)
- [SliTaz GNU/Linux](#) — Includes [MariaDB 10.0](#) in 5.0-rolling
- [SME Server](#) — started to use [MariaDB 5.5](#) from 10.0
- [Springdale Linux](#) — Defaulted to [MariaDB 5.5](#) in 7.2, 8.1 includes [MariaDB 10.3](#)
- [SuliX](#) — Defaults to [MariaDB 5.5](#) in 8.
- [SUSE Linux Enterprise](#) — [MariaDB 10.0](#) is the default relational database option on 12-SP2, 15-SP2 includes [MariaDB 10.4](#)
- [Ubuntu](#) — [MariaDB 5.5](#) was included in Trusty Tahr 14.04. 20.04 includes [MariaDB 10.3](#).
- [Void](#) — Includes [MariaDB 10.5](#) in current
- [Wifislax](#) — Defaulted to [MariaDB 10.0](#) in 4.11.1

## BSD Distributions

- **Dragonfly BSD** — 3.8 included [MariaDB 5.5](#) . 5.8.0 includes [MariaDB 10.4](#) .
- **FreeBSD** — MariaDB is available in the ports tree and the FreeBSD Manual has instructions on [Installing Applications: Packages and Ports](#) . [MariaDB 10.5](#) is included in 12.2
  - [MariaDB on FreshPorts](#)
- **NetBSD** — 6.1 and 7.0 include [MariaDB 5.5](#) .
- **OpenBSD** — [MariaDB 10.0](#) was included in 5.7, 6.8 includes [MariaDB 10.5](#) .

## macOS

- [Homebrew](#) — If you have Homebrew installed, you can install MariaDB Server by executing  

```
brew install mariadb
```

 . Find out more at [Installing MariaDB Server on macOS Using Homebrew](#) .
- [MacPorts](#) — This provides [mariadb](#) and [mariadb-server](#) . A [quick guide](#) on how to install it.

## 2.1.2.10 Running Multiple MariaDB Server Processes

### Contents

1. [Configuring Multiple MariaDB Server Processes](#)
2. [Starting Multiple MariaDB Server Processes](#)
  1. [Service Managers](#)
    1. [Systemd](#)
  2. [Starting the Server Process Manually](#)
    1. [mysqld](#)
    2. [mysqld\\_safe](#)
    3. [mysqld\\_multi](#)
  3. [Other Options](#)

It is possible to run multiple MariaDB Server processes on the same server, but there are certain things that need to be kept in mind. This page will go over some of those things.

## Configuring Multiple MariaDB Server Processes

If multiple MariaDB Server process are running on the same server, then at minimum, you will need to ensure that the different instances do not use the same

`datadir`

,

`port`

, and

`socket`

. The following example shows these options set in an [option file](#) :

```
[client]
# TCP port to use to connect to mysqld server
port=3306
# Socket to use to connect to mysqld server
socket=/tmp/mysql.sock
[mariadb]
# TCP port to make available for clients
port=3306
# Socket to make available for clients
socket=/tmp/mysql.sock
# Where MariaDB should store all its data
datadir=/usr/local/mysql/data
```

The above values are the defaults. If you would like to run multiple MariaDB Server instances on the same server, then you will need to set unique values for each instance.

There may be additional options that also need to be changed for each instance. Take a look at the full list of options for

`mysqld`

To see the current values set for an instance, see [Checking Program Options](#) for how to do so.

To list the default values, check the end of:

```
mysqld --help --verbose
```

## Starting Multiple MariaDB Server Processes

There are several different methods to start or stop the MariaDB Server process. There are two primary categories that most of these methods fall into: starting the process with the help of a service manager, and starting the process manually. See [Starting and Stopping MariaDB](#) for more information.

### Service Managers

[sysVinit](#) and [systemd](#) are the most common Linux service managers. [launchd](#) is used in MacOS X. [Upstart](#) is a less common service manager.

#### Systemd

RHEL/CentOS 7 and above, Debian 8 Jessie and above, and Ubuntu 15.04 and above use [systemd](#) by default.

For information on how to start and stop multiple MariaDB Server processes on the same server with this service manager, see [systemd: Interacting with Multiple MariaDB Server Processes](#).

### Starting the Server Process Manually

#### mysqld

[mysqld](#)

is the actual MariaDB Server binary. It can be started manually on its own.

If you want to force each instance to read only a single [option file](#), then you can use the

[--defaults-file](#)

option:

```
mysqld --defaults-file=/etc/my_instance1.cnf
```

#### mysqld\_safe

[mysqld\\_safe](#)

is a wrapper that can be used to start the

[mysqld](#)

server process. The script has some built-in safeguards, such as automatically restarting the server process if it dies. See [mysqld\\_safe](#) for more information.

If you want to force each instance to read only a single [option file](#), then you can use the

[--defaults-file](#)

option:

```
mysqld_safe --defaults-file=/etc/my_instance1.cnf
```

#### mysqld\_multi

[mysqld\\_multi](#)

is a wrapper that can be used to start the

[mysqld](#)

## Other Options

In some cases, there may be easier ways to run multiple MariaDB Server instances on the same server, such as:

- Using [dbdeployer](#).
- Starting multiple [Docker](#) containers.

### 2.1.2.11 Installing MariaDB Alongside MySQL

MariaDB was designed as a drop-in replacement of MySQL, with more features, new storage engines, fewer bugs, and better performance, but you can also install it alongside MySQL. (This can be useful, for example, if you want to migrate databases/applications one by one.)

Here are the steps to install MariaDB near an existing MySQL installation.

- Download the compiled binary tar.gz that contains the latest version ([mariadb-5.5.24-linux-x86\\_64.tar.gz](#) - as of writing this article) and extract the files in a directory of your choice. I will assume for this article that the directory was `/opt`.

```
[root@mariadb-near-mysql ~]# cat /etc/issue
CentOS release 6.2 (Final)

[root@mariadb-near-mysql ~]# rpm -qa mysql*
mysql-5.1.61-1.el6_2.1.x86_64
mysql-libs-5.1.61-1.el6_2.1.x86_64
mysql-server-5.1.61-1.el6_2.1.x86_64

[root@mariadb-near-mysql ~]# ps axf | grep mysqld
2072 pts/0    S+      0:00          \_ grep mysqld
1867 ?        S      0:01 /bin/sh /usr/bin/mysqld_safe --datadir=/var/lib/mysql --socket=/var/lib/mysql/mysql.sock ...
1974 ?        S1      0:06  \_ /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --user=mysql ...
```

- Create data directory and symlinks as below:

```
[root@mariadb-near-mysql opt]# mkdir mariadb-data
[root@mariadb-near-mysql opt]# ln -s mariadb-5.5.24-linux-x86_64 mariadb
[root@mariadb-near-mysql opt]# ls -al
total 20
drwxr-xr-x.  5 root root 4096 2012-06-06 07:27 .
dr-xr-xr-x. 23 root root 4096 2012-06-06 06:38 ..
lrwxrwxrwx.  1 root root   27 2012-06-06 07:27 mariadb -> mariadb-5.5.24-linux-x86_64
drwxr-xr-x. 13 root root 4096 2012-06-06 07:07 mariadb-5.5.24-linux-x86_64
drwxr-xr-x.  2 root root 4096 2012-06-06 07:26 mariadb-data
```

- Create group **mariadb** and user **mariadb** and set correct ownerships:

```
[root@mariadb-near-mysql opt]# groupadd --system mariadb
[root@mariadb-near-mysql opt]# useradd -c "MariaDB Server" -d /opt/mariadb -g mariadb --system mariadb
[root@mariadb-near-mysql opt]# chown -R mariadb:mariadb mariadb-5.5.24-linux-x86_64/
[root@mariadb-near-mysql opt]# chown -R mariadb:mariadb mariadb-data/
```

- Create a new **my.cnf** in `/opt/mariadb` from support files:

```
[root@mariadb-near-mysql opt]# cp mariadb/support-files/my-medium.cnf mariadb-data/my.cnf
[root@mariadb-near-mysql opt]# chown mariadb:mariadb mariadb-data/my.cnf
```

- Edit the file `/opt/mariadb-data/my.cnf` and add custom paths, socket, port, user and the most important of all: data directory and base directory. Finally the file should have at least the following:

```
[client]
port = 3307
socket = /opt/mariadb-data/mariadb.sock

[mysqld]
datadir      = /opt/mariadb-data
basedir      = /opt/mariadb
port = 3307
socket = /opt/mariadb-data/mariadb.sock
user        = mariadb
```

- Copy the init.d script from support files in the right location:

```
[root@mariadb-near-mysql opt]# cp mariadb/support-files/mysql.server /etc/init.d/mariadb
[root@mariadb-near-mysql opt]# chmod +x /etc/init.d/mariadb
```

- Edit **/etc/init.d/mariadb** replacing **mysql** with **mariadb** as below:

```
- # Provides: mysql
+ # Provides: mariadb
- basedir=
+ basedir=/opt/mariadb
- datadir=
+ datadir=/opt/mariadb-data
- lock_file_path="$lockdir/mysql"
+ lock_file_path="$lockdir/mariadb"
```

The trickiest part will be the last changes to this file. You need to tell mariadb to use only one cnf file. In the **start** section after **\$bindir/mysqld\_safe** add **--defaults-file=/opt/mariadb-data/my.cnf**. Finally the lines should look like:

```
# Give extra arguments to mysqld with the my.cnf file. This script
# may be overwritten at next upgrade.
$bindir/mysqld_safe --defaults-file=/opt/mariadb-data/my.cnf --datadir="$datadir" --pid-file="$mysqld_pid_file_path" $other_args
```

The same change needs to be made to the mysqladmin command below in the **wait\_for\_ready()** function so that the mariadb start command can properly listen for the server start. In the **wait\_for\_ready()** function, after **\$bindir/mysqladmin** add **--defaults-file=/opt/mariadb-data/my.cnf**. The lines should look like:

```
wait_for_ready () {
[...]
if $bindir/mysqladmin --defaults-file=/opt/mariadb-data/my.cnf ping >/dev/null 2>&1; then
```

- Run **mysql\_install\_db** by explicitly giving it the my.cnf file as argument:

```
[root@mariadb-near-mysql opt]# cd mariadb
[root@mariadb-near-mysql mariadb]# scripts/mysql_install_db --defaults-file=/opt/mariadb-data/my.cnf
```

- Now you can start MariaDB by

```
[root@mariadb-near-mysql opt]# /etc/init.d/mariadb start
Starting MySQL... [ OK ]
```

- Make MariaDB start at system start:

```
[root@mariadb-near-mysql opt]# cd /etc/init.d
[root@mariadb-near-mysql init.d]# chkconfig --add mariadb
[root@mariadb-near-mysql init.d]# chkconfig --levels 3 mariadb on
```

- Finally test that you have both instances running:

```
[root@mariadb-near-mysql ~]# mysql -e "SELECT VERSION();"
+-----+
| VERSION() |
+-----+
| 5.1.61    |
+-----+
[root@mariadb-near-mysql ~]# mysql -e "SELECT VERSION();" --socket=/opt/mariadb-data/mariadb.sock
+-----+
| VERSION()      |
+-----+
| 5.5.24-MariaDB |
+-----+
```

## What about MariaDB Upgrades ?

By having the **mariadb.socket** , **my.cnf** file and **databases** in **/opt/mariadb-data** if you want to upgrade the MariaDB version you will only need to:

- extract the new version from the archive in **/opt** near the current version
- stop MariaDB
- change the symlink **mariadb** to point to the new directory
- start MariaDB
- run upgrade script... but remember you will need to provide the socket option **--socket=/opt/mariadb-data/mariadb.sock**

## 2.1.2.12 GPG

### Contents

1. [New key](#)
2. [Old key](#)
3. [Configuring](#)

The MariaDB project signs their MariaDB packages for Debian, Ubuntu, Fedora, CentOS, and Red Hat.

### New key

Our repositories for Debian "Sid" and the Ubuntu 16.04 and beyond "Xenial" use a new GPG signing key. As detailed in [MDEV-9781](#), APT 1.2.7 (and later) prefers SHA2 GPG keys and now prints warnings when a repository is signed using a SHA1 key like our previous GPG key. We have created a new SHA2 key for use with these affected repositories.

Information about this key:

- The short Key ID is:  
0xC74CD1D8
- The long Key ID is:  
0xF1656F24C74CD1D8
- The full fingerprint of the key is:  
177F 4010 FE56 CA33 3630 0305 F165 6F24 C74C D1D8
- The key can be added on Debian-based systems using the following command:

```
sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8
```

The instructions in the repository configuration tool for Ubuntu 16.04 "Xenial" and Debian "Stretch" and higher have been updated to reference this new key. Repositories for previous versions of Debian and Ubuntu still use the old key, so no changes are needed there.

### Old key

The GPG Key ID of the MariaDB signing key is

0xCBCB082A1BB943DB

. The short form of the id is

0x1BB943DB

and the full key fingerprint is:

```
1993 69E5 404B D5FC 7D2F E43B CBCB 082A 1BB9 43DB
```

This key is still used by the yum/dnf/zypper repositories for RedHat, CentOS, Fedora, openSUSE, and SLES.

If you configure the mariadb.org rpm repositories using the repository configuration tool (see below) then your package manager will prompt you to import the key the first time you install a package from the repository.

You can also import the key directly using the following command:

```
sudo rpmkeys --import https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
```

### Configuring

See the [repository configuration tool](#) for details on configuring repositories that use these keys.

## 2.1.2.13 MariaDB Deprecation Policy

### Contents

1. [Current Package Platforms](#)
2. [Deprecated Package Platforms](#)
3. [Support for Deprecated Platforms](#)
4. [See Also](#)

The [MariaDB Foundation](#) tries to support as many different Operating Systems, Linux Distributions, and processor architectures as possible. However, when a distribution or OS stops receiving security and other updates it becomes difficult for the MariaDB project to provide freely packages for that platform. In such cases, our policy is to deprecate the platform and stop providing binary packages for it.

This policy and related deprecated dates are from the [MariaDB Foundation](#). For information on the MariaDB Corporation's policies related to supporting

software, see the [Engineering Policies](#) page.

## Current Package Platforms

The MariaDB project builds packages for the following:

Platform	Planned Deprecation Date
Debian 9 "Stretch"	Jun 2022
Windows 8.1	Jan 2023
Ubuntu 18.04 "Bionic"	Apr 2023
Windows Server 2012 R2	Oct 2023
Red Hat Enterprise Linux 7.x	Jun 2024
CentOS 7.x	Jun 2024
Debian 10 "Buster"	Jun 2024
SLES 12.x	Oct 2024
Ubuntu 20.04 "Focal"	Apr 2025
Debian 11 "Bullseye"	Jun 2026
SLES 15.x	Jul 2028
Red Hat Enterprise Linux 8.x	Jun 2029
Fedora 34	approximately 1 month after release of Fedora 36
Fedora 35	approximately 1 month after release of Fedora 37
SLES 12.5	6 months after release of SLES 12.6
SLES 15.0	6 months after release of SLES 15.1
Arch Linux	N/A - rolling distribution
Mageia	N/A - rolling distribution

- [Ubuntu Release Information](#) (End of Standard Support)
- [Debian LTS information](#)
- [General Debian release information](#)
- [Red Hat Release Information](#)
- [Fedora Release Information](#)
- [FreeBSD Security Information](#)
- [openSUSE Lifetime Information](#)
- [SLES Lifecycle Information](#)
- [Windows client lifecycle info](#)
- [Windows server lifecycle info](#)

## Deprecated Package Platforms

The MariaDB project no longer builds packages for the following Operating Systems and Linux Distributions:

Platform	Deprecation Date	Final MariaDB Version(s)
Ubuntu 21.04 "Hirsute"	Feb 2022	<a href="#">MariaDB 10.7.2</a> , <a href="#">MariaDB 10.6.6</a> , <a href="#">MariaDB 10.5.14</a>
Fedora 33	Feb 2022	<a href="#">MariaDB 10.7.2</a> , <a href="#">MariaDB 10.6.6</a> , <a href="#">MariaDB 10.5.14</a> , <a href="#">MariaDB 10.4.23</a>
CentOS 8.x	Feb 2022	<a href="#">MariaDB 10.7.2</a> , <a href="#">MariaDB 10.6.6</a> , <a href="#">MariaDB 10.5.14</a> , <a href="#">MariaDB 10.4.23</a> , <a href="#">MariaDB 10.3.33</a>
Ubuntu 20.10 "Groovy"	Jul 2021	<a href="#">MariaDB 10.6.4</a> , <a href="#">MariaDB 10.5.12</a> , <a href="#">MariaDB 10.4.21</a> , <a href="#">MariaDB 10.3.31</a>
Fedora 32	Apr 2021	<a href="#">MariaDB 10.5.10</a> , <a href="#">MariaDB 10.4.19</a> , <a href="#">MariaDB 10.3.29</a>
Ubuntu 16.04 LTS "Xenial"	Apr 2021	<a href="#">MariaDB 10.5.10</a> , <a href="#">MariaDB 10.4.19</a> , <a href="#">MariaDB 10.3.29</a> , <a href="#">MariaDB 10.2.38</a>
Mint 18 LTS "Serena"	Apr 2021	<a href="#">MariaDB 10.5.10</a> , <a href="#">MariaDB 10.4.19</a> , <a href="#">MariaDB 10.3.29</a> , <a href="#">MariaDB 10.2.38</a>
Fedora 31	Nov 2020	<a href="#">MariaDB 10.5.7</a> , <a href="#">MariaDB 10.4.16</a> , <a href="#">MariaDB 10.3.26</a>
Red Hat Enterprise Linux 6.x	Nov 2020	<a href="#">MariaDB 10.4.16</a> , <a href="#">MariaDB 10.3.26</a> , <a href="#">MariaDB 10.2.35</a> , <a href="#">MariaDB 10.1.48</a>
CentOS 6.x	Nov 2020	<a href="#">MariaDB 10.4.16</a> , <a href="#">MariaDB 10.3.26</a> , <a href="#">MariaDB 10.2.35</a> , <a href="#">MariaDB 10.1.48</a>
Fedora 30	Aug 2020	<a href="#">MariaDB 10.5.5</a> , <a href="#">MariaDB 10.4.14</a> , <a href="#">MariaDB 10.3.24</a>

Ubuntu 19.10 "Eoan"	Jul 2020	<a href="#">MariaDB 10.5.5</a> , <a href="#">MariaDB 10.4.14</a> , <a href="#">MariaDB 10.3.24</a>
Debian 8 "Jessie"	Jun 2020	<a href="#">MariaDB 10.4.13</a> , <a href="#">MariaDB 10.3.23</a> , <a href="#">MariaDB 10.2.32</a> , <a href="#">MariaDB 10.1.45</a>
Ubuntu 19.04 "Disco"	Jan 2020	<a href="#">MariaDB 10.4.12</a> , <a href="#">MariaDB 10.3.22</a>
Windows Server 2008	Jan 2020	<a href="#">MariaDB 10.4.12</a> , <a href="#">MariaDB 10.3.22</a>
Windows Server 2008 R2	Jan 2020	<a href="#">MariaDB 10.4.12</a> , <a href="#">MariaDB 10.3.22</a>
Windows 7	Jan 2020	<a href="#">MariaDB 10.4.12</a> , <a href="#">MariaDB 10.3.22</a>
Fedora 29	Dec 2019	<a href="#">MariaDB 10.4.11</a> , <a href="#">MariaDB 10.3.21</a>
Ubuntu 18.10 "Cosmic"	Jul 2019	<a href="#">MariaDB 10.4.7</a> , <a href="#">MariaDB 10.3.17</a> , <a href="#">MariaDB 10.2.26</a> , <a href="#">MariaDB 10.1.41</a>
openSUSE 42.3	Jun 2019	<a href="#">MariaDB 10.4.7</a> , <a href="#">MariaDB 10.3.16</a> , <a href="#">MariaDB 10.2.25</a> , <a href="#">MariaDB 10.1.41</a>
Fedora 28	May 2019	<a href="#">MariaDB 10.4.5</a> , <a href="#">MariaDB 10.3.15</a> , <a href="#">MariaDB 10.2.24</a>
Ubuntu 14.04 LTS "Trusty"	Apr 2019	<a href="#">MariaDB 10.4.4</a> , <a href="#">MariaDB 10.3.14</a> , <a href="#">MariaDB 10.2.23</a> , <a href="#">MariaDB 10.1.40</a> , <a href="#">MariaDB 5.5.64</a>
Mint 17.1 LTS "Rebecca"	Apr 2019	<a href="#">MariaDB 10.4.4</a> , <a href="#">MariaDB 10.3.14</a> , <a href="#">MariaDB 10.2.23</a> , <a href="#">MariaDB 10.1.40</a> , <a href="#">MariaDB 5.5.64</a>
Mint 17 LTS "Qiana"	Apr 2019	<a href="#">MariaDB 10.4.4</a> , <a href="#">MariaDB 10.3.14</a> , <a href="#">MariaDB 10.2.23</a> , <a href="#">MariaDB 10.1.40</a> , <a href="#">MariaDB 5.5.64</a>
SLES 11.4	Mar 2019	<a href="#">MariaDB 10.1.40</a> , <a href="#">MariaDB 5.5.64</a>
Fedora 27	Nov 2018	<a href="#">MariaDB 10.3.11</a> , <a href="#">MariaDB 10.2.19</a>
Ubuntu 17.10 "Artful"	Jul 2018	<a href="#">MariaDB 10.3.8</a> , <a href="#">MariaDB 10.2.16</a> , <a href="#">MariaDB 10.1.34</a>
Fedora 26	May 2018	<a href="#">MariaDB 10.3.7</a> , <a href="#">MariaDB 10.2.15</a> , <a href="#">MariaDB 10.1.33</a>
Debian 7 "Wheezy"	May 2018	<a href="#">MariaDB 10.3.7</a> , <a href="#">MariaDB 10.2.15</a> , <a href="#">MariaDB 10.1.33</a>
Fedora 25	Feb 2018	<a href="#">MariaDB 10.3.5</a> , <a href="#">MariaDB 10.2.13</a> , <a href="#">MariaDB 10.1.31</a>
Ubuntu 17.04 "Zesty"	Jan 2018	<a href="#">MariaDB 10.3.4</a> , <a href="#">MariaDB 10.2.12</a> , <a href="#">MariaDB 10.1.30</a>
openSUSE 42.2	Jan 2018	<a href="#">MariaDB 10.3.4</a> , <a href="#">MariaDB 10.2.12</a> , <a href="#">MariaDB 10.1.30</a>
Red Hat Enterprise Linux 7.2	Nov 2017	<a href="#">MariaDB 10.3.3</a> , <a href="#">MariaDB 10.2.12</a> , <a href="#">MariaDB 10.1.30</a> , <a href="#">MariaDB 10.0.33</a> , <a href="#">MariaDB 5.5.58</a>
CentOS 7.2	Nov 2017	<a href="#">MariaDB 10.3.3</a> , <a href="#">MariaDB 10.2.12</a> , <a href="#">MariaDB 10.1.30</a> , <a href="#">MariaDB 10.0.33</a> , <a href="#">MariaDB 5.5.58</a>
Fedora 24	Aug 2017	<a href="#">MariaDB 10.2.8</a>
Ubuntu 16.10 "Yakkety"	Jul 2017	<a href="#">MariaDB 10.2.7</a> , <a href="#">MariaDB 10.1.26</a> , <a href="#">MariaDB 10.0.32</a>
Ubuntu 12.04 LTS "Precise"	Apr 2017	<a href="#">MariaDB 10.1.24</a> , <a href="#">MariaDB 10.0.31</a> , <a href="#">MariaDB 5.5.56</a>
Mint 13 LTS "Maya"	Apr 2017	<a href="#">MariaDB 10.1.24</a> , <a href="#">MariaDB 10.0.31</a> , <a href="#">MariaDB 5.5.56</a>
Red Hat Enterprise Linux 7.1	Mar 2017	<a href="#">MariaDB 10.1.24</a> , <a href="#">MariaDB 10.0.31</a> , <a href="#">MariaDB 5.5.56</a>
CentOS 7.1	Mar 2017	<a href="#">MariaDB 10.1.24</a> , <a href="#">MariaDB 10.0.31</a> , <a href="#">MariaDB 5.5.56</a>
Red Hat Enterprise Linux 5	Mar 2017	<a href="#">MariaDB 10.1.22</a> , <a href="#">MariaDB 10.0.30</a> , <a href="#">MariaDB 5.5.54</a>
CentOS 5	Mar 2017	<a href="#">MariaDB 10.1.22</a> , <a href="#">MariaDB 10.0.30</a> , <a href="#">MariaDB 5.5.54</a>
Fedora 23	Feb 2017	<a href="#">MariaDB 10.2.4</a> , <a href="#">MariaDB 10.1.22</a> , <a href="#">MariaDB 10.0.30</a>
OpenSUSE 13	Jan 2017	<a href="#">MariaDB 10.2.4</a> , <a href="#">MariaDB 10.1.22</a> , <a href="#">MariaDB 10.0.30</a>
Fedora 22	Aug 2016	<a href="#">MariaDB 10.1.17</a> , <a href="#">MariaDB 10.0.27</a>
Ubuntu 15.10 "Wily"	Jul 2016	<a href="#">MariaDB 10.1.16</a> , <a href="#">MariaDB 10.0.26</a>
Windows 2003 Server	Apr 2016	<a href="#">MariaDB 10.1.13</a> , <a href="#">MariaDB 10.0.24</a> , <a href="#">MariaDB 5.5.48</a>
Windows XP	Apr 2016	<a href="#">MariaDB 10.1.13</a> , <a href="#">MariaDB 10.0.24</a> , <a href="#">MariaDB 5.5.48</a>
Debian 6 "Squeeze"	Feb 2016	<a href="#">MariaDB 10.0.24</a> , <a href="#">MariaDB 5.5.48</a>
Ubuntu 15.04 "Vivid"	Jan 2016	<a href="#">MariaDB 10.1.11</a> , <a href="#">MariaDB 10.0.24</a>
Fedora 21	Dec 2015	<a href="#">MariaDB 10.1.10</a> , <a href="#">MariaDB 10.0.23</a>
Fedora 20	Oct 2015	<a href="#">MariaDB 10.0.22</a> , <a href="#">MariaDB 5.5.46</a>
Ubuntu 14.10 "Utopic"	Jul 2015	<a href="#">MariaDB 10.0.22</a> , <a href="#">MariaDB 5.5.46</a>
Ubuntu 10.04 LTS "Lucid"	Apr 2015	<a href="#">MariaDB 10.0.18</a> , <a href="#">MariaDB 5.5.43</a>
Mint 9 LTS "Isadora"	Apr 2015	<a href="#">MariaDB 10.0.18</a> , <a href="#">MariaDB 5.5.43</a>
Fedora 19	Apr 2015	<a href="#">MariaDB 10.0.18</a> , <a href="#">MariaDB 5.5.43</a>

FreeBSD 9.2	Sep 2014	
Ubuntu 13.10 "Saucy"	Jul 2014	<a href="#">MariaDB 10.0.14</a> , <a href="#">MariaDB 5.5.40</a>
Mint 16 "Petra"	Jul 2014	<a href="#">MariaDB 10.0.14</a> , <a href="#">MariaDB 5.5.40</a>
Ubuntu 12.10 "Quantal"	Apr 2014	<a href="#">MariaDB 10.0.11</a> , <a href="#">MariaDB 5.5.37</a>
Mint 14 "Nadia"	Apr 2014	<a href="#">MariaDB 10.0.11</a> , <a href="#">MariaDB 5.5.37</a>
Ubuntu 13.04 "Raring"	Jan 2014	<a href="#">MariaDB 10.0.8</a> , <a href="#">MariaDB 5.5.35</a>
Mint 15 "Olivia"	Jan 2014	<a href="#">MariaDB 10.0.8</a> , <a href="#">MariaDB 5.5.35</a>
Fedora 18	Dec 2013	<a href="#">MariaDB 10.0.8</a> , <a href="#">MariaDB 5.5.35</a>
Fedora 17	Aug 2013	<a href="#">MariaDB 10.0.5</a> , <a href="#">MariaDB 5.5.34</a>
Ubuntu 8.04 LTS "Hardy"	Apr 2013	<a href="#">MariaDB 10.0.2</a> , <a href="#">MariaDB 5.5.31</a>
Ubuntu 11.10 "Oneiric"	Apr 2013	<a href="#">MariaDB 10.0.2</a> , <a href="#">MariaDB 5.5.31</a>
Mint 12 "Lisa"	Apr 2013	<a href="#">MariaDB 10.0.2</a> , <a href="#">MariaDB 5.5.31</a>
Fedora 16	Feb 2013	<a href="#">MariaDB 10.0.1</a> , <a href="#">MariaDB 5.5.29</a>
Ubuntu 10.10 "Maverick"	Jan 2013	<a href="#">MariaDB 10.0.1</a> , <a href="#">MariaDB 5.5.29</a>
Ubuntu 11.04 "Natty"	Jan 2013	<a href="#">MariaDB 10.0.1</a> , <a href="#">MariaDB 5.5.29</a>
Debian 5 "Lenny"	Jan 2013	<a href="#">MariaDB 10.0.1</a> , <a href="#">MariaDB 5.5.29</a>
Debian 4 "Etch"		
Ubuntu 9.10 "Karmic"		
Ubuntu 9.04 "Jaunty"		
Ubuntu 8.10 "Intrepid"		

## Support for Deprecated Platforms

If your chosen Linux Distribution or Operating System is deprecated, packages or support are not completely unavailable. The [MariaDB Corporation](#) provides support for all versions of MariaDB back to even very old MySQL versions. This includes packaged binaries. For specific dates related to each version and more details on the MariaDB Corporation's policies, see the [Engineering Policies](#) page.

## See Also

- [MariaDB Bug Fixing Policy](#)
- [MariaDB Maintenance Policy](#)
- [MariaDB Engineering Policies](#)

## 2.1.2.14 Automated MariaDB Deployment and Administration

It is possible to automate the deployment and administration of MariaDB servers and related technologies by using third-party software. This is especially useful when deploying and administering a large number of servers, but it also has benefits for small environments.

This section describes some automation technologies from MariaDB users perspective.



### Why to Automate MariaDB Deployments and Management

*The reasons to automate deployment and configuration of MariaDB.*



### A Comparison Between Automation Systems

*A summary of the differences between automation systems, to help evaluating them.*



### Ansible and MariaDB

*General information and hints on automating MariaDB deployments with Ansible.*



### Puppet and MariaDB

*General information on how to automate MariaDB deployments and configuration with Puppet.*



### Vagrant and MariaDB

*General information on how to setup development MariaDB servers with Vagrant.*



## Docker and MariaDB

[General information on how to setup MariaDB containers with Docker.](#)



## Kubernetes and MariaDB

[General information and tips on deploying MariaDB on Kubernetes.](#)



## Automating Upgrades with MariaDB.Org Downloads REST API

[How to use MariaDB.Org Downloads APIs to automate upgrades](#)



## HashiCorp Vault and MariaDB

[An overview of secret management with Vault for MariaDB users.](#)



## Orchestrator Overview

[Using Orchestrator to automate failover and replication operations.](#)



## Rotating Logs on Unix and Linux

[Rotating logs on Unix and Linux with logrotate.](#)



## Automating MariaDB Tasks with Events

[Using MariaDB events for automating tasks](#)

## 2.1.2.14.1 Why to Automate MariaDB Deployments and Management

MariaDB includes a powerful [configuration system](#). This is enough when we need to deploy a single MariaDB instance, or a small number of instances. But many modern organisations have many database servers. Deploying and upgrading them manually could require too much time, and would be error-prone.

### Contents

1. [Infrastructure as Code](#)
2. [Automated Failover](#)
3. [Resources](#)

## Infrastructure as Code

Several tools exist to deploy and manage several servers automatically. These tools operate at a higher level, and execute tasks like installing MariaDB, running queries, or generating new configuration files based on a template. Instead of upgrading servers manually, users can launch a command to upgrade a group of servers, and the automation software will run the necessary tasks.

Servers can be described in a code repository. This description can include MariaDB version, its configuration, users, backup jobs, and so on. This code is human-readable, and can serve as a documentation of which servers exist and how they are configured. The code is typically versioned in a repository, to allow collaborative development and track the changes that occurred over time. This is a paradigm called **Infrastructure as Code**.

Automation code is high-level and one usually doesn't care how operations are implemented. Their implementation is delegated to modules that handle specific components of the infrastructure. For example a module could equally work with apt and yum package managers. Other modules can implement operations for a specific cloud vendor, so we declare we want a snapshot to be done, but we don't need to write the commands to make it happen. For special cases, it is of course possible to write Bash commands, or scripts in every language, and declare that they must be run.

Manual interventions on the servers will still be possible. This is useful, for example, to investigate performance problems. But it is important to leave the servers in the state that is described by the code.

This code is not something you write once and never touch again. It is periodically necessary to modify infrastructures to update some software, add new replicas, and so on. Once the base code is in place, making such changes is often trivial and potentially it can be done in minutes.

## Automated Failover

Once [replication](#) is in place, two important aspects to automate are load balancing and failover.

Proxies can implement load balancing, redirecting the queries they receive to different server, trying to distribute the load equally. They can also monitor that MariaDB servers are running and in good health, thus avoiding sending queries to a server that is down or struggling.

However, this does not solve the problem with replication: if a primary server crashes, its replicas should point to another server. Usually this means that an existing replica is promoted to a master. This kind of changes are possible thanks to MariaDB [GTID](#).

One can promote a replica to a primary by making change to existing automation code. This is typically simple and relatively quick to do for a human operator. But this operation takes time, and in the meanwhile the service could be down.

Automating failover will minimise the time to recover. A way to do it is to use Orchestrator, a tool that can automatically promote a replica to a primary. The choice of the replica to promote is done in a smart way, keeping into account things like the servers versions and the [binary log](#) format.

## Resources

- [Continuous configuration automation on Wikipedia](#) .
- [Infrastructure as code on Wikipedia](#) .

Content initially contributed by [Vettibase Ltd](#) .

## 2.1.2.14.2 A Comparison Between Automation Systems

This page compares the automation systems that are covered by this section of the MariaDB Knowledge Base. More information about these systems are presented in the relevant pages, and more systems may be added in the future.

### Contents

1. [Code Structure Differences](#)
  1. [Ansible Code Structure](#)
  2. [Puppet Code Structure](#)
2. [Architectural Differences](#)
  1. [Ansible Architecture](#)
  2. [Puppet Architecture](#)
    1. [Agent-Master Architecture](#)
    2. [Standalone Architecture](#)
    3. [Inventory](#)
3. [Storing Secrets](#)
4. [Ecosystems and Communities](#)
  1. [Ansible Ecosystem](#)
  2. [Puppet Ecosystem](#)
5. [See Also](#)

## Code Structure Differences

Different automation systems provide different ways to describe our infrastructure. Understanding how they work is the first step to evaluate them and choose one for our organization.

### Ansible Code Structure

Ansible code consists of the following components:

- An **inventory** determines which **hosts** Ansible should be able to deploy. Each host may belong to one or more **groups** . Groups may have **children** , forming a hierarchy. This is useful because it allows us to deploy on a group, or to assign variables to a group.
- A **role** describes the state that a host, or group of hosts, should reach after a deploy.
- A **play** associates hosts or groups to their roles. Each role/group can have more than one role.
- A role consists of a list of **tasks** . Despite its name a task is not necessarily something to do, but something that must exist in a certain state.
- Tasks can use **variables** . They can affect how a task is executed (for example a variable could be a file name), or even whether a task is executed or not. Variables exist at role, group or host level. Variables can also be passed by the user when a play is applied.
- **Playbooks** are the code that is used to define tasks and variables.
- **Facts** are data that Ansible retrieves from remote hosts before deploying. This is a very important step, because facts may determine which tasks are executed or how they are executed. Facts include, for example, the operating system family or its version. A playbook sees facts as pre-set variables.
- **Modules** implement **actions** that tasks can use. Action examples are **file** (to declare that files and directories must exist) or **mysql\_variables** (to declare MySQL/MariaDB variables that need to be set).

See [Ansible Overview - concepts](#) for more details and an example.

### Puppet Code Structure

Puppet code consists of the following components:

- An **inventory file** defines a set of **groups** and their **targets** (the members of a group). **plugins** can be used to retrieve groups and target dynamically, so they are equivalent to Ansible dynamic inventories.
- A **manifest** is a file that describes a configuration.
- A **resource** is a component that should run on a server. For example, "file" and "service" are existing support types.
- An **attribute** relates to a resource and affects the way it is applied. For example, a resource of type "file" can have attributes like "owner" and "mode".
- A **class** groups resources and variables, describing a logical part of server configuration. A class can be associated to several servers. A class is part of a manifest.
- A **module** is a set of manifests and describes an infrastructure or a part of it.
- Classes can have typed **parameters** that affect how they are applied.
- **Properties** are variables that are read from the remote server, and cannot be arbitrarily assigned.
- **Facts** are pre-set variables collected by Puppet before applying or compiling a manifest.

## Architectural Differences

The architecture of the various systems is different. Their architectures determine how a deploy physically works, and what is needed to be able to deploy.

## Ansible Architecture

Ansible architecture is simple. Ansible can run from any host, and can apply its playbooks on remote hosts. To do this, it runs commands via SSH. In practice, in most cases the commands will be run as superuser via

```
sudo  
, though this is not always necessary.
```

Inventories can be dynamic. In this case, when we apply a playbook Ansible connects to remote services to discover hosts.

Ansible playbooks are applied via the

```
ansible-playbook  
binary. Changes to playbooks are only applied when we perform this operation.
```

To recap, Ansible does not need to be installed on the server it administers. It needs an SSH access, and normally its user needs to be able to run

```
sudo  
. It is also possible to configure a dynamic inventory, and a remote service to be used for this purpose.
```

## Puppet Architecture

Puppet supports two types of architecture: agent-master or standalone. The agent-master architecture is recommended by Puppet Labs, and it is the most popular among Puppet users. For this reason, those who prefer a standalone architecture tend to prefer Ansible.

### Agent-Master Architecture

When this architecture is chosen, manifests are sent to the **Puppet master**. There can be more than one master, for high availability reasons. All target hosts run a **Puppet agent**. Normally this is a service that automatically starts at system boot. The agent contacts a master at a given interval. It sends facts, and uses them to compile a **catalog** from the manifests. A catalog is a description of what exactly an individual server should run. The agent receives the catalog and checks if there are differences between its current configuration and the catalog. If differences are found, the agent applies the relevant parts of the catalog.

An optional component is **PuppetDB**. This is a central place where some data are stored, including manifests, retrieved facts and logs. PuppetDB is based on PostgreSQL and there are no plans to support MariaDB or other DBMSs.

If a manual change is made to a remote server, it will likely be overwritten the next time Puppet agent runs. To avoid this, the Puppet agent service can be stopped.

### Standalone Architecture

As mentioned, this architecture is not recommended by Puppet Labs nor popular amongst Puppet users. It is similar to Ansible architecture.

Users can apply manifests from any host with Puppet installed. This could be their laptop but, in order to emulate the behavior of an agent-master architecture, normally Puppet runs on a dedicated node as a cronjob. The **Puppet apply** application will require facts from remote hosts, it will compile a catalog for each host, will check which parts of it need to be applied, and will apply them remotely.

If a manual change is made to a remote server, it will be overwritten the next time Puppet apply runs. To avoid this, comment out any cron job running Puppet apply, or comment out the target server in the inventory.

### Inventory

As mentioned, Puppet supports plugins to retrieve the inventory dynamically from remote services. In an agent-master architecture, one has to make sure that each target host has access to these services. In a standalone architecture, one has to make sure that the hosts running Puppet apply have access to these services.

## Storing Secrets

Often our automation repositories need to contain secrets, like MariaDB user passwords or private keys for SSH authentication.

Both Ansible and Puppet support integration with secret stores, like Hashicorp Vault. For Puppet integration, see [Integrations with secret stores](#).

In the simplest case, Ansible allows encrypting secrets in playbooks and decrypting them during execution using **ansible-vault**. This implies a minimal effort to handle secrets. However, it is not the most secure way to store secrets. The passwords to disclose certain secrets need to be shared with the users who have the right to use them. Also, brute force attacks are possible.

## Ecosystems and Communities

Automation software communities are very important, because they make available a wide variety of modules to handle specific software.

### Ansible Ecosystem

Ansible is open source, released under the terms of the GNU GPL. It is produced by RedHat. RedHat has a page about [Red Hat Ansible Automation Platform Partners](#), who can provide support and consulting.

[Ansible Galaxy](#) is a big repository of Ansible roles produced by both the vendor and the community. Ansible comes with

`ansible-galaxy`

, a tool that can be used to create roles and upload them to Ansible Galaxy.

At the time of this writing, Ansible does not have specific MariaDB official modules. MySQL official modules can be used. However, be careful not try to use features that only apply to MySQL. There are several community-maintained MariaDB roles.

## Puppet Ecosystem

Puppet is open source, released under the GNU GPL. It is produced by a homonym company. The page [Puppet Partners](#) lists partners that can provide support and consulting about Puppet.

[Puppet Forge](#) is a big repository of modules produced by the vendor and by the community, as well as how-to guides.

Currently Puppet has many MariaDB modules.

## See Also

For more information about the systems mentioned in this page, from MariaDB users perspective:

- [Ansible and MariaDB](#) .
- [Puppet and MariaDB](#) .

---

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.3 Ansible and MariaDB

General information and hints on how to automate MariaDB deployments and configuration with Ansible.

Ansible is an open source tool to automate deployment, configuration and operations.



### Ansible Overview for MariaDB Users

*Overview of Ansible and how it works with MariaDB.*



### Deploying to Remote Servers with Ansible

*How to invoke Ansible to run commands or apply roles on remote hosts.*



### Deploying Docker Containers with Ansible

*How to deploy and manage Docker containers with Ansible.*



### Existing Ansible Modules and Roles for MariaDB

*Links to existing Ansible modules and roles for MariaDB.*



### Installing MariaDB .deb Files with Ansible

*How to install MariaDB from .deb files using Ansible.*



### Running mariadb-tzinfo-to-sql with Ansible

*Updating the timezone tables with mariadb-tzinfo-to-sql using Ansible.*



### Managing Secrets in Ansible

*How to store passwords as part of an Ansible repository.*

## 2.1.2.14.3.1 Ansible Overview for MariaDB Users

Ansible is a tool to automate servers configuration management. It is produced by Red Hat and it is open source software released under the terms of the GNU GPL.

It is entirely possible to use Ansible to automate MariaDB deployments and configuration. This page contains generic information for MariaDB users who want to learn, or evaluate, Ansible.

For information about how to install Ansible, see [Installing Ansible](#) in Ansible documentation.

### Contents

1. [Automation Hubs](#)
2. [Design Principles](#)
3. [Concepts](#)
  1. [Example](#)
4. [Architecture](#)
5. [Ansible Resources and References](#)

## Automation Hubs

Normally, Ansible can run from any computer that has access to the target hosts to be automated. It is not uncommon that all members of a team has Ansible installed on their own laptop, and use it to deploy.

Red Hat offers a commercial version of Ansible called [Ansible Tower](#). It consists of a REST API and a web-based interface that work as a hub that handles all normal Ansible operations.

An alternative is [AWX](#). AWX is the open source upstream project from which many Ansible Tower features are originally developed. AWX is released under the terms of the Apache License 2.0. However, Red Hat does not recommend to run AWX in production.

AWX development is fast. It has several features that may or may not end up in Ansible Tower. Ansible Tower is more focused on making AWS features more robust, providing a stable tool to automate production environments.

## Design Principles

Ansible allows us to write **playbooks** that describe how our servers should be configured. Playbooks are lists of **tasks**.

Tasks are usually **declarative**. You don't explain *how* to do something, you declare *what* should be done.

Playbooks are **idempotent**. When you apply a playbook, tasks are only run if necessary.

Here is a task example:

```
- name: Install Perl
  package:
    name: perl
    state: present
```

"Install Perl" is just a description that will appear on screen when the task is applied. Then we use the `package` module to declare that a package called "perl" should be installed. When we apply the playbook, if Perl is already installed nothing happens. Otherwise, Ansible installs it.

When we apply a playbook, the last information that appears on the screen is a recap like the following:

```
PLAY RECAP ****
mariadb-01 : ok=6    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

This means that six tasks were already applied (so no action was taken), and two tasks were applied.

As the above example shows, Ansible playbooks are written in YAML.

Modules (like

`package`) can be written in any language, as long as they are able to process a JSON input and produce a JSON output. However the Ansible community prefers to write them in Python, which is the language Ansible is written in.

## Concepts

A piece of Ansible code that can be applied to a server is called a **playbook**.

A **task** is the smallest brick of code in a playbook. The name is a bit misleading, though, because an Ansible task should not be seen as "something to do". Instead, it is a minimal description of a component of a server. In the example above, we can see a task.

A task uses a single **module**, which is an interface that Ansible uses to interact with a specific system component. In the example, the module is "package".

A task also has attributes, that describe what should be done with that module, and how. In the example above, "name" and "state" are both tasks. The `state`

attribute exists for every module, by convention (though there may be exceptions). Typically, it has at least the "present" and "absent" state, to indicate if an object should exist or not.

Other important code concepts are:

- An **inventory** determines which **hosts** Ansible should be able to deploy. Each host may belong to one or more **groups**. Groups may have **children**, forming a hierarchy. This is useful because it allows us to deploy on a group, or to assign variables to a group.
- A **role** describes the state that a host, or group of hosts, should reach after a deploy.
- A **play** associates hosts or groups to their roles. Each role/group can have more than one role.
- A **playbook** is a playbook that describes how certain servers should be configured, based on the logical role they have in the infrastructure. Servers can have multiple roles, for example the same server could have both the "mariadb" and the "mydumper" role, meaning that they run MariaDB and they have mydumper installed (as shown later).
- Tasks can use **variables**. They can affect how a task is executed (for example a variable could be a file name), or even whether a task is executed or not. Variables exist at role, group or host level. Variables can also be passed by the user when a play is applied.
- **Facts** are data that Ansible retrieves from remote hosts before deploying. This is a very important step, because facts may determine which tasks are executed or how they are executed. Facts include, for example, the operating system family or its version. A playbook sees facts as pre-set variables.
- **Modules** implement **actions** that tasks can use. Action examples are **file** (to declare that files and directories must exist) or **mysql\_variables** (to declare MySQL/MariaDB variables that need to be set).

## Example

Let's describe a hypothetical infrastructure to find out how these concepts can apply to MariaDB.

The **inventory** could define the following groups:

- "db-main" for the cluster used by our website. All nodes belong to this group.
- "db-analytics" for our replicas used by data analysts.
- "dump" for one or more servers that take dumps from the replicas.
- "proxysql" for one or more hosts that run ProxySQL.

Then we'll need the following nodes:

- "mariadb-node" for the nodes in "db-main". This role describes how to setup nodes of a cluster using Galera.
- "mariadb-replica" for the members of "db-analytics". It describes a running replica, and it includes the tasks that are necessary to provision the node if the data directory is empty when the playbook is applied. The hostname of the primary server is defined in a variable.
- "mariadb". The aforementioned "mariadb-node" and "mariadb-replica" can be children of this group. They have many things in common (filesystem for the data directory, some basic MariaDB configuration, some installed tools...), so it could make sense to avoid duplication and describe the common traits in a super-role.
- A "mariabackup" role to take backups with [Mariabackup](#), running jobs during the night. We can associate this role to the "db-main" group, or we could create a child group for servers that will take the backups.
- "mariadb-dump" for the server that takes dumps with [mariadb-dump](#). Note that we may decide to take dumps on a replica, so the same host may belong to "db-analytics" and "mariadb-dump".
- "proxysql" for the namesake group.

## Architecture

Ansible architecture is extremely simple. Ansible can run on any host. To apply playbooks, it connects to the target hosts and runs system commands. By default the connection happens via ssh, though it is possible to develop connection plugins to use different methods. Applying playbooks locally without establishing a connection is also possible.

Modules can be written in any language, though Python is the most common choice in the Ansible community. Modules receive JSON "requests" and facts from Ansible core, they are supposed to run useful commands on a target host, and then they should return information in JSON. Their output informs Ansible whether something has changed on the remote server and if the operations succeeded.

Ansible is not centralized. It can run on any host, and it is common for a team to run it from several laptops. However, to simplify things and improve security, it may be desirable to run it from a dedicated host. Users will connect to that host, and apply Ansible playbooks.

## Ansible Resources and References

- [Ansible.com](#)
- [AWX](#)
- [Ansible Tower](#)
- [Ansible Galaxy](#)
- [Ansible on Wikipedia](#)
- [Ansible Automation Platform](#) YouTube channel
- [Ansible: Getting Started](#)
- [MariaDB Deployment and Management with Ansible](#) (video)

Further information about the concepts discussed in this page can be found in Ansible documentation:

- [Basic Concepts](#).
- [Glossary](#).

---

Content initially contributed by [Vettibase Ltd](#).

## 2.1.2.14.3.2 Deploying to Remote Servers with Ansible

If we manage several remote servers, running commands on them manually can be frustrating and time consuming. Ansible allows one to run commands on a whole group of servers.

This page shows some examples of ansible-playbook invocations. We'll see how to deploy roles or parts of them to remote servers. Then we'll see how to run commands on remote hosts, and possibly to get information from them. Make sure to read [Ansible Overview](#) first, to understand Ansible general concepts.

### Contents

1. [Pinging Remote Servers](#)
2. [Running Commands on Remote Servers](#)
3. [Applying Roles to Remote Servers](#)
  1. [Check mode](#)
4. [References](#)

# Pinging Remote Servers

Let's start with the simplest example: we just want our local Ansible to ping remote servers to see if they are reachable. Here's how to do it:

```
ansible -i production-mariadb all -m ping
```

Before proceeding with more useful examples, let's discuss this syntax.

- **ansible** is the executable we can call to run a command from remote servers.
- **-i production-mariadb** means that the servers must be read from an inventory called production-mariadb.
- **all** means that the command must be executed against all servers from the above inventory.
- **-m ping** specifies that we want to run the ping module. This is not the ping Linux command. It tells us if Ansible is able to connect a remote server and run a simple commands on them.

To run ping on a specific group or host, we can just replace "all" with a group name or host name from the inventory:

```
ansible -i production-mariadb main_cluster -m ping
```

## Running Commands on Remote Servers

The previous examples show how to run an Ansible module on remote servers. But it's also possible to run custom commands over SSH. Here's how:

```
ansible -i production-mariadb all -a 'echo $PATH'
```

This command shows the value of

\$PATH  
on all servers in the inventory "production-mariadb".

We can also run commands as root by adding the

-b  
(or  
--become  
) option:

```
# print a MariaDB variable
ansible -i production-mariadb all -b -a 'mysql -e "SHOW GLOBAL VARIABLES LIKE \'innodb_buffer_pool_size\';"'

# reboot servers
ansible -i production-mariadb all -b -a 'reboot'
```

## Applying Roles to Remote Servers

We saw how to run commands on remote hosts. Applying roles to remote hosts is not much harder, we just need to add some information. An example:

```
ansible-playbook -i production-mariadb production-mariadb.yml
```

Let's see what changed:

- **ansible-playbook** is the executable file that we need to call to apply playbooks and roles.
- **production-mariadb.yml** is the play that associates the servers listed in the inventory to their roles.

If we call ansible-playbook with no additional arguments, we will apply all applicable roles to all the servers mentioned in the play.

To only apply roles to certain servers, we can use the

-l  
parameter to specify a group, an individual host, or a pattern:

```
# Apply to the mariadb-main role role
ansible-playbook -i production-mariadb -l mariadb-main production-mariadb.yml

# Apply to the mariadb-main-01 host
ansible-playbook -i production-mariadb -l mariadb-main-01 production-mariadb.yml

# Apply to multiple hosts whose name starts with "mariadb-main-"
ansible-playbook -i production-mariadb -l mariadb-main-* production-mariadb.yml
```

We can also apply tasks from roles selectively. Tasks may optionally have tags, and each tag corresponds to an operation that we may want to run on our remote hosts. For example, a "mariadb" role could have the "timezone-update" tag, to update the contents of the [timezone tables](#). To only apply the tasks with the "timezone-update" tag, we can use this command:

```
ansible-playbook -i production-mariadb --tag timezone-update production-mariadb.yml
```

Using tags is especially useful for database servers. While most of the technologies typically managed by Ansible are stateless (web servers, load balancers, etc.) database servers are not. We must pay special attention not to run tasks that could cause a database server outage, for example destroying its data directory or restarting the service when it is not necessary.

## Check mode

We should always test our playbooks and roles on test servers before applying them to production. However, if test servers and production servers are not exactly in the same state (which means, some facts may differ) it is still possible that applying roles will fail. If it fails in the initial stage, Ansible will not touch the remote hosts at all. But there are cases where Ansible could successfully apply some tasks, and fail to apply another task. After the first failure, `ansible-playbook` will show errors and exit. But this could leave a host in an inconsistent state.

Ansible has a *check mode* that is meant to greatly reduce the chances of a failure. When run in check mode, `ansible-playbook` will read the inventory, the play and roles; it will figure out which tasks need to be applied; then it will connect to target hosts, read facts, and value all the relevant variables. If all these steps succeed, it is unlikely that running `ansible-playbook` without check mode will fail.

To run `ansible-playbook` in check mode, just add the

```
--check  
(or  
-C  
) parameter.
```

## References

Further documentation can be found in the Ansible website:

- [ansible tool](#).
- [ansible-playbook tool](#).
- [Validating tasks: check mode and diff mode](#) .

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.3.3 Deploying Docker Containers with Ansible

Ansible can be used to manage Docker container upgrades and configuration changes. Docker has native ways to do this, namely [Dockerfiles](#) and [Docker Compose](#) . But sometimes there are reasons to start basic containers from an image and then manage configuration with Ansible or similar software. See [Benefits of Managing Docker Containers with Automation Software](#) .

In this page we'll discuss how to use Ansible to manage Docker containers.

### Contents

1. [How to Deploy a Container with Ansible](#)
2. [References](#)

## How to Deploy a Container with Ansible

Ansible has modules to manage the Docker server, Docker containers, and Docker Compose. These modules are maintained by the community.

A dynamic inventory plugin for Docker exists. It retrieves the list of existing containers from Docker.

Docker modules and the Docker inventory plugin communicate with Docker using its API. The connection to the API can use a TSL connection and supports key authenticity verification.

To communicate with Docker API, Ansible needs a proper Python module installed on the Ansible node (

```
docker  
or  
docker-py  
().
```

Several roles exist to deploy Docker and configure it. They can be found in Ansible Galaxy.

## References

Further information can be found in Ansible documentation.

- [Docker Guide](#) .
- [docker\\_container module](#).

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.3.4 Installing MariaDB .deb Files with Ansible

This page refers to the operations described in [Installing MariaDB .deb Files](#). Refer to that page for a complete list and explanation of the tasks that should be performed.

Here we discuss how to automate such tasks using Ansible. For example, here we show how to install a package or how to import a GPG key; but for an updated list of the necessary packages and for the keyserver to use, you should refer to [Installing MariaDB .deb Files](#).

### Adding apt Repositories

To add a repository :

```
- name: Add specified repository into sources list
ansible.builtin.apt_repository:
  repo: deb [arch=amd64,arm64,ppc64el] http://sfo1.mirrors.digitalocean.com/mariadb/repo/10.3/ubuntu bionic main
  state: present
```

If you prefer to keep the repository information in a [source list file](#) in the Ansible repository, you can upload that file to the target hosts in this way:

```
- name: Create a symbolic link
ansible.builtin.file:
  src: ./file/mariadb.list
  dest: /etc/apt/sources.list.d/
  owner: root
  group: root
  mode: 644
  state: file
```

### Updating the Repository Cache

Both the Ansible modules `ansible.builtin.apt` and `ansible.builtin.apt_repository` have an

`update_cache` attribute. In `ansible.builtin.apt` it is set to "no" by default. Whenever a task sets it to 'yes', `apt-get update` is run on the target system. You have three ways to make sure that repositories are updated.

The first is to use `ansible.builtin.apt_repository` to add the desired repository, as shown above. So you only need to worry about updating repositories if you use the file method.

The second is to make sure that

`update_cache` is set to 'yes' when you install a repository:

```
- name: Install foo
apt:
  name: foo
  update_cache: yes
```

But if you run certain tasks conditionally, this option may not be very convenient. So the third option is to update the repository cache explicitly as a separate task:

```
- name: Update repositories
apt:
  - update_cache: yes
```

### Importing MariaDB GPG Key

To [import the GPG key](#) for MariaDB we can use the `ansible.builtin.apt_key` Ansible module. For example:

```
- name: Add an apt key by id from a keyserver
ansible.builtin.apt_key:
  keyserver: hkp://keyserver.ubuntu.com:80
  id: 0xF1656F24C74CD1D8
```

### Installing Packages

To install Deb packages into a system:

```
- name: Install software-properties-common
  apt:
    name: software-properties-common
    state: present
```

To make sure that a specific version is installed, performing an upgrade or a downgrade if necessary:

```
- name: Install foo 1.0
  apt:
    name: foo=1.0
```

To install a package or upgrade it to the latest version, use:

```
state: latest
```

To install multiple packages at once:

```
- name: Install the necessary packages
  apt:
    pkg:
      - pkg1
      - pkg2=1.0
```

If all your servers run on the same system, you will always use `ansible.builtin.apt` and the names and versions of the packages will be the same for all servers. But suppose you have some servers running systems from the Debian family, and others running systems from the Red Hat family. In this case, you may find convenient to use two different task files for two different types of systems. To include the proper file for the target host's system:

```
- include: mariadb-debian.yml
when: "{{ ansible_facts['os_family'] }} == 'Debian'
```

The variables you can use to run the tasks related to the proper system are:

- `ansible_fact['distribution']`
- `ansible_fact['distribution_major_version']`
- `ansible_fact['os_family']`

There is also a system-independent [package module](#), but if the package names depend on the target system using it may be of very little benefit.

## See Also

- [Installing MariaDB .deb Files](#)

---

Content initially contributed by [Vettabase Ltd](#).

### 2.1.2.14.3.5 Running mariadb-tzinfo-to-sql with Ansible

For documentation about the

`mariadb-tzinfo-to-sql` utility, see [mysql\\_tzinfo\\_to\\_sql](#). This page is about running it using Ansible.

## Installing or Upgrading the Package

First, we should install

`mariadb-tzinfo-to-sql`

if it is available on our system. For example, to install it on Ubuntu, we can use this task. For other systems, use the proper module and package name.

```
- name: Update timezone info
  tags: [ timezone-update ]
  apt:
    name: tzdata
    state: latest
    install_recommends: no
  register: timezone_info
```

This task installs the latest version of the

`tzdata`

, unless it is already installed and up to date. We register the

`timezone_info`

variables, so we can only run the next task if the package was installed or updated.

We also specify a

`timezone-update`  
tag, so we can apply the role to only update the timezone tables.

## Running the Script

The next task runs

```
mariadb-tzinfo-to-sql
```

```
- name: Move system timezone info into MariaDB
tags: [ timezone-update ]
shell: >
  mysql_tzinfo_to_sql /usr/share/zoneinfo \
    | grep -v "Warning" \
    | mysql --database=mysql
when: timezone_info.changed
```

We use the

```
shell
module to run the command. Running a command in this way is not idempotent, so we specify
when: timezone_info.changed
to only run it when necessary. Some warnings may be generated, so we pipe the output of
mysql_tzinfo_to_sql
to
grep
to filter warnings out.
```

## Using Galera

If we're using [MariaDB Galera Cluster](#) we'll want to only update the timezone tables in one node, because the other nodes will replicate the changes. For our convenience, we can run this operation on the first node. If the nodes hostnames are defined in a list called

```
cluster_hosts
, we can check if the current node is the first in this way:
```

```
when: timezone_info.changed and inventory_hostname == cluster_hosts[0].hostname
```

---

Content initially contributed by [Vettabase Ltd](#).

### 2.1.2.14.3.6 Managing Secrets in Ansible

An Ansible role often runs commands that require certain privileges, so it must perform some forms of login, using passwords or key pairs. In the context of database automation, we normally talk about: SSH access, sudo access, and access to MariaDB. If we write these secrets (passwords or private keys) in clear text in an Ansible repository, anyone who has access to the repository can access them, and this is not what we want.

Let's see how we can manage secrets.

#### Contents

1. [The SSH Password or Keys](#)
2. [Avoiding Sharing Secrets](#)
3. [ansible-vault](#)

## The SSH Password or Keys

Most of the times, Ansible connects to the target hosts via SSH. It is common to use the system username and the SSH keys installed in

```
/.ssh
```

, which is the SSH clients default. In this case, nothing has to be done on the clients to be able to allow Ansible to use SSH, as long as they are already able to connect to the target hosts.

It is also possible to specify a different username as

```
ANSIBLE_REMOTE_USER
```

and an SSH configuration file as

```
ANSIBLE_NETCONF_SSH_CONFIG
```

. These settings can be specified in Ansible configuration file or as environment variables.

#### [ANSIBLE\\_ASK\\_PASS](#)

can be specified. If this is the case, Ansible will prompt the user asking to type an SSH password.

## Avoiding Sharing Secrets

As a general rule, any configuration that implies communicating sensible information to the persons who will connects to a system implies some degree of risk. Therefore, the most common choice is to allow users to login into remote systems with their local usernames, using SSH keys.

Once Ansible is able to connect remote hosts, it can also be used to install the public keys of some users to grant them access. Sharing these keys implies no risk. Sharing private keys is never necessary, and must be avoided.

MariaDB has a [UNIX\\_SOCKET](#) plugin that can be used to let some users avoid entering a password, as far as they're logged in the operating system. This authentication method is used by default for the root user. This is a good way to avoid having one more password and possibly writing to a

`.my.cnf`  
file so that the user doesn't have to type it.

Even for users who connect remotely, it is normally not necessary to insert passwords in an Ansible file. When we create a user with a password, a hash of the original password is stored in MariaDB. That hash can be found in the [mysql.user table](#). To know the hash of a password without even creating a user, we can use the [PASSWORD\(\)](#) function:

```
SELECT PASSWORD('my_password12') AS hash;
```

When we create a user, we can actually specify a hash instead of the password the user will have to type:

```
CREATE USER user@host IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECB71D01F08549980';
```

## ansible-vault

Even if you try to avoid sharing secrets, it's likely you'll have to keep some in Ansible. For example, MariaDB users that connect remotely have passwords, and if we want Ansible to create and manage those users, the hashes must be placed somewhere in our Ansible repository. While a hash cannot be converted back to a password, treating hashes as secrets is usually a good idea. Ansible provides a native way to handle secrets: [ansible-vault](#).

In the simplest case, we can manage all our passwords with a single ansible-vault password. When we add or change a new password in some file (typically a file in

`host_vars`  
or  
`group_vars`

) we'll use ansible-vault to crypt this password. While doing so, we'll be asked to insert our ansible-vault password. When we apply a role and Ansible needs to decrypt this password, it will ask us to enter again our ansible-vault password.

ansible-vault can use more than one password. Each password can manage a different set of secrets. So, for example, some users may have the password to manage regular MariaDB users passwords, and only one may have the password that is needed to manage the root user.

---

Content initially contributed by [Vettabase Ltd](#).

### 2.1.2.14.4 Puppet and MariaDB

General information and hints on how to automate MariaDB deployments and configuration with Puppet.

Puppet is an open source tool deployment, configuration and operations.



#### **Puppet Overview for MariaDB Users**

[Overview of Puppet and how it works with MariaDB.](#)



#### **Bolt Examples**

[How to invoke Bolt to run commands or apply roles on remote hosts.](#)



#### **Puppet hiera Configuration System**

[Using hiera to handle Puppet configuration files.](#)



#### **Deploying Docker Containers with Puppet**

[How to deploy and manage Docker containers with Puppet.](#)



#### **Existing Puppet Modules for MariaDB**

[Links to existing Puppet modules for MariaDB.](#)

## 2.1.2.14.4.1 Puppet Overview for MariaDB Users

Puppet is a tool to automate servers configuration management. It is produced by Puppet Inc, and released under the terms of the Apache License, version 2.

It is entirely possible to use Ansible to automate MariaDB deployments and configuration. This page contains generic information for MariaDB users who want to learn, or evaluate, Puppet.

Puppet modules can be searched using [Puppet Forge](#). Most of them are also published on GitHub with open source licenses. Puppet Forge allows filtering modules to only view the most reliable: supported by Puppet, supported by a Puppet partner, or approved.

For information about installing Puppet, see [Installing and upgrading](#) in Puppet documentation.

### Contents

- 1. [Design Principles](#)
  - 1. [Defining Resources](#)
  - 2. [Defining Nodes](#)
- 2. [Concepts](#)
- 3. [Architecture](#)
  - 1. [Agent-master Architecture](#)
  - 2. [Standalone Architecture](#)
  - 3. [PuppetDB](#)
  - 4. [External Node Classifiers](#)
  - 5. [Bolt](#)
- 4. [hiera](#)
- 5. [Puppet Resources](#)

## Design Principles

With Puppet, you write **manifests** that describe the resources you need to run on certain servers and their **attributes**.

Therefore manifests are **declarative**. You don't write the steps to achieve the desired result. Instead, you describe the desired result. When Puppet detects differences between your description and the current state of a server, it decides what to do to fix those differences.

Manifests are also **idempotent**. You don't need to worry about the effects of applying a manifest twice. This may happen (see Architecture below) but it won't have any side effects.

## Defining Resources

Here's an example of how to describe a resource in a manifest:

```
file { '/etc/motd':  
  content => '',  
  ensure => present,  
}
```

This block describes a resource. The resource type is

file  
, while the resource itself is  
/etc/motd  
. The description consists of a set of attributes. The most important is  
ensure  
, which in this case states that the file must exist. It is also common to use this resource to indicate that a file (probably created by a previous version of the manifest) doesn't exist.

These classes of resource types exist:

- **Built-in resources**, or **Puppet core resources**: Resources that are part of Puppet, maintained by the Puppet team.
- **Defined resources**: Resources that are defined as a combination of other resources. They are written in the Puppet domain-specific language.
- **Custom resources**: Resources that are written by users, in the Ruby language.

To obtain information about resources:

```
# list existing resource types  
puppet resource --types  
# print information about the file resource type  
puppet describe file
```

To group several resources in a reusable class:

```

class ssh_server {
  file { '/etc/motd':
    content => '',
    ensure => present,
  }
  file { '/etc/issue.net':
    content => '',
    ensure => present,
  }
}

```

There are several ways to include a class. For example:

```
include Class['ssh_server']
```

## Defining Nodes

Puppet has a **main manifest** that could be a

- site.pp
- file or a directory containing
- .pp

files. For simple infrastructures, we can define the nodes here. For more complex infrastructures, we may prefer to import other files that define the nodes.

Nodes are defined in this way:

```

node 'maria-1.example.com' {
  include common
  include mariadb
}

```

The resource type is

- node
- . Then we specify a hostname that is used to match this node to an existing host. This can also be a list of hostnames, a regular expression that matches multiple nodes, or the
- default
- keyword that matches all hosts. To use a regular expression:

```

node /^(maria|mysql)-[1-3]\.example\.com$/ {
  include common
}

```

## Concepts

The most important Puppet concepts are the following:

- **Target** : A host whose configuration is managed via Puppet.
- **Group** : A logical group of targets. For example there may be a
  - mariadb
  - group, and several targets may be part of this group.
- **Facts** : Information collected from the targets, like the system name or system version. They're collected by a Ruby gem called [Facter](#) . They can be [core facts](#) (collected by default) or [custom facts](#) (defined by the user).
- **Manifest** : A description that can be applied to a target.
- **Catalog** : A compiled manifest.
- **Apply** : Modifying the state of a target so that it reflects its description in a manifest.
- **Module** : A set of manifests.
- **Resource** : A minimal piece of description. A manifest consists of a piece of resources, which describe components of a system, like a file or a service.
- **Resource type** : Determines the class of a resource. For example there is a
  - file
  - resource type, and a manifest can contain any number of resources of this type, which describe different files.
- **Attribute** : It's a characteristic of a resource, like a file owner, or its mode.
- **Class** : A group of resources that can be reused in several manifests.

## Architecture

Depending on how the user decides to deploy changes, Puppet can use two different architectures:

- An **Agent-master** architecture. This is the preferred way to use Puppet.
- A **standalone architecture** , that is similar to [Ansible architecture](#) .

## Agent-master Architecture

A **Puppet master** stores a catalog for each target. There may be more than one Puppet master, for redundancy.

Each target runs a **Puppet agent** in background. Each Puppet agent periodically connects to the Puppet master, sending its facts. The Puppet master compiles the relevant manifest using the facts it receives, and send back a catalog. Note that it is also possible to store the catalogs in PuppetDB instead.

Once the Puppet agent receives the up-to-date catalog, it checks all resources and compares them with its current state. It applies the necessary changes to make sure that its state reflects the resources present in the catalog.

## Standalone Architecture

With this architecture, the targets run **Puppet apply**. This application usually runs as a Linux cron job or a Windows scheduled task, but it can also be manually invoked by the user.

When Puppet apply runs, it compiles the latest versions of manifests using the local facts. Then it checks every resource from the resulting catalogs and compares it to the state of the local system, applying changes where needed.

Newly created or modified manifests are normally deployed to the targets, so Puppet apply can read them from the local host. However it is possible to use PuppetDB instead.

## PuppetDB

PuppetDB is a Puppet node that runs a PostgreSQL database to store information that can be used by other nodes. PuppetDB can be used with both the Agent-master and the standalone architectures, but it is always optional. However it is necessary to use some advanced Puppet features.

PuppetDB stored the following information:

- The latest facts from each target.
- The latest catalogs, compiled by Puppet apply or a Puppet master.
- Optionally, the recent history of each node activities.

## External Node Classifiers

With both architectures, it is possible to have a component called an External Node Classifier (ENC). This is a script or an executable written in any language that Puppet can call to determine the list of classes that should be applied to a certain target.

An ENC received a node name in input, and should return a list of classes, parameters, etc, as a YAML hash.

## Bolt

Bolt can be used in both architectures to run operations against a target or a set of targets. These operations can be commands passed manually to Bolt, scripts, Puppet tasks or plans. Bolt directly connects to targets via ssh and runs system commands.

See [Bolt Examples](#) to get an idea of what you can do with Bolt.

## hiera

hiera is a hierarchical configuration system that allows us to:

- Store configuration in separate files;
- Include the relevant configuration files for every server we automate with Puppet.

See [Puppet hiera Configuration System](#) for more information.

## Puppet Resources

- [Puppet documentation](#).
- [forge.puppet.com](#).
- [Puppet on GitHub](#).
- [Puppet on Wikipedia](#).

More information about the topics discussed in this page can be found in the Ansible documentation:

- [Puppet Glossary](#) in Puppet documentation.
- [Overview of Puppet's architecture](#) in Puppet documentation.
- [PuppetDB documentation](#).
- [Classifying nodes](#) in Puppet documentation.
- [Hiera](#) in Puppet documentation.
- [Bolt documentation](#).

## 2.1.2.14.4.2 Bolt Examples

### Contents

1. [Inventory Files](#)
2. [Running Commands on Targets](#)
3. [Copying Files](#)
4. [Running Scripts on Targets](#)
5. [Running Tasks on Targets](#)
6. [Applying Puppet Code on Targets](#)
7. [Bolt Resources and References](#)

This page shows some examples of what we can do with Bolt to administer a set of MariaDB servers. Bolt is a tool that is part of the [Puppet](#) ecosystem.

For information about installing Bolt, see [Installing Bolt](#) in Bolt documentation.

## Inventory Files

The simplest way to call Bolt and instruct it to do something on some remote targets is the following:

```
bolt ... --nodes 100.100.100.100,200.200.200.200,300,300,300,300
```

However, for non-trivial setups it is usually better to use an inventory file. An example:

```
targets:  
  - uri: maria-1.example.com  
    name: maria_1  
    alias: mariadb_main  
  ...
```

In this way, it will be possible to refer the target by name or alias.

We can also define groups, followed by the group members. For example:

```
groups:  
  - name: mariadb-staging  
    targets:  
      - uri: maria-1.example.com  
        name: maria_1  
      - uri: maria-2.example.com  
        name: maria_2  
  - name: mariadb-production  
    targets:  
      ...  
  ...
```

With an inventory of this type, it will be possible to run Bolt actions against all the targets that are members of a group:

```
bolt ... --nodes mariadb-staging
```

In the examples in the rest of the page, the

```
--targets
```

parameter will be indicated in this way, for simplicity:

```
--targets <targets>
```

## Running Commands on Targets

The simplest way to run a command remotely is the following:

```
bolt command run 'mariadb-admin start-all-slaves' --targets <targets>
```

## Copying Files

To copy a file or a whole directory to targets:

```
bolt file upload /path/to/source /path/to/destination --targets <targets>
```

To copy a file or a whole directory from the targets to the local host:

```
bolt file download /path/to/source /path/to/destination --targets <targets>
```

## Running Scripts on Targets

We can use Bolt to run a local script on remote targets. Bolt will temporarily copy the script to the targets, run it, and delete it from the targets. This is convenient for scripts that are meant to only run once.

```
bolt script run rotate_logs.sh --targets <targets>
```

## Running Tasks on Targets

Puppet tasks are not always as powerful as custom scripts, but they are simpler and many of them are idempotent. The following task stops MariaDB replication:

```
bolt task run mysql::sql --targets <targets> sql="STOP REPLICA"
```

## Applying Puppet Code on Targets

It is also possible to apply whole manifests or portions of Puppet code (resources) on the targets.

To apply a manifest:

```
bolt apply manifests/server.pp --targets <targets>
```

To apply a resource description:

```
bolt apply --execute "file { '/etc/mysql/my.cnf': ensure => present }" --targets <targets>
```

## Bolt Resources and References

- [Bolt documentation](#).
- [Bolt on GitHub](#).

Further information about the concepts explained in this page can be found in Bolt documentation:

- [Inventory Files](#) in Bolt documentation.
- [Applying Puppet code](#) in Bolt documentation.

Content initially contributed by [Vettabase Ltd](#).

## 2.1.2.14.4.3 Puppet hiera Configuration System

hiera is part of [Puppet](#). It is a hierarchical configuration system that allows us to:

- Store configuration in separate files;
- Include the relevant configuration files for every server we automate with Puppet.

### Contents

1. [hiera Configuration Files](#)
2. [Configuration files](#)

## hiera Configuration Files

Each hierarchy allows to one choose the proper configuration file for a resource, based on certain criteria. For example criteria may include node names, node groups, operating systems, or datacenters. Hierarchies are defined in a

```
hiera.yaml  
file, which also defines a path for the files in each hierarchy.
```

Puppet facts are commonly used to select the proper files to use. For example, a path may be defined as

```
"os/%{facts.os.name}.yaml"
```

. In this case, each resource will use a file named after the operating system it uses, in the os directory. You may need to use custom facts, for example to check which microservices will use a MariaDB server, or in which datacenter it runs.

We do not have to create a file for each possible value of a certain fact. We can define a default configuration file with settings that are reasonable for most resources. Other files, when included, will override some of the default settings.

A hiera configuration file will look like this:

```
version: 5
defaults:
  datadir: global
  data_hash: yaml_data

hierarchy:
  - name: "Node data"
    path: "nodes/%{trusted.certname}.yaml"

  - name: "OS data"
    path: "os/%{facts.os.family}.yaml"

  - name: "Per-datacenter business group data" # Uses custom facts.
    path: "location/%{facts.whereami}/%{facts.group}.yaml"
```

This file would include the global files, the OS-specific files and the node-specific files. Each hierarchy will override settings from previous hierarchies.

We can actually have several hiera configuration files.

`hiera.yaml`

is the global file. But we will typically have additional hiera configuration files for each environment. So we can include the configuration files that apply to production, staging, etc, plus global configuration files that should be included for every environment.

Importantly, we can also have hiera configuration files for each module. So, for example, a separate

`mariadb/hiera.yaml`

file may defined the hierarchies for MariaDB servers. This allow us to define, for example, different configuration files for MariaDB and for MaxScale, as most of the needed settings are typically different.

## Configuration files

You probably noticed that, in the previous example, we defined

`data_hash: yaml_data`

, which indicates that configuration files are written in YAML. Other allowed formats are JSON and HOCON. The

`data_hash`

setting is defined in

`defaults`

, but it can be overridden by hierarchies.

---

Content initially contributed by [Vettabase Ltd](#).

### 2.1.2.14.4 Deploying Docker Containers with Puppet

Puppet can also be used to manage Docker container upgrades and configuration changes. Docker has more specific tools for this purpose, but sometimes there are reasons to choose alternatives. See [Benefits of Managing Docker Containers with Automation Software](#).

In this page you will find out what managing Docker with Puppet looks like. All the snippets in this page use the

`docker`

resource type, supported by the Puppet company.

#### Contents

1. [How to Install, Upgrade or Uninstall Docker with Puppet](#)
2. [How to Build or Pull Docker Images with Puppet](#)
3. [How to Deploy Containers with Puppet](#)
4. [References](#)

### How to Install, Upgrade or Uninstall Docker with Puppet

Installing or upgrading Docker is simple:

```
class { 'docker':
  use_upstream_package_source => false,
  version => '17.09.0~ce-0~debian',
}
```

In this example we are using our system's repositories instead of Docker official repositories, and we are specifying the desired version. To upgrade

Docker later, all we need to do is to modify the version number. While specifying a version is not mandatory, it is a good idea because it makes our manifest more reproducible.

To uninstall Docker:

```
class { 'docker':
  ensure => absent
}
```

Check the

docker resource type documentation to find out how to use more features: for example you can use Docker Enterprise Edition, or bind the Docker daemon to a TCP port.

## How to Build or Pull Docker Images with Puppet

To pull an image from Dockerhub:

```
docker::image { 'mariadb:10.0': }
```

We specified the

10.0 tag to get the desired MariaDB version. If we don't, the image with the latest tag will be used. Note that this is not desirable in production, because it can lead to unexpected upgrades.

You can also write a Dockerfile yourself, and then build it to create a Docker image. To do so, you need to instruct Puppet to copy the Dockerfile to the target and then build it::

```
file { '/path/to/remote/Dockerfile':
  ensure => file,
  source => 'puppet:///path/to/local/Dockerfile',
}

docker::image { 'image_name':
  docker_file => '/path/to/remote/Dockerfile'
}
```

It is also possible to subscribe to Dockerfile changes, and automatically rebuild the image whenever a new file is found:

```
docker::image { 'image_name':
  docker_file => '/path/to/remote/Dockerfile'
  subscribe => File['/path/to/remote/Dockerfile'],
}
```

To remove an image that was possibly built or pulled:

```
docker::image { 'mariadb':
  ensure => absent
}
```

## How to Deploy Containers with Puppet

To run a container:

```
docker::run { 'mariadb-01':
  image    => 'mariadb:10.5',
  ports    => ['3306:6606']
}
```

mariadb-01 is the contained name. We specified the optional 10.5 tag, and we mapped the guest port 3306 to the host port 6606. In production, you normally don't map ports because you don't need to connect MariaDB clients from the host system to MariaDB servers in the containers. Third-party tools can be installed as separate containers.

## References

- [docker resource type documentation](#) , in Puppet documentation.

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.4.5 Existing Puppet Modules for MariaDB

This page contains links to Puppet modules that can be used to automate MariaDB deployment and configuration. The list is not meant to be exhaustive. Use it as a starting point, but then please do your own research.

### Contents

1. [Puppet Forge](#)
2. [Acceptance Tests](#)
3. [Supported Modules for MariaDB](#)
4. [Resources and References](#)

## Puppet Forge

Puppet Forge is the website to search for Puppet modules, maintained by the Puppet company. Modules are searched by the technology that needs to be automated, and the target operating system.

Search criteria include whether the modules are supported by Puppet or its partners, and whether a module is approved by Puppet. Approved modules are certified by Puppet based on their quality and maintenance standards.

## Acceptance Tests

Some modules that support the Puppet Development Kit allow some types of acceptance tests.

We can run a static analysis on a module's source code to find certain bad practices that are likely to be a source of bugs:

```
pdk validate
```

If a module's authors wrote unit tests, we can run them in this way:

```
pdk test unit
```

## Supported Modules for MariaDB

At the time of writing, there are no supported or approved modules for MariaDB.

However there is a [mysql module](#) supported by Puppet, that supports the Puppet Development Kit. Though it doesn't support MariaDB-specific features, it works with MariaDB. Its documentation shows how to use the module to install MariaDB on certain operating systems.

Several unsupported and not approved modules exist for MariaDB and MaxScale.

## Resources and References

- [Puppet Forge](#) website.
- [Puppet Development Kit](#) documentation.
- [Modules overview](#) in Puppet documentation.
- [Beginner's guide to writing modules](#) in Puppet documentation.
- [Puppet Supported Modules](#) page in Puppet Forge.

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.5 Vagrant and MariaDB

Vagrant is an open source tool to quickly setup machines that can be used for development and testing. These machines can be local virtual machines, Docker containers, AWS EC2 instances, and so on. Vagrant allows one to easily and quickly setup test MariaDB servers.



### Vagrant Overview for MariaDB Users

*Vagrant architecture, general concepts and basic usage.*



### Creating a Vagrantfile

*How to create a new Vagrant box running MariaDB.*



### Vagrant Security Concerns

*Security matters related to Vagrant machines.*



### 2.1.2.14.5.1 Vagrant Overview for MariaDB Users

Vagrant is a tool to create and manage development machines (Vagrant boxes). They are usually virtual machines on the localhost system, but they could also be Docker containers or remote machines. Vagrant is open source software maintained by HashiCorp and released under the MIT license.

Vagrant benefits include simplicity, and a system to create test boxes that is mostly independent from the technology used.

For information about installing Vagrant, see [Installation](#) in Vagrant documentation.

In this page we discuss basic Vagrant concepts.

#### Contents

- 1. [Vagrant Concepts](#)
  - 1. [Example](#)
  - 2. [Vagrantfiles](#)
  - 3. [Providers](#)
  - 4. [Provisioners](#)
  - 5. [Plugins](#)
  - 6. [Changes in Vagrant 3.0](#)
- 2. [Vagrant Commands](#)
- 3. [Vagrant Resources and References](#)

## Vagrant Concepts

A **Vagrant machine** is compiled from a box. It can be a virtual machine, a container or a remote server from a cloud service.

A **box** is a package that can be used to create Vagrant machines. We can download boxes from [app.vagrantup.com](#), or we can build a new box from a Vagrantfile. A box can be used as a base for another box. The base boxes are usually operating system boxes downloaded from [app.vagrantup.com](#).

A **provider** is responsible for providing the virtualization technology that will run our machine.

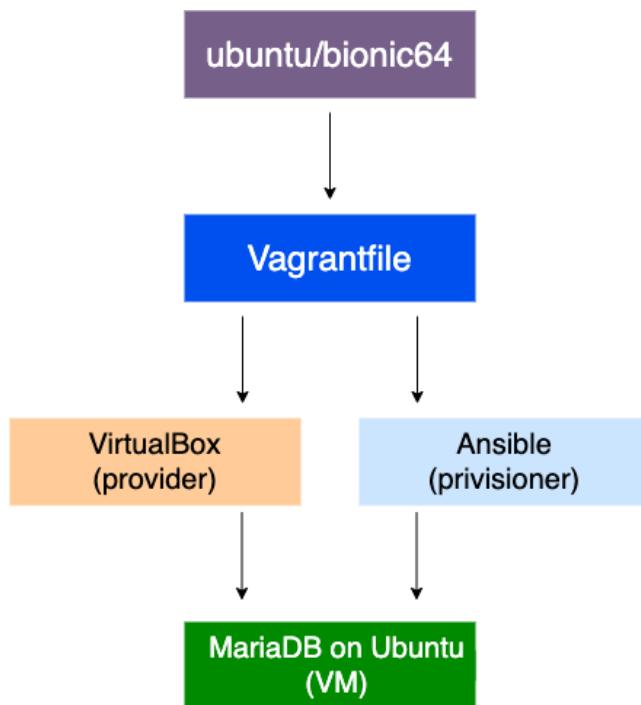
A **provisioner** is responsible for installing and configuring the necessary software on a newly created Vagrant machine.

## Example

The above concepts are probably easier to understand with an example.

We can use an Ubuntu box as a base to build a Vagrant machine with MariaDB. So we write a Vagrantfile for this purpose. In the Vagrantfile we specify VirtualBox as a provider. And we use the Ansible provisioner to install and configure MariaDB. Once we finish this Vagrantfile, we can run a Vagrant command to start a Vagrant machine, which is actually a VirtualBox VM running MariaDB on Ubuntu.

The following diagram should make the example clear:



## Vagrantfiles

A Vagrantfile is a file that describes how to create one or more Vagrant machines. Vagrantfiles use the Ruby language, as well as objects provided by

Vagrant itself.

A Vagrantfile is often based on a box, which is usually an operating system in which we are going to install our software. For example, one can create a MariaDB Vagrantfile based on the

```
ubuntu/trusty64
```

box. A Vagrantfile can describe a box with a single server, like MariaDB, but it can also contain a whole environment, like LAMP. For most practical use cases, having the whole environment in a single box is more convenient.

Boxes can be searched in [Vagrant Cloud](#). Most of their Vagrantfiles are available on GitHub. Searches can be made, among other things, by keyword to find a specific technology, and by provider.

## Providers

A provider adds support for creating a specific type of machines. Vagrant comes with several providers, for example:

- VirtualBox allows one to create virtual machines with VirtualBox.
- Hyper-V allows one to create virtual machines with Microsoft Hyper-V.
- Docker allows one to create Docker containers.

Alternative providers are maintained by third parties or sold by HashiCorp. They allow one to create different types of machines, for example using VMWare.

Some examples of useful providers, recognized by the community:

- [Vagrant AWS Provider](#).
- [Vagrant Google Compute Engine \(GCE\) Provider](#).
- [Vagrant Azure Provider](#).
- [OpenVZ](#).
- [vagrant-lxc](#).

If you need to create machines with different technologies, or deploy them to unsupported cloud platforms, you can develop a custom provider in Ruby language. To find out how, see [Plugin Development: Providers](#) in Vagrant documentation. The [Vagrant AWS Provider](#) was initially written as an example provider.

## Provisioners

A provisioner is a technology used to deploy software to the newly created machines.

The simplest provisioner is

```
shell  
, which runs a shell file inside the Vagrant machine.  
powershell  
is also available.
```

Other providers use automation software to provision the machine. There are provisioners that allow one to use [Ansible](#), [Puppet](#), Chef or Salt. Where relevant, there are different provisioners allowing the use of these technologies in a distributed way (for example, using Puppet apply) or in a centralized way (for example, using a Puppet server).

It is interesting to note that there is both a Docker provider and a Docker provisioner. This means that a Vagrant machine can be a Docker container, thanks to the

```
docker  
provisioner. Or it could be any virtualisation technology with Docker running in it, thanks to the  
docker  
provisioner. In this case, Docker pulls images and starts containers to run the software that should be running in the Vagrant machine.
```

If you need to use an unsupported provisioning method, you can develop a custom provisioner in Ruby language. See [Plugin Development: Provisioners](#) in Vagrant documentation.

## Plugins

It is possible to install a plugin with this command:

```
vagrant plugin install <plugin_name>
```

A Vagrantfile can require that a plugin is installed in this way:

```
require 'plugin_name'
```

A plugin can be a Vagrant plugin or a Ruby gem installable from [rubygems.org](#). It is possible to install a plugin that only exists locally by specifying its path.

## Changes in Vagrant 3.0

HashiCorp published an article that describes its [plans for Vagrant 3.0](#).

Vagrant will switch to a client-server architecture. Most of the logic will be stored in the server, while the development machines will run a thin client that communicates with the server. It will be possible to store the configuration in a central database.

Another notable change is that Vagrant is switching from Ruby to Go. For some time, it will still be possible to use Vagrantfiles and plugins written in Ruby. However, in the future Vagrantfiles and plugins should be written in one of the languages that support [gRPC](#) (not necessarily Go). Vagrantfiles can also be written in [HCL](#), HashiCorp Configuration Language.

## Vagrant Commands

This is a list of the most common Vagrant commands. For a complete list, see [Command-Line Interface](#) in Vagrant documentation.

To list the available machines:

```
vagrant box list
```

To start a machine from a box:

```
cd /box/directory  
vagrant up
```

To connect to a machine:

```
vagrant ssh
```

To see all machines status and their id:

```
vagrant global-status
```

To destroy a machine:

```
vagrant destroy <id>
```

## Vagrant Resources and References

Here are some valuable websites and pages for Vagrant users.

- [Vagrant Up](#) .
- [app.vagrantup.com](#) .
- [Vagrant Community](#) .
- [Vagrant on Wikipedia](#) .
- [Vagrant on HashiCorp Learn](#) .

---

Content initially contributed by [Vettabase Ltd](#) .

### 2.1.2.14.5.2 Creating a Vagrantfile

In this page we discuss how to create a Vagrantfile, which you can use to create new boxes or machines. This content is specifically written to address the needs of MariaDB users.

#### Contents

1. [A Basic Vagrantfile](#)
2. [Providers](#)
3. [Provisioners](#)
  1. [The shell Provisioner](#)
  2. [Uploading Files](#)
  3. [Provisioning Vagrant with Ansible](#)
  4. [Provisioning Vagrant with Puppet](#)
4. [Sharing Files Between the Host and a Guest System](#)
5. [Network Communications](#)
  1. [Private Networks](#)
  2. [Public Networks](#)
  3. [Exposing Ports](#)
  4. [Use Cases](#)
6. [References](#)

## A Basic Vagrantfile

A Vagrantfile is a Ruby file that instructs Vagrant to create, depending on how it is executed, new Vagrant machines or boxes. You can see a box as a compiled Vagrantfile. It describes a type of Vagrant machines. From a box, we can create new Vagrant machines. However, while a box is easy to distribute to a team or to a wider public, a Vagrantfile can also directly create one or more Vagrant machines, without generating any box.

Here is a simple Vagrantfile example:

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/bionic64"
  config.vm.provider "virtualbox"
  config.vm.provision :shell, path: "bootstrap.sh"
end
```

Vagrant.configure("2")  
returns the Vagrant configuration object for the new box. In the block, we'll use the  
config  
alias to refer this object. We are going to use version 2 of Vagrant API.

vm.box

is the base box that we are going to use. It is Ubuntu BionicBeaver (18.04 LTS), 64-bit version, provided by HashiCorp. The schema for box names is simple: the maintainer account in [Vagrant Cloud](#) followed by the box name.

We use

```
vm.provision  
to specify the name of the file that is going to be executed at the machine creation, to provision the machine.  
bootstrap.sh  
is the conventional name used in most cases.
```

To create new Vagrant machines from the Vagrantfile, move to the directory that contains the Vagrant project and run:

```
vagrant up
```

To compile the Vagrantfile into a box:

```
vagrant package
```

These operations can take time. To check if the Vagrantfile contains syntax errors or certain types of bugs:

```
vagrant validate
```

## Providers

A provider allows Vagrant to create a Vagrant machine using a certain technology. Different providers may enable a virtual machine manager ([VirtualBox](#), [VMWare](#), [Hyper-V](#) ...), a container manager ([Docker](#)), or remote cloud hosts ([AWS](#), [Google Compute Engine](#) ...).

Some providers are developed by third parties. [app.vagrant.com](#) supports search for boxes that support the most important third parties providers. To find out how to develop a new provider, see [Plugin Development: Providers](#).

Provider options can be specified. Options affect the type of Vagrant machine that is created, like the number of virtual CPUs. Different providers support different options.

It is possible to specify multiple providers. In this case, Vagrant will try to use them in the order they appear in the Vagrantfile. It will try the first provider; if it is not available it will try the second; and so on.

Here is an example of providers usage:

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/bionic64"
  config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "--memory", 1024 * 4]
  end
  config.vm.provider "vmware_fusion"
end
```

In this example, we try to use VirtualBox to create a virtual machine. We specify that this machine must have 4G of RAM (1024M \* 4). If VirtualBox is not available, Vagrant will try to use VMWare.

This mechanism is useful for at least a couple of reasons:

- Different users may use different systems, and maybe they don't have the same virtualization technologies installed.
- We can gradually move from one provider to another. For a period of time, some users will have the new virtualization technology installed, and they will use it; other users will only have the old technology installed, but they will still be able to create machines with Vagrant.

# Provisioners

We can use different methods for provisioning. The simplest provisioner is

```
shell
```

, that allows one to run a Bash file to provision a machine. Other provisioners allow setting up the machines using automation software, including Ansible, Puppet, Chef and Salt.

To find out how to develop a new provisioner, see [Plugin Development: Provisioners](#).

## The

### shell Provisioner

In the example above, the `shell` provisioner runs `bootstrap.sh` inside the Vagrant machine to provision it. A simple `bootstrap.sh` may look like the following:

```
#!/bin/bash

apt-get update
apt-get install -y
```

To find out the steps to install MariaDB on your system of choice, see the [Getting, Installing, and Upgrading MariaDB](#) section.

You may also want to restore a database backup in the new Vagrant machine. In this way, you can have the database needed by the application you are developing. To find out how to do it, see [Backup and Restore Overview](#). The most flexible type of backup (meaning that it works between different MariaDB versions, and in some cases even between MariaDB and different DBMSs) is a `dump`.

On Linux machines, the

```
shell
```

provisioner uses the default shell. On Windows machines, it uses PowerShell.

## Uploading Files

If we use the

```
shell
```

provisioner, we need a way to upload files to the new machine when it is created. We could use the `file`

provisioner, but it works by connecting the machine via ssh, and the default user doesn't have permissions for any directory except for the synced folders. We could change the target directory owner, or we could add the default user to a group with the necessary privileges, but these are not considered good practices.

Instead, we can just put the file we need to upload somewhere in the synced folder, and then copy it with a shell command:

```
cp ./files/my.cnf /etc/mysql/conf.d/
```

## Provisioning Vagrant with Ansible

Here is an example of how to provision a Vagrant machine or box by running Ansible:

```
Vagrant.configure("2") do |config|
  ...
  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "vagrant.yml"
  end
end
```

With the [Ansible provisioner](#), Ansible runs in the host system. In this example, it runs a playbook called

```
vagrant.yml
```

. The [Ansible Local provisioner](#) runs the playbook in the vagrant machine.

For an introduction to Ansible for MariaDB users, see [Ansible and MariaDB](#).

## Provisioning Vagrant with Puppet

To provision a Vagrant machine or box by running Puppet:

```
Vagrant.configure("2") do |config|
  ...
  config.vm.provision "puppet" do |puppet|
    puppet.manifests_path = "manifests"
    puppet.manifest_file = "default.pp"
  end
end
```

In this example, Puppet Apply runs in the host system and no Puppet Server is needed. Puppet expects to find a

```
manifests
directory in the project directory. It expects it to contain
default.pp
, which will be used as an entry point. Note that
puppet.manifests_path
and
puppet.manifest_file
are set to their default values.
```

Puppet needs to be installed in the guest machine.

To use a Puppet server, the

```
puppet_server
provisioner can be used:
```

```
Vagrant.configure("2") do |config|
  ...
  config.vm.provision "puppet_server" do |puppet|
    puppet.puppet_server = "puppet.example.com"
  end
end
```

See the [Puppet Apply provisioner](#) and the [Puppet Agent Provisioner](#).

For an introduction to Puppet for MariaDB users, see [Puppet and MariaDB](#).

## Sharing Files Between the Host and a Guest System

To restore a backup into MariaDB, in most cases we need to be able to copy it from the host system to the box. We may also want to occasionally copy MariaDB logs from the box to the host system, to be able to investigate problems.

The project directory (the one that contains the Vagrantfile) by default is shared with the virtual machine and mapped to the

```
/vagrant
directory (the synced folder). It is a good practice to put there all files that should be shared with the box when it is started. Those files should normally be versioned.
```

The synced folder can be changed. In the above example, we could simply add one line:

```
config.vm.synced_folder "/host/path", "/guest/path"
```

The synced folder can also be disabled:

```
config.vm.synced_folder '.', '/vagrant', disabled: true
```

Note that multiple Vagrant machines may have synced folders that point to the same directory on the host system. This can be useful in some cases, if you prefer to test some functionalities quickly, rather than replicating production environment as faithfully as possible. For example, to test if you're able to take a backup from one machine and restore it to another, you can store the backup in a common directory.

## Network Communications

It is often desirable for a machine to be able to communicate with "the outside". This can be done in several ways:

- Private networks;
- Public networks;
- Exposing ports to the host.

Remember that Vagrant doesn't create machines, but it asks a provisioner to create machines. Some provisioners support all of these communication methods, others may support some of them, or even none of them. When you create a Vagrantfile that starts machines using one of these features, it is implicit that this can only happen if the provisioner you are using supports the features you need. Check your provisioner documentation to find out which features it supports.

The default provisioner, VirtualBox, supports all these communication methods, including multiple networks.

## Private Networks

A private network is a networks that can only be accessed by machines that run on the same host. Usually this also means that the machines must run on the same provisioner (for example, they all must be VirtualBox virtual machines).

Some provisioners support multiple private networks. This means that every network has a different name and can be accessed by different machines.

The following line shows how to create or join a private network called "example", where this machine's IP is assigned by the provisioner via DHCP:

```
config.vm.network 'private_network', name: 'example', type: 'dhcp'
```

While this is very convenient to avoid IP conflicts, sometimes you prefer to assign some IP's manually, in this way:

```
config.vm.network 'private_network', name: 'example', ip: '111.222.111.222'
```

## Public Networks

As explained above, public networks are networks that can be accessed by machines that don't run on the same host with the same provider.

To let a machine join a public network:

```
# use provisioner DHCP:  
config.vm.network "public_network", use_dhcp_assigned_default_route: true  
  
# assign ip manually:  
config.vm.network "public_network", ip: "111.222.111.222"
```

To improve security, you may want to configure a gateway:

```
config.vm.provision "shell", run: "always", inline: "route add default gw 111.222.111.222"
```

## Exposing Ports

Vagrant allows us to map a TCP or UDP port in a guest system to a TCP or UDP port in the host system. For example, you can map a virtual machine port 3306 to the host port 12345. Then you can connect MariaDB in this way:

```
mariadb -hlocalhost -P12345 -u<user> -p<password>
```

You are not required to map a port to a port with a different number. In the above example, if the port 3306 in your host is not in use, you are free to map the guest port 3306 to the host port 3306.

There are a couple of caveats:

- You can't map a single host port to multiple guest ports. If you want to expose the port 3306 from multiple Vagrant machines, you'll have to map them to different host ports. When running many machines this can be hard to maintain.
- Ports with numbers below 1024 are privileged ports. Mapping privileged ports requires root privileges.

To expose a port:

```
config.vm.network 'forwarded_port', guest: 3306, host: 3306
```

## Use Cases

Suppose you run MariaDB and an application server in two separate Vagrant machines. It's usually best to let them communicate via a private network, because this greatly increases your security. The application server will still need to expose ports to the host, so the application can be tested with a web browser.

Suppose you have multiple environments of the same type, like the one described above. They run different applications that don't communicate with each other. In this case, if your provisioner supports this, you will run multiple private networks. You will need to expose the applications servers ports, mapping them to different host ports.

You may even want to implement different private networks to create an environment that reflects production complexity. Maybe in production you have a cluster of three MariaDB servers, and the application servers communicate with them via a proxy layer (ProxySQL, HAProxy, or MaxScale). So the applications can communicate with the proxies, but have no way to reach MariaDB directly. So there is a private network called "database" that can be accessed by the MariaDB servers and the proxy servers, and another private network called "application" that can be accessed by the proxy servers and the application servers. This requires that your provisioner supports multiple private networks.

Using public networks instead of private one will allow VMs that run on different hosts to be part of your topology. In general this is considered as an insecure practice, so you should probably ask yourself if you really need to do this.

## References

The [vagrant-mariadb-examples](#) repository is an example of a Vagrantfile that creates a box containing MariaDB and some useful tools for developers.

Further information can be found in Vagrant documentation.

- [Vagrantfile](#) .
- [Providers](#) .
- [Synced Folders](#) .
- [Ansible Provisioner](#) .
- [Puppet Apply Provisioner](#) .
- [Puppet Agent Provisioner](#) .

See also [Ruby documentation](#) .

---

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.5.3 Vagrant Security Concerns

Databases typically contain information to which access should be restricted. For this reason, it's worth discussing some security concerns that Vagrant users should be aware of.

### Contents

1. [Access to the Vagrant Machine](#)
2. [Synced Folders](#)
3. [Reporting Security Bugs](#)

## Access to the Vagrant Machine

By default, Vagrant machines are only accessible from the localhost. SSH access uses randomly generated key pairs, and therefore it is secure.

The password for

root  
and  
vagrant  
is "vagrant" by default. Consider changing it.

## Synced Folders

By default, the project folder in the host system is shared with the machine, which sees it as

/vagrant

This means that whoever has access to the project folder also has read and write access to the synced folder. If this is a problem, make sure to properly restrict the access to the synced folder.

If we need to exchange files between the host system and the Vagrant machine, it is not advisable to disable the synced folder. This is because the only alternative is to use the

file

provider, which works by copying files to the machine via ssh. The problem is that the default ssh user does not have permissions to write to any directory by default, and changing this would be less secure than using a synced folder.

When a machine is provisioned, it should read the needed files from the synced folder or copy them to other places. Files in the synced folder should not be accessed by the Vagrant machine during its normal activities. For example, it is fine to load a dump from the synced folder during provisioning; and it is fine to copy configuration files from the synced folder to directories in

/etc

during provisioning. But it is a bad practice to let MariaDB use table files located in the synced folder.

## Reporting Security Bugs

Note that security bugs are not reported as normal bugs. Information about security bugs are not public. See [Security at HashiCorp](#) for details.

---

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.5.4 Running MariaDB ColumnStore Docker containers on Linux, Windows and MacOS

### Contents

1. [Introduction](#)
2. [Windows Linux Subsystem](#)
3. [Docker](#)

# Introduction

Docker allows for a simple and lightweight setup of a MariaDB ColumnStore single server instance for evaluation purposes. The configuration is designed for simplified developer / evaluation setup rather than production use. It allows to evaluate ColumnStore on a Windows or MacOS system, setting up a Linux system in a container. The Docker image uses a base OS of CentOS and currently require separate download of the CentOS RPM install bundle.

## Windows Linux Subsystem

If you have Windows 10 Creators update installed, then you can install the Ubuntu installation into the Bash console. Please follow the Ubuntu instructions in getting started. If you have recently upgraded and had Bash installed previously, ensure you uninstall and reinstall Bash first to have a clean Ubuntu installation. Note that ColumnStore will be terminated should you terminate the Bash console.

## Docker

[Docker](#) manages lightweight containers that allows for creation of lightweight and reproducible containers with a dedicated function. On Windows and MacOS systems, Docker transparently runs on a Linux virtual machine.

Since MariaDB ColumnStore relies on a Syslog daemon, the container must start both ColumnStore and rsyslogd and the runit utility is used to achieve this.

A single node docker image can be found at [MariaDB on docker hub](#).

```
docker run -d --name mcs mariadb/columnstore  
docker exec -it mcs bash
```

A ColumnStore cluster can be brought up using a compose file provided in the ColumnStore github repository:

```
git clone https://github.com/mariadb-corporation/mariadb-columnstore-docker.git  
cd mariadb-columnstore-docker/columnstore  
docker-compose up -d
```

For more information about how to manage Docker containers, see [Installing and Using MariaDB via Docker](#).

To test an application that uses ColumnStore, it is desirable to setup several containers that will communicate with each other. To do this, we can use Docker Compose. See [Setting Up a LAMP Stack with Docker Compose](#) for more information.

### 2.1.2.14.6 Docker and MariaDB

Docker is an open source container manager that can be used to quickly create ephemeral containers running specific software. Docker containers can be used for production, development or testing, though it is not the preferred choice to run MariaDB in production.



#### Benefits of Managing Docker Containers with Orchestration Software

[Benefits of managing Docker with Automation Software.](#)



#### Installing and Using MariaDB via Docker

[Creating and managing a MariaDB Docker container.](#)



#### Running MariaDB ColumnStore Docker containers on Linux, Windows and MacOS

[Docker allows for a simple setup of a ColumnStore single server instance for evaluation purposes](#)



#### Creating a Custom Docker Image

[How to write a Dockerfile to create custom Docker images.](#)



#### Setting Up a LAMP Stack with Docker Compose

[How to use Docker Compose to set up containers running a LAMP stack.](#)



#### Docker Security Concerns

[Security matters related to Docker containers.](#)



#### MariaDB Container Cheat Sheet

[Get the images Images can be found on MariaDB Docker Hub. To get the list o...](#)

There are [6 related questions](#).

### 2.1.2.14.6.1 Benefits of Managing Docker Containers with Orchestration Software

In this page we'll discuss why automating [Docker](#) containers with software like [Ansible](#) or [Puppet](#) may be desirable in some cases. To talk about this, we'll first need to discuss why Docker containers are defined *ephemeral*, and how this applies to containerized database servers (particularly MariaDB).

During the discussion, we should keep in mind that Docker can be used to setup production and/or development environments. These use cases are very different from a database perspective: a production database may be big, and typically contains data that we don't want to lose. Development environments usually contain small sample data that can be rebuilt relatively quickly. This page focuses on the latter case.

## Docker's Ephemeral Nature

Docker images are compiled from Dockerfiles. Containers are created from images. Normally, a container is not modified from the moment it is created. In other words, containers are usually designed to be **ephemeral**, meaning that they can be destroyed and replaced with new containers at any time. Provided that there is proper redundancy (for example, there are several web servers running the same services) destroying one container and starting a new one of the same type won't cause any damage.

We will discuss a bit later how this applies to MariaDB, and more generally to database servers.

When something should change, for example some software version or configuration, normally Dockerfiles are updated and containers are recreated from the latest image versions. For this reason, containers shouldn't contain anything that shouldn't be lost, and recreating them should be an extremely cheap operation. **Docker Compose** or the **Swarm mode** are used to declare which containers form a certain environment, and how they communicate with each other.

On the contrary, Ansible and Puppet are mainly built to manage the configuration of existing servers. It doesn't recreate servers, it changes their configuration. So Docker and Ansible have very different approaches. For this reason, Ansible and Puppet are not frequently used to deploy containers to production. However, using them together can bring some benefits, especially for development environments.

More on this later in the page. First, we need to understand how these concepts apply to database servers.

## Stateful Technologies

Using ephemeral containers works very well for *stateless* technologies, like web servers and proxies. These technologies virtually only need binaries, configuration and small amounts of data (web pages). If some data need to be restored after a container creation, it will be a fast operation.

In the case of a database, the problem is that data can be large and need to be written somewhere. We don't want all databases to disappear when we destroy a container. Even if we had an up-to-date backup, restoring it would take time.

However, Docker has features called **volumes** and **volume containers**. We won't discuss the difference here, let's focus on their purpose. A volume is a directory in the host system mapped to a directory in one or more containers. Volumes are not destroyed when containers are destroyed. They can be used to share data between any number of containers and the host system. Therefore, they are also a good way to persist data.

Suppose a MariaDB container called

```
mariadb-main-01  
uses a volume that is mapped to  
/var/docker/volumes/mariadb-main
```

. At some point we want to use a more recent MariaDB version. As explained earlier, the Docker way to do this is to destroy the container and create a new one that uses a more recent version of the MariaDB image.

So, we will destroy

```
mariadb-main-01  
. The volume is still there. Then we create a new container with the same name, but based on a newer image. We make sure to link the volume  
to the new container too, so it will be able to use  
/var/docker/volumes/mariadb-main  
again. At this point we may want to run mysql\_upgrade, but apart from that, everything should just work.
```

The above described steps are simple, but running them manually is time consuming and error-prone. Automating them with some automation software like Ansible or Puppet is often desirable.

## Ways to Deploy Docker Containers

Docker containers can be deployed in the following ways:

- Manually. See [Installing and Using MariaDB via Docker](#). This is not recommended for production, or for complex environments. However, it can easily be done for the simplest cases. If we want to make changes to our [custom images](#), we'll need to modify the Dockerfiles, destroy the containers and recreate them.
- With Docker Compose. See [Setting Up a LAMP Stack with Docker Compose](#) for a simple example. When we modify a Dockerfile, we'll need to destroy the containers and recreate them, which is usually as simple as running

```
docker-compose down  
followed by  
docker-compose-up  
. After changing  
docker-compose.yml  
(maybe to add a container or a network) we'll simply need to run  
docker-compose-up  
again, because it is idempotent.
```

- Using Ansible, Puppet or other automation software, as mentioned before. We can use Ansible or Puppet to create the containers, and run them again every time we want to apply some change to the containers. This means that the containers are potentially created once and modified any

number of times.

In all these cases, it is entirely possible to add [Vagrant](#) to the picture. Vagrant is a way to deploy or provision several hosts, including virtual machines (the most common case), and containers. It is agnostic in regarding the underlying technology, so it can deploy to a virtual machine, a container, or even a remote server in the same way. Docker can work with Vagrant in two ways:

- As a [provisioner](#). In this case Vagrant will most commonly deploy a virtual machine, and will use Docker to setup the applications that need to run in it, as containers. This guarantees a higher level of isolation, compared to running the containers in the local host. Especially if you have different environments to deploy locally, because you can have them on different virtual machines.
- As a [provider](#). Vagrant will deploy one or more Docker containers locally. Once each container is up, Vagrant can optionally use a provisioner on it, to make sure that the container runs the proper software with proper configuration. In this case, Ansible, Puppet or other automation software can be used as a provisioner. But again, this is optional: it is possible to make changes to the Dockerfiles and recreate the containers every time.

## Benefits of Managing Docker Containers with Automation Software

Docker containers can be entirely managed with Docker Compose or the Swarm mode. This is often a good idea.

However, choosing to use automation software like Ansible or Puppet has some benefits too. Benefits include:

- Docker containers allow working without modifying the host system, and their creation is very fast. Much faster than virtual machines. This makes Docker desirable for development environments.
- As explained, making all containers ephemeral and using volumes to store important data is possible. But this means adding some complexity to adapt an ephemeral philosophy to technologies that are not ephemeral by nature (databases). Also, many database professionals don't like this approach. Using automation software allows easily triggering upgrades and configuration changes in the containers, treating them as non-ephemeral systems.
- Sometimes Docker is only used in development environments. If production databases are managed via Ansible, Puppet, or other automation software, this could lead to some code duplication. Dealing with configuration changes using the same procedures will reduce the cost of maintenance.
- While recreating containers is fast, being able to apply small changes with Ansible or Puppet can be more convenient in some cases: particularly if we write files into the container itself, or if recreating a container bootstrap involves some lengthy procedure.
- Trying to do something non-standard with Dockerfiles can be tricky. For example, running two processes in a container is possible but can be problematic, as Docker is designed to run a process per container. However there are situations when this is desirable. For example PMM containers run several different processes. Launching additional processes with Ansible or Puppet may be easier than doing it with a Dockerfile.

With all this in mind, let's see some examples of cases when managing Docker containers with Ansible, Puppet or other automation software is preferable, rather than destroying containers every time we want to make a change:

- We use Ansible or Puppet in production, and we try to keep development environments as similar as possible to production. By using Ansible/Puppet in development too, we can reuse part of the code.
- We make changes to the containers often, and recreating containers is not as fast as it should be (for example because a MariaDB [dump](#) needs to be restored).
- Creating a container implies some complex logic that does not easily fit a Dockerfile or Docker Compose (including, but not limited to, running multiple processes per container).

That said, every case is different. There are environments where these advantages do not apply, or bring a very small benefit. In those cases, the cost of adding some automation with Ansible, Puppet or similar software is probably not justified.

## How to Deploy to Container from Orchestration Software

Suppose you want to manage containers configuration with Ansible.

**At a first glance**, the simplest way is to run Ansible in the host system. It will need to connect to the containers via SSH, so they need to expose the 22 port. But we have multiple containers, so we'll need to map the 22 port of each container to a different port in the host. This is hard to maintain and potentially insecure: in production you want to avoid exposing any container port to the host.

A better solution is to run Ansible itself in a container. The playbooks will be in a Docker volume, so we can access them from the host system to manage them more easily. The Ansible container will communicate with other containers using a Docker network, using the standard 22 port (or another port of your choice) for all containers.

## See Also

See these pages on how to manage Docker containers with different automation technologies:

- [Deploying Docker Containers with Ansible](#).
- [Deploying Docker Containers with Puppet](#).

---

Content initially contributed by [Vettabase Ltd](#).

## 2.1.2.14.6.2 Installing and Using MariaDB via Docker

## Contents

- 1. [Installing Docker on Your System with the Universal Installation Script](#)
  - 1. Starting dockerd
- 2. [Using MariaDB Images](#)
  - 1. Downloading an Image
  - 2. Creating a Container
  - 3. Running and Stopping the Container
    - 1. Automatic Restart
    - 2. Pausing Containers
  - 4. Troubleshooting a Container
  - 5. Accessing the Container
- 6. [Connecting to MariaDB from Outside the Container](#)
  - 1. Forcing a TCP Connection
  - 2. Port Configuration for Clustered Containers and Replication
- 3. [Installing MariaDB on Another Image](#)
  - 1. Daemonizing the Operating System
  - 2. Installing MariaDB
- 4. [See Also](#)

Sometimes we want to install a specific version of MariaDB, [MariaDB ColumnStore](#), or [MaxScale](#) on a certain system, but no packages are available. Or maybe, we simply want to isolate MariaDB from the rest of the system, to be sure that we won't cause any damage.

A virtual machine would certainly serve the scope. However, this means installing a system on the top of another system. It requires a lot of resources.

In many cases, the best solution is using containers. Docker is a framework that runs containers. A container is meant to run a specific daemon, and the software that is needed for that daemon to properly work. Docker does not virtualize a whole system; a container only includes the packages that are not included in the underlying system.

Docker requires a very small amount of resources. It can run on a virtualized system. It is used both in development and in production environments. Docker is an open source project, released under the Apache License, version 2.

Note that, while your package repositories could have a package called

```
docker  
, it is probably not the Docker we are talking about. The Docker package could be called  
docker.io  
or  
docker-engine
```

For information about installing Docker, see [Get Docker](#) in Docker documentation.

## Installing Docker on Your System with the Universal Installation Script

The script below will install the Docker repositories, required kernel modules and packages on the most common Linux distributions:

```
curl -sSL https://get.docker.com/ | sh
```

## Starting dockerd

On some systems you may have to start the

```
dockerd daemon  
yourself:
```

```
sudo systemctl start docker  
sudo gpasswd -a "${USER}" docker
```

If you don't have

```
dockerd  
running, you will get the following error for most  
docker  
commands:
```

```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
```

## Using MariaDB Images

The easiest way to use MariaDB on Docker is choosing a MariaDB image and creating a container.

## Downloading an Image

You can download a MariaDB image for Docker from the [Official Docker MariaDB](#), or choose another image that better suits your needs. You can search Docker Hub (the official set of repositories) for an image with this command:

```
docker search mariadb
```

Once you have found an image that you want to use, you can download it via Docker. Some layers including necessary dependencies will be downloaded too. Note that, once a layer is downloaded for a certain image, Docker will not need to download it again for another image.

For example, if you want to install the default MariaDB image, you can type:

```
docker pull mariadb:10.4
```

This will install the 10.4 version. Versions 10.2, 10.3, 10.5 are also valid choices.

You will see a list of necessary layers. For each layer, Docker will say if it is already present, or its download progress.

To get a list of installed images:

```
docker images
```

## Creating a Container

An image is not a running process; it is just the software needed to be launched. To run it, we must create a container first. The command needed to create a container can usually be found in the image documentation. For example, to create a container for the official MariaDB image:

```
docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -p 3306:3306 -d docker.io/library/mariadb:10.3
```

mariadbtest

is the name we want to assign the container. If we don't specify a name, an id will be automatically generated.

10.2 and 10.5 are also valid target versions:

```
docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -p 3306:3306 -d docker.io/library/mariadb:10.2
```

```
docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -p 3306:3306 -d docker.io/library/mariadb:10.5
```

Optionally, after the image name, we can specify some [options for mysqld](#). For example:

```
docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=mypass -p 3306:3306 -d mariadb:10.3 --log-bin --binlog-format=MIXED
```

Docker will respond with the container's id. But, just to be sure that the container has been created and is running, we can get a list of running containers in this way:

```
docker ps
```

We should get an output similar to this one:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
819b786a8b48	mariadb	/docker-entrypoint.	4 minutes ago	Up 4 minutes	3306/tcp

## Running and Stopping the Container

Docker allows us to restart a container with a single command:

```
docker restart mariadbtest
```

The container can also be stopped like this:

```
docker stop mariadbtest
```

The container will not be destroyed by this command. The data will still live inside the container, even if MariaDB is not running. To restart the container and see our data, we can issue:

```
docker start mariadbtest
```

With

```
docker stop  
, the container will be gracefully terminated: a  
SIGTERM  
signal will be sent to the  
mysqld  
process, and Docker will wait for the process to shutdown before returning the control to the shell. However, it is also possible to set a timeout,  
after which the process will be immediately killed with a  
SIGKILL  
. Or it is possible to immediately kill the process, with no timeout.
```

```
docker stop --time=30 mariadbtest  
docker kill mariadbtest
```

In case we want to destroy a container, perhaps because the image does not suit our needs, we can stop it and then run:

```
docker rm mariadbtest
```

Note that the command above does not destroy the data volume that Docker has created for /var/lib/mysql. If you want to destroy the volume as well, use:

```
docker rm -v mariadbtest
```

## Automatic Restart

When we start a container, we can use the

```
--restart  
option to set an automatic restart policy. This is useful in production.
```

Allowed values are:

- no  
: No automatic restart.
- on-failure  
: The container restarts if it exits with a non-zero exit code.
- unless-stopped  
: Always restart the container, unless it was explicitly stopped as shown above.
- always  
: Similar to  
unless-stopped  
, but when Docker itself restarts, even containers that were explicitly stopped will restart.

It is possible to change the restart policy of existing, possibly running containers:

```
docker update --restart always mariadb  
# or, to change the restart policy of all containers:  
docker update --restart always $(docker ps -q)
```

A use case for changing the restart policy of existing containers is performing maintenance in production. For example, before upgrading the Docker version, we may want to change all containers restart policy to

```
always  
, so they will restart as soon as the new version is up and running. However, if some containers are stopped and not needed at the moment, we  
can change their restart policy to  
unless-stopped
```

## Pausing Containers

A container can also be frozen with the

```
pause  
command. Docker will freeze the process using cgroups. MariaDB will not know that it is being frozen and, when we  
unpause  
it, MariaDB will resume its work as expected.
```

Both

```
pause  
and  
unpause
```

accept one or more container names. So, if we are running a cluster, we can freeze and resume all nodes simultaneously:

```
docker pause node1 node2 node3
docker unpause node1 node2 node3
```

Pausing a container is very useful when we need to temporarily free our system's resources. If the container is not crucial at this moment (for example, it is performing some batch work), we can free it to allow other programs to run faster.

## Troubleshooting a Container

If the container doesn't start, or is not working properly, we can investigate with the following command:

```
docker logs mariadbtest
```

This command shows what the daemon sent to the stdout since the last attempt of starting - the text that we typically see when we invoke

```
mysqld
from the command line.
```

On some systems, commands such as

```
docker stop mariadbtest
and
docker restart mariadbtest
may fail with a permissions error. This can be caused by AppArmor, and even
sudo
```

won't allow you to execute the command. In this case, you will need to find out which profile is causing the problem and correct it, or disable it.

**Disabling AppArmor altogether is not recommended, especially in production.**

To check which operations were prevented by AppArmor, see [AppArmor Failures](#) in AppArmor documentation.

To disable a profile, create a symlink with the profile name (in this example,

```
mysqld
) to
/etc/apparmor.d/disable
, and then reload profiles:
```

```
ln -s /etc/apparmor.d/usr.sbin.mysql /etc/apparmor.d/disable/
sudo apparmor_parser -R /etc/apparmor.d/usr.sbin.mysql
```

For more information, see [Policy Layout](#) in AppArmor documentation.

After disabling the profile, you may need to run:

```
sudo service docker restart
docker system prune --all --volumes
```

Restarting the system will then allow Docker to operate normally.

## Accessing the Container

To access the container via Bash, we can run this command:

```
docker exec -it mariadbtest bash
```

Now we can use normal Linux commands like `cd`, `ls`, etc. We will have root privileges. We can even install our favorite file editor, for example:

```
apt-get update
apt-get install vim
```

In some images, no repository is configured by default, so we may need to add them.

Note that if we run [mysqladmin shutdown](#) or the [SHUTDOWN](#) command to stop the container, the container will be deactivated, and we will automatically exit to our system.

## Connecting to MariaDB from Outside the Container

If we try to connect to the MariaDB server on

```
localhost
, the client will bypass networking and attempt to connect to the server using a socket file in the local filesystem. However, this doesn't work
when MariaDB is running inside a container because the server's filesystem is isolated from the host. The client can't access the socket file which is inside
the container, so it fails to connect.
```

Therefore connections to the MariaDB server must be made using TCP, even when the client is running on the same machine as the server container.

Find the IP address that has been assigned to the container:

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mariadbtest
```

You can now connect to the MariaDB server using a TCP connection to that IP address.

## Forcing a TCP Connection

After enabling network connections in MariaDB as described above, we will be able to connect to the server from outside the container.

On the host, run the client and set the server address ("h") to the container's IP address that you found in the previous step:

```
mysql -h 172.17.0.2 -u root -p
```

This simple form of the connection should work in most situations. Depending on your configuration, it may also be necessary to specify the port for the server or to force TCP mode:

```
mysql -h 172.17.0.2 -P 3306 --protocol=TCP -u root -p
```

## Port Configuration for Clustered Containers and Replication

Multiple MariaDB servers running in separate Docker containers can connect to each other using TCP. This is useful for forming a Galera cluster or for replication.

When running a cluster or a replication setup via Docker, we will want the containers to use different ports. The fastest way to achieve this is mapping the containers ports to different port on our system. We can do this when creating the containers (

```
docker run  
command), by using the  
-p  
option, several times if necessary. For example, for Galera nodes we will use a mapping similar to this one:
```

```
-p 4306:3306 -p 5567:5567 -p 5444:5444 -p 5568:5568
```

## Installing MariaDB on Another Image

It is possible to download a Linux distribution image, and to install MariaDB on it. This is not much harder than installing MariaDB on a regular operating system (which is easy), but it is still the hardest option. Normally we will try existing images first. However, it is possible that no image is available for the exact version we want, or we want a custom installation, or perhaps we want to use a distribution for which no images are available. In these cases, we will install MariaDB in an operating system image.

## Daemonizing the Operating System

First, we need the system image to run as a daemon. If we skip this step, MariaDB and all databases will be lost when the container stops.

To demonize an image, we need to give it a command that never ends. In the following example, we will create a Debian Jessie daemon that constantly pings the 8.8.8.8 special address:

```
docker run --name debian -p 3306:3306 -d debian /bin/sh -c "while true; do ping 8.8.8.8; done"
```

## Installing MariaDB

At this point, we can enter the shell and issue commands. First we will need to update the repositories, or no packages will be available. We can also update the packages, in case some of them are newer than the image. Then, we will need to install a text editor; we will need it to edit configuration files. For example:

```
# start an interactive Bash session in the container  
docker exec -ti debian bash  
apt-get -y update  
apt-get -y upgrade  
apt-get -y install vim
```

Now we are ready to [install MariaDB](#) in the way we prefer.

## See Also

- [Official MariaDB Docker Images Webinar](#) .
- [Docker official site](#) .

- [Docker Hub](#) .
- [Docker documentation](#) .

## 2.1.2.14.6.3 Running MariaDB ColumnStore Docker containers on Linux, Windows and MacOS

### Contents

1. [Introduction](#)
2. [Windows Linux Subsystem](#)
3. [Docker](#)

## Introduction

Docker allows for a simple and lightweight setup of a MariaDB ColumnStore single server instance for evaluation purposes. The configuration is designed for simplified developer / evaluation setup rather than production use. It allows to evaluate ColumnStore on a Windows or MacOS system, setting up a Linux system in a container. The Docker image uses a base OS of CentOS and currently require separate download of the CentOS RPM install bundle.

## Windows Linux Subsystem

If you have Windows 10 Creators update installed, then you can install the Ubuntu installation into the Bash console. Please follow the Ubuntu instructions in getting started. If you have recently upgraded and had Bash installed previously, ensure you uninstall and reinstall Bash first to have a clean Ubuntu installation. Note that ColumnStore will be terminated should you terminate the Bash console.

## Docker

[Docker](#) manages lightweight containers that allows for creation of lightweight and reproducible containers with a dedicated function. On Windows and MacOS systems, Docker transparently runs on a Linux virtual machine.

Since MariaDB ColumnStore relies on a Syslog daemon, the container must start both ColumnStore and rsyslogd and the runit utility is used to achieve this.

A single node docker image can be found at [MariaDB on docker hub](#) .

```
docker run -d --name mcs mariadb/columnstore
docker exec -it mcs bash
```

A ColumnStore cluster can be brought up using a compose file provided in the ColumnStore github repository:

```
git clone https://github.com/mariadb-corporation/mariadb-columnstore-docker.git
cd mariadb-columnstore-docker/columnstore
docker-compose up -d
```

For more information about how to manage Docker containers, see [Installing and Using MariaDB via Docker](#) .

To test an application that uses ColumnStore, it is desirable to setup several containers that will communicate with each other. To do this, we can use Docker Compose. See [Setting Up a LAMP Stack with Docker Compose](#) for more information.

## 2.1.2.14.6.4 Creating a Custom Docker Image

OCI containers, frequently called Docker containers, are created from OCI images. An image contains software that can be launched, including the underlying system. A container is an instance of that software.

When we want to automate MariaDB, creating an image with MariaDB and the desired configuration, we may want to create an image by ourselves, which fulfills our needs.

### Contents

1. [Images Architecture](#)
2. [Dockerfile Syntax](#)
  1. [Using Variables](#)
3. [Versioning and Deploying Images](#)
  1. [Container registries](#)
  2. [Choosing Image Names and Tags](#)
  3. [Pushing and Pulling Images](#)
  4. [Docker Content Trust](#)
4. [Good Practices and Caveats](#)
5. [References](#)

## Images Architecture

One "source code" of an image is a Dockerfile. A Dockerfile is written in Docker specific language, and can be compiled into an image by the

```
docker  
binary, using the  
docker build  
command. It can also be compiled by
```

```
buildah
```

```
using  
buildah bud
```

Most images are based on another image. The base image is specified at the beginning of the Dockerfile, with the

```
FROM
```

directive. If the base image is not present in the local system, it is downloaded from the repository specified, or if not specified, from the default repository of the build program. This is often Docker Hub. For example, we can build a

```
mariadb-rocksdb:10.5  
image starting from the  
debian:13
```

image. In this way, we'll have all the software included in a standard Debian image, and we'll add MariaDB and its configuration upon that image.

All the following Dockerfile directives are compiled into a new Docker image, identified by an SHA256 string. Each of these images is based on the image compiled from the previous directive. A physical compiled image can serve as a base for any number of images. This mechanism saves a lot of disk space, download time and build time.

The following diagram shows the relationship between Dockerfiles, images and containers:



## Dockerfile Syntax

Here's a simple Dockerfile example:

```
FROM ubuntu:20.04  
  
RUN apt-get update  
RUN apt-get install -y mariadb-server  
  
EXPOSE 3306  
  
LABEL version="1.0"  
LABEL description="MariaDB Server"  
  
HEALTHCHECK --start-period=5m \  
CMD mariadb -e 'SELECT @@datadir;' || exit 1  
  
CMD ["mysqld"]
```

This example is not very good for practical purposes, but it shows what a Dockerfile looks like.

First, we declare that the base image to use is

```
ubuntu:20.04
```

Then we run some commands to install MariaDB from the Ubuntu default repositories and stop the MariaDB service.

We define some metadata about the image with

```
LABEL  
. Any label is valid.
```

We declare that the port 3306 (MariaDB default port) should be exposed. However, this has no effect if the port is not exposed at container creation.

We also define a healthcheck. This is a command that is run to check if the container is healthy. If the return code is 0 the healthcheck succeeds, if it's 1 it fails. In the MariaDB specific case, we want to check that it's running and able to answer a simple query. This is better than just checking that MariaDB process is running, because MariaDB could be running but unable to respond, for example because `max_connections` was reached or data was corrupted.

We read a system variable, because we should not assume that any user-created table exists. We also specify

```
--start-period
```

to allow some time for MariaDB to start, keeping in mind that restarting it may take some time if some data is corrupted. Note that there can be only one healthcheck: if the command is specified multiple times, only the last occurrence will take effect.

Finally, we start the container command: `mysqld`. This command is run when a container based on this image starts. When the process stops or crashes, the container will immediately stop.

Note that, in a container, we normally run `mysqld` directly, rather than running `mysqld_safe` or running MariaDB as a service. Containers restart can be handled by the container service. See [automatic restart](#).

See the documentation links below to learn the syntax allowed in a Dockerfile.

## Using Variables

It is possible to use variables in a Dockerfile. This allows us, for example, to install different packages, install different versions of a package, or configure software differently depending on how variables are set, without modifying the Dockerfile itself.

To use a variable, we can do something like this:

```
FROM ubuntu:20.04

ARG MARIADB_CONFIG_FILE

...
RUN mysqld --defaults-file=$MARIADB_CONFIG_FILE
```

Here

```
ARG
```

is used after the

```
FROM
```

directive, thus the variable cannot be used in

```
FROM
```

. It is also possible to declare a variable before

```
FROM
```

, so we can use a variable to select the base image to use or its tag, but in this case the variable cannot be used after the

```
FROM
```

directive. Here is an example:

```
ARG UBUNTU_VERSION
FROM ubuntu:$UBUNTU_VERSION

# But this will cause a build error:
RUN echo 'Ubuntu version: $UBUNTU_VERSION' > /var/build_log
```

We'll have to assign variables a value when we build the Dockerfile, in this way:

```
docker build --build-arg UBUNTU_VERSION=20.04 .
```

Note that Dockerfile variables are just placeholders for values. Dockerfiles do not support assignment, conditionals or loops.

## Versioning and Deploying Images

Dockerfiles are normally versioned, as well as the files that are copied to the images.

Once an image is built, it can be pushed to a container registry. Whenever an image is needed on a host to start containers from it, it is pulled from the registry.

## Container registries

A default container registry for OCI images is Docker Hub. It contains images created by both the Docker Library team and the community. Any individual or organization can open an account and push images to Docker Hub. Most Docker images are open source: the Dockerfiles and the needed files to build the images are usually on GitHub.

It is also possible to setup a self-hosted registry. Images can be pushed to that registry and pulled from it, instead of using Docker Hub. If the registry is not publicly accessible, it can be used to store images used by the organization without making them publicly available.

But a self-hosted registry can also be useful for open source images: if an image is available on Docker Hub and also on a self-hosted registry, in case Docker Hub is down or not reachable, it will still be possible to pull images.

## Choosing Image Names and Tags

The names of images developed by the community follow this schema:

```
repository/maintainer/technology
```

It doesn't matter if the maintainer is an individual or an organization. For images available on Docker Hub, the maintainer is the name of a Docker Hub account.

Official images maintained by the Docker Library maintainers don't contain the name of the maintainer. For example, the official MariaDB image is called `mariadb` which is an alias for `docker.io/library/mariadb`.

All images have a tag, which identifies the version or the variant of an image. For example, all MariaDB versions available on Docker are used as image tags. [MariaDB 10.5](#) is called

```
mariadb:10.5
```

By convention, tags form a hierarchy. So for example, there is a

```
10.1.1  
tag whose meaning will not change over time.  
10.5  
will always identify the latest version in the 10.5 branch. For some time it was  
10.5.1  
, then it became  
10.5.2  
, and so on.
```

When we pull an image without specifying a tag (ie,

```
docker pull mariadb  
)  
, we are implicitly requiring the image with the  
latest  
tag. This is even more mutable: at different periods of time, it pointed to the latest  
10.0  
version, to the latest  
10.1  
version, and so on.
```

In production, it is always better to know for sure which version we are installing. Therefore it is better to specify a tag whose meaning won't change over time, like

```
10.5.1
```

## Pushing and Pulling Images

To pull an image from Docker Hub or a self-hosted registry, we use the

```
docker pull  
command. For example:
```

```
docker pull mariadb:10.5
```

This command downloads the specified image if it is not already present in the system, or if the local version is not up to date.

After modifying a Dockerfile, we can build an image in this way:

```
docker build .
```

This step can be automated by services like Docker Hub and GitHub. Check those service's documentation to find out how this feature works.

Once an image is created, it can be pushed to a registry. We can do it in this way:

```
docker push <image_name>:<tag>
```

## Docker Content Trust

Docker has a feature called Docker Content Trust (DCT). It is a system used to digitally sign images, based on PEM keys. For environments where security is a major concern, it is important to sign images before pushing them. This can be done with both Docker Hub and self-hosted registries.

## Good Practices and Caveats

As mentioned, a Dockerfile is built by creating a new image for each directive that follows

- FROM  
 . This leads to some considerations.
- Sometimes it can be a good idea to run several shell commands in a single  
`RUN`  
 directive to avoid creating images that are not useful.
  - Modifying a directive means that all subsequent directives also need to be rebuilt. When possible, directives that are expected to change often should follow directives that will change seldom.
  - Directives like  
`LABEL`  
 or  
`EXPOSE`  
 should be placed close to the end of Dockerfiles. In this way they will be rebuilt often, but this operation is cheap. On the other side, changing a label should not trigger a long rebuild process.
  - Variables should be used to avoid Dockerfiles proliferation. But if a variable is used, changing its value should be tested. So, be sure not to use variables without a good reason.
  - Writing logic into a Dockerfile is impossible or very hard. Call shell scripts instead, and write your logic into them. For example, in a shell script it is easy to perform a certain operation only if a variable is set to a certain value.
  - If you need MariaDB containers with different configurations or different sets of plugins, use the method explained above. Do not create several Dockerfiles, with different tags, for each desired configuration or plugin set. This may lead to undesired code duplication and increased maintenance costs.

## References

More details can be found in the Docker documentation:

- [Dockerfile reference](#) .
- [docker build](#) .
- [Repositories](#) .
- [Deploy a registry server](#) .
- [Content trust in Docker](#) .

See also:

- [Privacy-Enhanced Mail](#) on Wikipedia.

---

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.6.5 Setting Up a LAMP Stack with Docker Compose

Docker Compose is a tool that allows one to declare which Docker containers should run, and which relationships should exist between them. It follows the **infrastructure as code** approach, just like most automation software and Docker itself.

For information about installing Docker Compose, see [Install Docker Compose](#) in Docker documentation.

### Contents

1. [The docker-compose.yml File](#)
  1. [About Volumes](#)
  2. [Using Variables](#)
2. [Docker Compose Commands](#)
3. [Docker Compose Resources and References](#)

## The

### docker-compose.yml File

When using Docker Compose, the Docker infrastructure must be described in a YAML file called  
`docker-compose.yml`

Let's see an example:

```

version: "3"

services:
  web:
    image: "apache:${PHP_VERSION}"
    restart: 'always'
    depends_on:
      - mariadb
    restart: 'always'
    ports:
      - '8080:80'
    links:
      - mariadb
  mariadb:
    image: "mariadb:${MARIADB_VERSION}"
    restart: 'always'
    volumes:
      - "/var/lib/mysql/data:${MARIADB_DATA_DIR}"
      - "/var/lib/mysql/logs:${MARIADB_LOG_DIR}"
      - /var/docker/mariadb/conf:/etc/mysql
    environment:
      MYSQL_ROOT_PASSWORD: "${MYSQL_ROOT_PASSWORD}"
      MYSQL_DATABASE: "${MYSQL_DATABASE}"
      MYSQL_USER: "${MYSQL_USER}"
      MYSQL_PASSWORD: "${MYSQL_PASSWORD}"

```

In the first line we declare that we are using version 3 of the Docker compose language.

Then we have the list of services, namely the

- `web`
- and the
- `mariadb`
- services.

Let's see the properties of the services:

- `port`  
maps the 8080 container port to the 80 host system port. This is very useful for a development environment, but not in production, because it allows us to connect our browser to the containerized web server. Normally there is no need to connect to MariaDB from the host system.
- `links`  
declares that this container must be able to connect  
`mariadb`  
. The hostname is the container name.
- `depends_on`  
declares that  
`mariadb`  
needs to start before  
`web`  
. This is because we cannot do anything with our application until MariaDB is ready to accept connections.
- `restart: always`  
declares that the containers must restart if they crash.
- `volumes`  
creates volumes for the container if it is set in a service definition, or a volume that can be used by any container if it is set globally, at the same level as  
`services`  
. Volumes are directories in the host system that can be accessed by any number of containers. This allows destroying a container without losing data.
- `environment`  
sets environment variables inside the container. This is important because in setting these variables we set the MariaDB root credentials for the container.

## About Volumes

It is good practice to create volumes for:

- The [data directory](#), so we don't lose data when a container is created or replaced, perhaps to upgrade MariaDB.
- The directory where we put all the logs, if it is not the datadir.

- The directory containing all configuration files (for development environments), so we can edit those files with the editor installed in the host system. Normally no editor is installed in containers. In production we don't need to do this, because we can copy files from a repository located in the host system to the containers.

Note that Docker Compose variables are just placeholders for values. Compose does not support assignment, conditionals or loops.

## Using Variables

In the above example you can see several variables, like

```
 ${MARIADB_VERSION}
 . Before executing the file, Docker Compose will replace this syntax with the
 MARIADB_VERSION
 variable.
```

Variables allow making Docker Compose files more re-usable: in this case, we can use any MariaDB image version without modifying the Docker Compose file.

The most common way to pass variables is to write them into a file. This has the benefit of allowing us to version the variable file along with the Docker Compose file. It uses the same syntax you would use in BASH:

```
PHP_VERSION=8.0
MARIADB_VERSION=10.5
...
```

For bigger setups, it could make sense to use different environment files for different services. To do so, we need to specify the file to use in the Compose file:

```
services:
  web:
    env_file:
      - web-variables.env
...
```

## Docker Compose Commands

Docker Compose is operated using

```
docker-compose
. Here we'll see the most common commands. For more commands and for more information about the commands mentioned here, see the documentation.
```

Docker Compose assumes that the Composer file is located in the current directory and it's called

```
docker-compose.yml
. To use a different file, the
-f <filename>
parameter must be specified.
```

To pull the necessary images:

```
docker-compose pull
```

Containers described in the Compose file can be created in several ways.

To create them only if they do not exist:

```
docker-compose up --no-recreate
```

To create them if they do not exist and recreate them if their image or configuration have changed:

```
docker-compose up
```

To recreate containers in all cases:

```
docker-compose up --force-recreate
```

Normally

```
docker-compose up
starts the containers. To create them without starting them, add the
--no-start
option.
```

To restart containers without recreating them:

```
docker-compose restart
```

To kill a container by sending it a

```
SIGKILL
```

```
:
```

```
docker-compose kill <service>
```

To instantly remove a running container:

```
docker-compose rm -f <service>
```

To tear down all containers created by the current Compose file:

```
docker-compose down
```

## Docker Compose Resources and References

- [Overview of Docker Compose](#) in the Docker documentation.
- [Compose file](#) in the Docker documentation.
- [Docker Compose](#) on GitHub.

Further information about the concepts explained in this page can be found in Docker documentation:

- [Environment variables in Compose](#).
- [Overview of docker-compose CLI](#).
- [Compose command-line reference](#).

Content initially contributed by [Vettabase Ltd](#).

### 2.1.2.14.6.6 Docker Security Concerns

When using Docker containers in production, it is important to be aware of Docker security concerns.

#### Contents

1. [Host System Security](#)
2. [Images Security](#)
3. [References](#)

## Host System Security

All Docker containers are built upon the host system's kernel. If the host system's kernel has security bugs, those bugs are also present in the containers.

In particular, Docker leverages two Linux features:

- Namespaces, to isolate containers from each other and make sure that a container can't establish unauthorized connections to another container.
- cgroups, to limit the resources (CPU, memory, IO) that each container can consume.

The administrators of a system running Docker should be particularly careful to upgrade the kernel whenever security bugs to these features are fixed.

Docker, like most container technologies, uses the runC open source library. runC security bugs are likely to affect Docker.

Finally, Docker's own security bugs potentially affect all containers.

It is important to note that when we upgrade the kernel, runC or Docker itself we cause downtime for all the containers running on the system.

## Images Security

Docker containers are built from images. If security is a major concern, you should make sure that the images you use are secure.

If you want to be sure that you are pulling authentic images, you should only pull images signed with [Docker Content Trust](#).

The images should create and use a system user with less privileges than root. For example, the MariaDB daemon usually runs as a user called

```
mysql  
, who belongs the  
mysql
```

group. There is no need to run it as root (and by default it will refuse to start as root). Containers should not run it as root. Using an unprivileged user will reduce the chances that a bug in Docker or in the kernel will allow the user to gain access to the host system.

Updated images should be used. An image usually downloads packages information at build time. If the image is not recently built, a newly created container will have old packages. Updating the packages on container creation and regularly re-updating them will ensure that the container uses packages with the most recent versions. Rebuilding an image often will reduce the time necessary to update the packages the first time.

Security bugs are usually important for a database server, so you don't want your version of MariaDB to contain known security bugs. But suppose you also have a bug in Docker, in runC, or in the kernel. A bug in a user-facing application may allow an attacker to exploit a bug in those lower level technologies. So, after gaining access to the container, an attacker may gain access to the host system. This is why system administrators should keep both the host system and the software running in the containers updated.

## References

For more information, see the following links:

- [Docker security](#) on Docker documentation.
- [Linux namespaces](#) on Wikipedia.
- [cgroups](#) on Wikipedia.
- [runC repository](#).

Content initially contributed by [Vettabase Ltd](#).

## 2.1.2.14.6.7 MariaDB Container Cheat Sheet

- Get the images  
Images can be found on [MariaDB Docker Hub](#).  
To get the list of images run

```
$ docker images ls
```

- Create the network

```
$ docker network create mynetwork
```

It is good practice to create the Docker network and attach the container to the network.

- Start the container with server options  
To start the container in the background with the MariaDB server image run:

```
$ docker run --rm --detach \
--env MARIADB_ROOT_PASSWORD=sosecret \
--network mynetwork \
--name mariadb-server \
mariadb:latest
```

Additionally [environment variables](#) are also provided.

- Get the list of running containers (specify the flag `-a` in case you want to see all containers)

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ad374ec8a272	mariadb:latest	"docker-entrypoint.s..."	3 seconds ago	Up 1 second	3306/tcp	mariadb-server

- Start the client from the container

To start the `mariadb` client inside the created container and run specific commands, run the following:

```
$ docker exec -it mariadb-server mariadb -psosecret -e "SHOW PLUGINS"
```

- Inspect logs from the container

```
$ docker logs mariadb-server
```

In the logs you can find status information about the server, plugins, generated passwords, errors and so on.

- Restart the container

```
$ docker restart mariadb-server
```

- Run commands within the container

```
$ docker exec -it mariadb-server bash
```

- Use a volume to specify configuration options

```
$ docker run --detach --env MARIADB_USER=anel \
--env MARIADB_PASSWORD=anel \
--env MARIADB_DATABASE=my_db \
--env MARIADB_RANDOM_ROOT_PASSWORD=1 \
--volume $PWD/my_container_config:/etc/mysql/conf.d:z \
--network mynetwork \
--name mariadb-server1 \
mariadb:latest
```

One can specify custom [configuration files](#) through the `/etc/mysql/conf.d` volume during container startup.

- Use a volume to specify grants during container start

```
$ docker run --detach --env MARIADB_USER=anel \
--env MARIADB_PASSWORD=anel \
--env MARIADB_DATABASE=my_db \
--env MARIADB_RANDOM_ROOT_PASSWORD=1 \
--volume $PWD/my_init_db:/docker-entrypoint-initdb.d \
--network mynetwork \
--name mariadb-server1 \
mariadb:latest
```

User created with the environment variables has full grants only to the `MARIADB_DATABASE`. In order to override those grants, one can specify grants to a user, or execute any SQL statements from host file to `docker-entrypoint-initdb.d`. In the `local_init_dir` directory we can find the file, created like this:

```
$ echo "GRANT ALL PRIVILEGES ON *.* TO anel;" > my_init_db/my_grants.sql
```

## See Also

- [Installing and using MariaDB via Docker](#)

### 2.1.2.14.7 Kubernetes and MariaDB

General information and hints on how to deploy MariaDB Kubernetes containers.

Kubernetes is an open source containers orchestration system. It automates deployments, horizontal scaling, configuration and operations. It is often referred to as K8s.



#### Kubernetes Overview for MariaDB Users

[An overview of Kubernetes and how it works with MariaDB.](#)



#### Kubernetes Operators for MariaDB

[An overview of Kubernetes operators that can be used with MariaDB](#)

### 2.1.2.14.8 Kubernetes Overview for MariaDB Users

Kubernetes, or K8s, is software to orchestrate containers. It is released under the terms of an open source license, Apache License 2.0.

Kubernetes was originally developed by Google. Currently it is maintained by the Cloud Native Computing Foundation (CNCF), with the status of Graduated Project.

For information about how to setup a learning environment or a production environment, see [Getting started](#) in Kubernetes documentation.

## Contents

1. <a href="#">Architecture</a>
1. <a href="#">Nodes</a>
1. <a href="#">Kubelet</a>
2. <a href="#">kube-proxy</a>
3. <a href="#">Container Runtime</a>
2. <a href="#">Controllers</a>
3. <a href="#">Control Plane</a>
1. <a href="#">API Server</a>
2. <a href="#">kube-controller-manager</a>
3. <a href="#">etcd</a>
4. <a href="#">kube-scheduler</a>
5. <a href="#">cloud-controller-manager</a>
4. <a href="#">Clients and Tools</a>
1. <a href="#">kubectl</a>
2. <a href="#">kubeadm</a>
3. <a href="#">kind and minikube</a>
2. <a href="#">Ansible Resources and References</a>

# Architecture

Kubernetes runs in a **cluster**. A cluster runs a **workload**: a set of servers that are meant to work together (web servers, database servers, etc).

A Kubernetes cluster consists of the following components:

- **Nodes** run containers with the servers needed by our applications.
- **Controllers** constantly check the cluster nodes current state, and compare it with the desired state.
- A **Control Plane** is a set of different components that store the cluster desired state and take decisions about the nodes. The Control Plane provides an API that is used by the controllers.

For more information on Kubernetes architecture, see [Concepts](#) and [Kubernetes Components](#) in Kubernetes documentation.

## Nodes

A node is a system that is responsible to run one or more pods. A pod is a set of containers that run a Kubernetes workload or part of it. All containers that run in the same pod are also located on the same node. Usually identical pods run on different nodes for fault tolerance.

For more details, see [Nodes](#) in the Kubernetes documentation.

Every node must necessarily have the following components:

- **kubelet**
- **kube-proxy**
- A **container runtime**

### kubelet

kubelet has a set of **PodSpecs** which describe the desired state of pods. It checks that the current state of the pods matches the desired state. It especially takes care that containers don't crash.

### kube-proxy

In a typical Kubernetes cluster, several containers located in different pods need to connect to other containers, located in the same pods (for performance and fault tolerance reasons). Therefore, when we develop and deploy an application, we can't know in advance the IPs of the containers to which it will have to connect. For example, an application server may need to connect to MariaDB, but the MariaDB IP will be different for every pod.

The main purpose of kube-proxy is to implement the concept of Kubernetes **services**. When an application needs to connect to MariaDB, it will connect to the MariaDB service. kube-proxy will receive the request and will redirect it to a running MariaDB container in the same pod.

## Container Runtime

Kubernetes manages the containers in a pod via a container runtime, or container manager, that supports the Kubernetes Container Runtime Interface (CRI). Container runtimes that meet this requisite are listed in the [Container runtimes](#) page in the Kubernetes documentation. More information about the Container Runtime Interface can be found [on GitHub](#).

Originally, Kubernetes used Docker as a container runtime. This was later deprecated, but Docker images can still be used using any container runtime.

## Controllers

Controllers constantly check if there are differences between the pod's current state and their desired state. When differences are found, controllers try to fix them. Each node type controls one or more resource types. Several types of controllers are needed to run a cluster.

Most of the actions taken by the controllers user the API server in the Control Plane. However, this is not necessarily true for custom controllers. Also, some actions cannot be performed via the Control Plane. For example, if some nodes crashed, adding new nodes involves taking actions outside of the Kubernetes cluster, and controllers will have to do this themselves.

It is possible to write custom controllers to perform checks that require knowledge about a specific technology. For example, a MariaDB custom controller may want to check if [replication](#) is working by issuing [SHOW REPLICAS STATUS](#) commands. This logic is specific to the way MariaDB works, and can only be implemented in a customer controller. Custom controllers are usually part of operators.

For more information, see [Controllers](#) in the Kubernetes documentation.

## Control Plane

The control plane consists of the following components.

For more information about the control plane, see [Control Plane Components](#) in Kubernetes documentation.

## API Server

An API Server exposes API functions both internally and externally. It is essential to coordinate Kubernetes components so that they react to node's change of state, and it allows the user to send commands.

The default implementation of the API Server is kube-apiserver. It is able to scale horizontally and to balance the load between its instances.

## kube-controller-manager

Most controllers run in this component.

## etcd

etcd contains all data used by a Kubernetes cluster. It is a good idea to take regular backups of etcd data.

## kube-scheduler

When a new pod is created, kube-scheduler decides which node should host it. The decision is made based on several criteria, like the resource requirements for the pod.

## cloud-controller-manager

cloud-controller-manager implements the logic and API of a cloud provider. It receives requests from the API Server and performs specific actions, like creating an instance in AWS. It also runs controllers that are specific to a cloud vendor.

# Clients and Tools

Kubernetes comes with a set of tools that allow us to communicate with the API server and test a cluster.

## kubectl

kubectl allows communication with the API server and run commands on a Kubernetes cluster.

## kubeadm

kubeadm allows creating a Kubernetes cluster that is ready to receive commands from kubectl.

## kind and minikube

These tools are meant to create and manage test clusters on a personal machine. They work on Linux, MacOS and Windows. kind creates a cluster that consists of Docker containers, therefore it requires Docker to be installed. minikube runs a single-node cluster on the local machine.

# Ansible Resources and References

- [Kubernetes website](#) .
- [Kubernetes](#) on Wikipedia.
- [Kubernetes organization](#) on GitHub.
- [OperatorHub.io](#)
- [Kubernetes Community Forums](#) .
- (video) [MariaDB database clusters on Kubernetes](#) , by Pengfei Ma, at MariaDB Server Fest 2020.

---

Content initially contributed by [Vettabase Ltd](#) .

## 2.1.2.14.9 Kubernetes Operators for MariaDB

Operators basically instruct Kubernetes about how to manage a certain technology. Kubernetes comes with some default operators, but it is possible to create custom operators. Operators created by the community can be found on [OperatorHub.io](#) .

### Contents

1. [Custom Operators](#)
2. [MariaDB Operator](#)
3. [Other Operators](#)

## Custom Operators

Kubernetes provides a declarative API. To support a specific (i.e. MariaDB) technology or implement a desired behavior (i.e. provisioning a [replica](#) ), we extend Kubernetes API. This involves creating two main components:

- A custom resource.
- A custom controller.

A custom resource adds an API endpoint, so the resource can be managed via the API server. It includes functionality to get information about the resource, like a list of the existing servers.

A custom controller implements the checks that must be performed against the resource to check if its state should be corrected using the API. In the case of MariaDB, some reasonable checks would be verifying that it accepts connections, replication is running, and a server is (or is not) read only.

# MariaDB Operator

[OperatorHub.io](#) has a [MariaDB operator](#). At the time of this writing it is in alpha stage, so please check its maturity before using it.

MariaDB operator is open source and is released under the terms of the MIT license. The source code is [available on GitHub](#). The `README.md` file shows usage examples.

It defines the following custom resources:

- MariaDB server.
- MariaDB Backup. [mariabackup](#) is used.
- MariaDB Monitor. Prometheus metrics are exposed on port 9090.

## Other Operators

If you know about other MariaDB operators, feel free to add them to this page (see [Writing and Editing Knowledge Base Articles](#)).

MySQL and Percona Server operators should work as well, though some changes may be necessary to fix [incompatibilities](#) or take advantage of certain [MariaDB features](#).

---

Content initially contributed by [Vettabase Ltd](#).

## 2.1.2.14.10 Automating Upgrades with MariaDB.Org Downloads REST API

The MariaDB Foundation maintains a Downloads REST API. See the [Downloads API documentation](#) to find out all the tasks that you can accomplish with this API. Generally speaking, we can say that it provides information about MariaDB products and available versions. This allows to easily automate upgrades for MariaDB and related products.

The Downloads API exposes HTTPS endpoints that return information in JSON format. HTTP and JSON are extremely common standards that can be easily used with any programming language. All the information provided by the API is public, so no authentication is required.

### How to Use the API with a Unix Shell

Linux shells are great for writing simple scripts. They are compatible to each other to some extent, so simple scripts can be run on almost any Unix/Linux system. In the following examples we'll use Bash.

On Linux, some programs you'll generally need to work with any REST API are:

- `curl`, to call HTTP URLs and get their output.
- `js`, to extract or transform information from a JSON document.

### Example: Check When a New Version Becomes GA

A trivial use case is to write a script that checks the list of MariaDB GA major versions and, when something changes, send us an email. So we can test the newest GA version and eventually install it.

The script in this example will be extremely simple. We'll do it this way:

- Retrieve the JSON object describing all MariaDB versions.
- For each element of the array, only show the
  - `release_id`
  - and
  - `release_status`
  - properties, and concatenate them.
- Apply a filter, so we only select the rows containing 'stable' but not 'old' (so we exclude 'Old Stable').
- From the remaining rows, only show the first column (the version number).
- If the results we obtained are different from the previously written file (see last point) send an email.
- Save the results into a file.

This is something that we can easily do with a Unix shell:

```

#!/bin/bash

current_ga_versions=$(
    curl https://downloads.mariadb.org/rest-api/mariadb/ | \
    jq -r '.major_releases[] | .release_id + " " + .release_status' | \
    grep -i 'stable' | grep -vi 'old' | \
    cut -d ' ' -f 1
)

# create file if it doesn't exist, then compare version lists
touch ga_versions
previous_ga_versions=$( cat ga_versions )

echo "$current_ga_versions" > ga_versions

if [ "$current_ga_versions" != "$previous_ga_versions" ];
then
    mail -s 'NOTE: New MariaDB GA Versions' devops@example.com <<< 'There seems to be a new MariaDB GA version! Yay!'
fi

```

The only non-standard command here is jq. It is a great way to manipulate JSON documents, so if you don't know it you may want to take a look at [jq documentation](#).

## How to use the API with a Python script

To use the API with Python, we need a module that is able to send HTTP requests and parse a JSON output. The

requests  
module has both these features. It can be installed as follows:

```
pip install requests
```

The following script prints stable versions to the standard output:

```

#!/usr/bin/env python

import requests

response = requests.get('https://downloads.mariadb.org/rest-api/mariadb/').json()

for x in response['major_releases']:
    if x['release_status'] == 'Stable':
        print(x['release_id'])

requests.get()
makes an HTTP call of type GET, and
requests.json()
returns a dictionary representing the previously obtained JSON document.

```

Content initially contributed by [Vettabase Ltd](#).

## 2.1.2.14.11 HashiCorp Vault and MariaDB

Vault is open source software for secret management provided by HashiCorp. It is designed to avoid sharing secrets of various types, like passwords and private keys. When building automation, Vault is a good solution to avoid storing secrets in plain text in a repository.

MariaDB and Vault may relate each other in several ways:

- MariaDB secrets can be stored in Vault. Typically this includes user's passwords and private keys for SSH access.
- MariaDB (and MySQL) can be used as a secret engine, a component which stores, generates, or encrypts data.
- MariaDB (and MySQL) can be used as a backend storage, providing durability for Vault data.

For information about how to install Vault, see [Install Vault](#).

### Contents

1. [Vault Features](#)
2. [Vault Architecture](#)
3. [Dev Mode](#)
4. [Vault Resources and References](#)

## Vault Features

Vault is used via an HTTP/HTTPS API.

Vault is identity-based. Users login and Vault sends them a token that is valid for a certain amount of time, or until certain conditions occur. Users with a valid token may request to obtain secrets for which they have proper permissions.

Vault encrypts the secrets it stores.

Vault can optionally audit changes to secrets and secrets requests by the users.

## Vault Architecture

Vault is a server. This allows decoupling the secrets management logic from the clients, which only need to login and keep a token until it expires.

The sever can actually be a cluster of servers, to implement high availability.

The main Vault components are:

- **Storage Backend** : This is where the secrets are stored. Vault only send encrypted data to the backend storage.
- **HTTP API** : This API is used by the clients, and provides an access to Vault server.
- **Barrier** : Similarly to an actual barrier, it protects all inner Vault components. The HTTP API and the storage backend are outside of the barrier and could be accessed by anyone. All communications from and to these components have to pass through the barrier. The barrier verifies data and encrypts it. The barrier can have two states: *sealed* or *unsealed*. Data can only pass through when the barrier is unsealed. All the following components are located inside the barrier.
- **Auth Method** : Handles login attempts from clients. When a login succeeds, the auth method returns a list of security policies to Vault core.
- **Token Store** : Here the tokens generated as a result of a succeeded login are stored.
- **Secrets Engines** : These components manage secrets. They can have different levels of complexity. Some of them simply expect to receive a key, and return the corresponding secret. Others may generate secrets, including one-time-passwords.
- **Audit Devices** : These components log the requests received by Vault and the responses sent back to the clients. There may be multiple devices, in which case an **Audit Broker** sends the request or response to the proper device.

## Dev Mode

It is possible to start Vault in dev mode:

```
vault server -dev
```

Dev mode is useful for learning Vault, or running experiments on some particular features. It is extremely insecure, because dev mode is equivalent to starting Vault with several insecure options. This means that Vault should never run in production in dev mode. However, this also means that all the regular Vault features are available in dev mode.

Dev mode simplifies all operations. Actually, no configuration is necessary to get Vault up and running in dev mode. It makes it possible to communicate with the Vault API from the shell without any authentication. Data is stored in memory by default. Vault is unsealed by default, and if explicitly sealed, it can be unsealed using only one key.

For more details, see "["Dev" Server Mode](#)" in Vault documentation.

## Vault Resources and References

- [Documentation](#).
- [MySQL/MariaDB Database Secrets Engine](#).
- [MySQL Storage Backend](#).

---

Content initially contributed by [Vettabase Ltd](#).

## 2.1.2.14.12 Orchestrator Overview

Orchestrator is a MySQL and MariaDB high availability and replication management tool. It is released by Shlomi Noach under the terms of the Apache License, version 2.0.

Orchestrator provides automation for MariaDB replication in the following ways:

- It can be used to perform certain operations, like repairing broken replication or moving a replica from one master to another. These operations can be requested using CLI commands, or via the GUI provided with Orchestrator. The actual commands sent to MariaDB are automated by Orchestrator, and the user doesn't have to worry about the details.
- Orchestrator can also automatically perform a failover in case a master crashes or is unreachable by its replicas. If that is the case, Orchestrator will promote one of the replicas to a master. The replica to promote is chosen based on several criteria, like the server versions, the [binary log](#) formats in use and the datacenters locations.

Note that, if we don't want to use Orchestrator to automate operations, we can still use it as a dynamic inventory. Other tools can use it to obtain a list of existing MariaDB servers via its REST API or CLI commands.

Orchestrator has several big users, listed in the documentation [Users](#) page. It is also included in the PMM monitoring solution.

To install Orchestrator, see:

- The [install.md](#) for a manual installation;
- The links in [README.md](#), to install Orchestrator using automation tools.

## Contents

1. [Supported Topologies](#)
2. [Architecture](#)
3. [CLI Examples](#)
4. [Orchestrator Resources and References](#)

## Supported Topologies

Currently, Orchestrator fully supports MariaDB [GTID](#), [replication](#), and [semi-synchronous replication](#). While Orchestrator does not support Galera specific logic, it works with Galera clusters. For details, see [Supported Topologies and Versions](#) in Orchestrator documentation.

## Architecture

Orchestrator consists of a single executable called

```
orchestrator
```

. This is a process that periodically connects to the target servers. It will run SQL queries against target servers, so it needs a user with proper permissions. When the process is running, a GUI is available via a web browser, at the URL '<https://localhost:3000>'. It also exposes a REST API (see [Using the web API](#) in the Orchestrator documentation).

Orchestrator expects to find a JSON configuration file called

```
orchestrator.conf.json
, in
/etc
```

A database is used to store the configuration and the state of the target servers. By default, this is done using built-in SQLite. However, it is possible to use an external MariaDB or MySQL server instance.

If a cluster of Orchestrator instances is running, only one central database is used. One Orchestrator node is active, while the others are passive and are only used for failover. If the active node crashes or becomes unreachable, one of the other nodes becomes the active instance. The

```
active_node
table shows which node is active. Nodes communicate between them using the Raft protocol.
```

## CLI Examples

As mentioned, Orchestrator can be used from the command-line. Here you can find some examples.

List clusters:

```
orchestrator -c clusters
```

Discover a specified instance and add it to the known topology:

```
orchestrator -c discover -i <host>:<port>
```

Forget about an instance:

```
orchestrator -c topology -i <host>:<port>
```

Move a replica to a different master:

```
orchestrator -c move-up -i <replica-host>:<replica-port> -d <master-host>:<master-port>
```

Move a replica up, so that it becomes a "sibling" of its master:

```
orchestrator -c move-up -i <replica-host>:<replica-port>
```

Move a replica down, so that it becomes a replica of its "sibling":

```
orchestrator -c move-below -i <replica-host>:<replica-port> -d <master-host>:<master-port>
```

Make a node read-only:

```
orchestrator -c set-read-only -i <host>:<port>
```

Make a node writeable:

```
orchestrator -c set-writeable -i <host>:<port>
```

The

```
--debug  
and  
--stack
```

options can be added to the above commands to make them more verbose.

## Orchestrator Resources and References

- [Orchestrator on GitHub](#) .
- [Documentation](#) .
- [Raft consensus protocol website](#) .

The [README.md](#) file lists some related community projects, including modules to automate Orchestrator with [Puppet](#) and other technologies.

On GitHub you can also find links to projects that allow the use of automation software to deploy and manage Orchestrator.

---

Content initially contributed by [Vettabase Ltd](#) .

### 2.1.2.14.13 Rotating Logs on Unix and Linux

#### Contents

1. [Configuring Locations and File Names of Logs](#)
2. [Configuring Authentication for Logrotate](#)
3. [Configuring Logrotate](#)
4. [Testing Log Rotation](#)
5. [Logrotate in Ansible](#)

Unix and Linux distributions offer the

[logrotate](#)

utility, which makes it very easy to rotate log files. This page will describe how to configure log rotation for the [error log](#) , [general query log](#) , and the [slow query log](#) .

## Configuring Locations and File Names of Logs

The first step is to configure the locations and file names of logs. To make the log rotation configuration easier, it can be best to put these logs in a dedicated log directory.

We will need to configure the following:

- The [error log](#) location and file name is configured with the

[log\\_error](#)

system variable.

- The [general query log](#) location and file name is configured with the

[general\\_log\\_file](#)

system variable.

- The [slow query log](#) location and file name is configured with the

[slow\\_query\\_log\\_file](#)

system variable.

If you want to enable the [general query log](#) and [slow query log](#) immediately, then you will also have to configure the following:

- The [general query log](#) is enabled with the

[general\\_log](#)

system variable.

- The [slow query log](#) is enabled with the

[slow\\_query\\_log](#)

system variable.

These options can be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example, if we wanted to put our log files in

/var/log/mysql/

, then we could configure the following:

```
[mariadb]
...
log_error=/var/log/mysql/mariadb.err
general_log
general_log_file=/var/log/mysql/mariadb.log
slow_query_log
slow_query_log_file=/var/log/mysql/mariadb-slow.log
long_query_time=5
```

We will also need to create the relevant directory:

```
sudo mkdir /var/log/mysql/
sudo chown mysql:mysql /var/log/mysql/
sudo chmod 0770 /var/log/mysql/
```

If you are using [SELinux](#), then you may also need to set the SELinux context for the directory. See [SELinux: Setting the File Context for Log Files](#) for more information. For example:

```
sudo semanage fcontext -a -t mysqld_log_t "/var/log/mysql(/.*)?"
sudo restorecon -Rv /var/log/mysql
```

After MariaDB is [restarted](#), it will use the new log locations and file names.

## Configuring Authentication for Logrotate

The

[logrotate](#)

utility needs to be able to authenticate with MariaDB in order to flush the log files.

The easiest way to allow the

[logrotate](#)

utility to authenticate with MariaDB is to configure the

root@localhost

user account to use

[unix\\_socket](#)

authentication.

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, the

root@localhost

user account is configured to use

[unix\\_socket](#)

authentication by default, so this part can be skipped in those versions.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, a user account is only able to have one authentication method at a time. In these versions, this means that once you enable

[unix\\_socket](#)

authentication for the

root@localhost

user account, you will no longer be able to use a password to log in with that user account. The user account will only be able to use

`unix_socket`

authentication.

In MariaDB 10.3 and before, you need to [install the `unix\_socket` plugin](#) before you can configure the `root@localhost` user account to use it. For example:

```
INSTALL SONAME 'auth_socket';
```

After the plugin is installed, the

`root@localhost` user account can be configured to use

`unix_socket`

authentication. How this is done depends on the version of MariaDB.

MariaDB starting with 10.2

In MariaDB 10.2 and later, the

`root@localhost` user account can be altered to use

`unix_socket`

authentication with the

`ALTER USER`

statement. For example:

```
ALTER USER 'root'@'localhost' IDENTIFIED VIA unix_socket;
```

MariaDB until 10.1

In MariaDB 10.1 and before, the

`root@localhost` user account can be altered to use

`unix_socket`

authentication with the

`GRANT`

statement. For example:

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED VIA unix_socket WITH GRANT OPTION;
```

## Configuring Logrotate

At this point, we can configure the

`logrotate`

utility to rotate the log files.

On many systems, the primary

`logrotate`

configuration file is located at the following path:

- `/etc/logrotate.conf`

And the

### logrotate

configuration files for individual services are located in the following directory:

- /etc/logrotate.d/

We can create a

### logrotate

configuration file for MariaDB by executing the following command in a shell:

```
$ sudo tee /etc/logrotate.d/mariadb <<EOF
/var/log/mysql/* {
    su mysql mysql
    missingok
    create 660 mysql mysql
    notifempty
    daily
    minsize 1M # only use with logrotate >= 3.7.4
    maxsize 100M # only use with logrotate >= 3.8.1
    rotate 30
    # dateext # only use if your logrotate version is compatible with below dateformat
    # dateformat .%Y-%m-%d-%H-%M-%S # only use with logrotate >= 3.9.2
    compress
    delaycompress
    sharedscripts
    olddir archive/
    createolddir 770 mysql mysql # only use with logrotate >= 3.8.9
postrotate
    # just if mysqld is really running
    if test -x /usr/bin/mysqladmin && \
        /usr/bin/mysqladmin ping &>/dev/null
    then
        /usr/bin/mysqladmin --local flush-error-log \
            flush-engine-log flush-general-log flush-slow-log
    fi
endscript
}
EOF
```

You may have to modify this configuration file to use it on your system, depending on the specific version of the

### logrotate

utility that is installed. See the description of each configuration directive below to determine which

### logrotate

versions support that configuration directive.

Each specific configuration directive does the following:

- **missingok**  
: This directive configures it to ignore missing files, rather than failing with an error.
- **create 660 mysql mysql**  
: This directive configures it to recreate the log files after log rotation with the specified permissions and owner.
- **notifempty**  
: This directive configures it to skip a log file during log rotation if it is empty.
- **daily**  
: This directive configures it to rotate each log file once per day.
- **minsize 1M**  
: This directive configures it to skip a log file during log rotation if it is smaller than 1 MB. This directive is only available with

## `logrotate`

3.7.4 and later.

- `maxsize 100M`

: This directive configures it to rotate a log file more frequently than daily if it grows larger than 100 MB. This directive is only available with

## `logrotate`

3.8.1 and later.

- `rotate 30`

: This directive configures it to keep 30 old copies of each log file.

- `dateext`

: This directive configures it to use the date as an extension, rather than just a number. This directive is only available with

## `logrotate`

3.7.6 and later.

- `dateformat .%Y-%m-%d-%H-%M-%S`

: This directive configures it to use this date format string (as defined by the format specification for

## `strftime`

) for the date extension configured by the

`dateext` directive. This directive is only available with

## `logrotate`

3.7.7 and later. Support for

`%H`

is only available with

## `logrotate`

3.9.0 and later. Support for

`%M`

and

`%S`

is only available with

## `logrotate`

3.9.2 and later.

- `compress`

: This directive configures it to compress the log files with

## `gzip`

- `delaycompress`

: This directive configures it to delay compression of each log file until the next log rotation. If the log file is compressed at the same time that it is rotated, then there may be cases where a log file is being compressed while the MariaDB server is still writing to the log file. Delaying compression of a log file until the next log rotation can prevent race conditions such as these that can happen between the compression operation and the MariaDB server's log flush operation.

- `olddir archive/`

: This directive configures it to archive the rotated log files in  
`/var/log/mysql/archive/`

- `createolddir 770 mysql mysql`

: This directive configures it to create the directory specified by the  
`olddir` directive with the specified permissions and owner, if the directory does not already exist. This directive is only available with

### `logrotate`

3.8.9 and later.

- **sharedscripts**  
: This directive configures it to run the `postrotate` script just once, rather than once for each rotated log file.
- **postrotate**  
: This directive configures it to execute a script after log rotation. This particular script executes the `mysqladmin` utility, which executes the `FLUSH`

statement, which tells the MariaDB server to flush its various log files. When MariaDB server flushes a log file, it closes its existing file handle and reopens a new one. This ensures that MariaDB server does not continue writing to a log file after it has been rotated. This is an important component of the log rotation process.

If our system does not have

### `logrotate`

3.8.9 or later, which is needed to support the `createolddir` directive, then we will also need to create the relevant directory specified by the `olddir` directive:

```
sudo mkdir /var/log/mysql/archive/
sudo chown mysql:mysql /var/log/mysql/archive/
sudo chmod 0770 /var/log/mysql/archive/
```

## Testing Log Rotation

We can test log rotation by executing the

### `logrotate`

utility with the `--force` option. For example:

```
sudo logrotate --force /etc/logrotate.d/mariadb
```

Keep in mind that under normal operation, the

### `logrotate`

utility may skip a log file during log rotation if the utility does not believe that the log file needs to be rotated yet. For example:

- If you set the `notifempty` directive mentioned above, then it will be configured to skip a log file during log rotation if the log file is empty.
- If you set the `daily` directive mentioned above, then it will be configured to only rotate each log file once per day.
- If you set the `minsize 1M` directive mentioned above, then it will be configured to skip a log file during log rotation if the log file size is smaller than 1 MB.

However, when running tests with the

`--force` option, the

### `logrotate`

utility does not take these options into consideration.

After a few tests, we can see that the log rotation is indeed working:

```
$ sudo ls -l /var/log/mysql/archive/
total 48
-rw-rw---- 1 mysql mysql 440 Mar 31 15:31 mariadb.err.1
-rw-rw---- 1 mysql mysql 138 Mar 31 15:30 mariadb.err.2.gz
-rw-rw---- 1 mysql mysql 145 Mar 31 15:28 mariadb.err.3.gz
-rw-rw---- 1 mysql mysql 1007 Mar 31 15:27 mariadb.err.4.gz
-rw-rw---- 1 mysql mysql 1437 Mar 31 15:32 mariadb.log.1
-rw-rw---- 1 mysql mysql 429 Mar 31 15:31 mariadb.log.2.gz
-rw-rw---- 1 mysql mysql 439 Mar 31 15:28 mariadb.log.3.gz
-rw-rw---- 1 mysql mysql 370 Mar 31 15:27 mariadb.log.4.gz
-rw-rw---- 1 mysql mysql 3915 Mar 31 15:32 mariadb-slow.log.1
-rw-rw---- 1 mysql mysql 554 Mar 31 15:31 mariadb-slow.log.2.gz
-rw-rw---- 1 mysql mysql 569 Mar 31 15:28 mariadb-slow.log.3.gz
-rw-rw---- 1 mysql mysql 487 Mar 31 15:27 mariadb-slow.log.4.gz
```

## Logrotate in Ansible

Let's see an example of how to configure logrotate in Ansible.

First, we'll create a couple of tasks in our playbook:

```
- name: Create mariadb_logrotate_old_dir
  file:
    path: "{{ mariadb_logrotate_old_dir }}"
    owner: mysql
    group: mysql
    mode: '770'
    state: directory

- name: Configure logrotate
  template:
    src: "../templates/logrotate.j2"
    dest: "/etc/logrotate.d/mysql"
```

The first task creates a directory to store the old, compressed logs, and set proper permissions.

The second task uploads logrotate configuration file into the proper directory, and calls it

```
mysql
.
As you can see the original name is different, and it ends with the
.j2
extension, because it is a Jinja 2 template.
```

The file will look like the following:

```
{{ mariadb_log_dir }}/* {
  su mysql mysql
  missingok
  create 660 mysql mysql
  notifempty
  daily
  minsize 1M {{ mariadb_logrotate_min_size }}
  maxsize 100M {{ mariadb_logrotate_max_size }}
  rotate {{ mariadb_logrotate_old_dir }}
  dateformat %Y-%m-%d-%H-%M-%S # only use with logrotate >= 3.9.2
  compress
  delaycompress
  sharedscripts
  olddir archive/
  createolddir 770 mysql mysql # only use with logrotate >= 3.8.9
postrotate
  # just if mysqld is really running
  if test -x /usr/bin/mysqladmin && \
    /usr/bin/mysqladmin ping &>/dev/null
  then
    /usr/bin/mysqladmin --local flush-error-log \
      flush-engine-log flush-general-log flush-slow-log
  fi
endscript
}
```

The file is very similar to the file shown above, which is obvious because we're still uploading a logrotate configuration file. Ansible is just a tool we've

chosen to do this.

However, both in the tasks and in the template, we used some variables. This allows to use different paths and rotation parameters for different hosts, or host groups.

If we have a group host called

```
mariadb
that contains the default configuration for all our MariaDB servers, we can define these variables in a file called
group_vars/mariadb.yml
:
```

```
# MariaDB writes its logs here
mariadb_log_dir: /var/lib/mysql/logs

# logrotate configuration

mariadb_logrotate_min_size: 500M
mariadb_logrotate_max_size: 1G
mariadb_logrotate_old_files: 7
mariadb_logrotate_old_dir: /var/mysql/old-logs
```

After setting up logrotate in Ansible, you may want to deploy it to a non-production server and test it manually as explained above. Once you're sure that it works fine on one server, you can be confident in the new Ansible tasks and deploy them on all servers.

For more information on how to use Ansible to automate MariaDB configuration, see [Ansible and MariaDB](#).

## 2.1.2.14.14 Automating MariaDB Tasks with Events

MariaDB has an event scheduler that can be used to automate tasks, making them run at regular intervals of time. This page is about using events for [automation](#). For more information about events themselves, and how to work with them, see [event scheduler](#).

### Pro's and Con's of Using Events for Automation

Events can be compared to Unix cron jobs or Windows scheduled tasks. MariaDB events have at least the following benefits compared to those tools:

- Events are system-independent.
- Events are written in procedural SQL. There is no need for other languages.
- Events run in MariaDB. An implication, for example, is that the results of queries remain in MariaDB itself and are not sent to a client.

Some drawbacks of using events are the following:

- Events can only perform tasks that can be developed in SQL. So, for example, it is not possible to send alerts. Access to files or remote databases is limited.
- The event scheduler is a single thread. While this thread is busy running an event, other events cannot start. They are just skipped, not postponed. This can be a big problem when automating a number of tasks that should run in a limited time range.
- For more events limitations, see [Event Limitations](#).

In many cases you may prefer to develop scripts in an external programming language. However, you should know that simple tasks consisting of a few queries can easily be implemented as events.

## Good Practices

When using events to automate tasks, there are good practices one may want to follow.

Move your SQL code in a stored procedure. All the event will do is to call a stored procedures. Several events may call the same stored procedure, maybe with different parameters. The procedure may also be called manually, if necessary. This will avoid code duplication. This will separate the logic from the schedule, making it possible to change an event without a risk of making changes to the logic, and the other way around.

Just like cron jobs, events should log whether if they succeed or not. Logging debug messages may also be useful for non-trivial events. This information can be logged into a dedicated table. The contents of the table can be monitored by a monitoring tool like Grafana. This allows to visualize in a dashboard the status of events, and send alerts in case of a failure.

## Examples

Some examples of tasks that could easily be automated with events:

- Copying data from a remote table to a local table by night, using the [CONNECT](#) storage engine. This can be a good idea if many rows need be copied, because data won't be sent to an external client.
- Periodically delete historical data. For example, rows that are older than 5 years. Nothing prevents us from doing this with an external script, but probably this wouldn't add any value.
- Periodically delete invalid rows. In an e-commerce, they could be abandoned carts. In a messaging system, they could be messages to users that don't exist anymore.
- Add a new [partition](#) to a table and drop the oldest one.

## 2.1.2.15 MariaDB Package Repository Setup and Usage

If you are looking to set up MariaDB Server, it is often easiest to use a repository. The MariaDB Foundation has a repository configuration tool at <https://downloads.mariadb.org/> and MariaDB Corporation provides a convenient shell script to configure access to their MariaDB Package Repositories. It is available at:

- [https://r.mariadb.com/downloads/mariadb\\_repo\\_setup](https://r.mariadb.com/downloads/mariadb_repo_setup)

The script by default sets up 3 different repositories in a single repository configuration file. The repositories are

- MariaDB Server Repository
- MariaDB MaxScale Repository
- MariaDB Tools Repository

The script can be executed in the following way:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash
```

For the script to work, the

curl

and

ca-certificates

packages need to be installed on your system. Additionally on Debian and Ubuntu the

apt-transport-https

package needs to be installed. The script will check if these are installed and let you know before it attempts to create the repository configuration on your system.

### Contents

1. [Repositories](#)
  1. [MariaDB Repository](#)
  2. [MariaDB MaxScale Repository](#)
2. [Supported Distributions](#)
3. [Using the MariaDB Package Repository Setup Script](#)
  1. [Options](#)
    1. [--mariadb-server-version](#)
    2. [--mariadb-maxscale-version](#)
    3. [--os-type and --os-version](#)
    4. [--write-to-stdout](#)
  2. [Platform-Specific Behavior](#)
    1. [Platform-Specific Behavior on RHEL and CentOS](#)
    2. [Platform-Specific Behavior on Debian and Ubuntu](#)
    3. [Platform-Specific Behavior on SLES](#)
4. [Installing Packages with the MariaDB Package Repository](#)
  1. [Installing Packages on RHEL and CentOS](#)
  2. [Installing Packages on Debian and Ubuntu](#)
  3. [Installing Packages on SLES](#)
5. [Versions](#)

## Repositories

The script will set up 2 different repositories in a single repository configuration file.

### MariaDB Repository

The **MariaDB Repository** contains software packages related to MariaDB Server, including the server itself, [clients and utilities](#), [client libraries](#), [plugins](#), and [Mariabackup](#).

The binaries in MariaDB Corporation's **MariaDB Repository** are currently identical to the binaries in MariaDB Foundation's MariaDB Repository that is configured with the [MariaDB Repository Configuration Tool](#).

By default, the script will configure your system to install from the repository of the latest GA version of MariaDB. That is currently [MariaDB 10.5](#). If a new major GA release occurs and you would like to upgrade to it, then you will need to either manually edit the repository configuration file to point to the new version, or run the MariaDB Package Repository setup script again.

The script can also configure your system to install from the repository of a different version of MariaDB if you use the

```
--mariadb-server-version
```

option.

If you would not like to configure the **MariaDB Repository** on your system, then you can use the

```
--skip-server
```

option to prevent the MariaDB Package Repository setup script from configuring it.

## MariaDB MaxScale Repository

The **MariaDB MaxScale Repository** contains software packages related to **MariaDB MaxScale**.

By default, the script will configure your system to install from the repository of the *latest* GA version of MariaDB MaxScale. When a new major GA release occurs, the repository will automatically switch to the new version. If instead you would like to stay on a particular version you will need to manually edit the repository configuration file and change '

```
latest
```

' to the version you want (e.g. '

```
6.1
```

') or run the MariaDB Package Repository setup script again, specifying the particular version or series you want.

Older versions of the MariaDB Package Repository setup script would configure a specific MariaDB MaxScale series in the repository (i.e. '

```
2.4
```

'), so if you used the script in the past to set up your repository and want MariaDB MaxScale to automatically use the latest GA version then change '

```
2.4
```

' or '

```
2.3
```

' in the repository configuration to '

```
latest
```

'. Or download the current version of the script and re-run it to set up the repository again.

The script can configure your system to install from the repository of an older version of MariaDB MaxScale if you use the

```
--mariadb-maxscale-version
```

option. For example,

```
--mariadb-maxscale-version=2.4
```

if you want the latest release in the MariaDB MaxScale 2.4.x series.

If you do not want to configure the **MariaDB MaxScale Repository** on your system, then you can use the

```
--skip-maxscale
```

option to prevent the MariaDB Package Repository setup script from configuring it.

MariaDB MaxScale is licensed under the [Business Source License 1.1](#), so it is not entirely free to use for organizations who do not have a subscription with MariaDB Corporation. If you would like more information, see the information at [MariaDB Business Source License \(BSL\)](#): [Frequently Asked Questions](#). If you would like to know how much a subscription to use MariaDB MaxScale would cost, see [MariaDB Corporation's subscription pricing](#).

## Supported Distributions

The script supports Linux distributions that are officially supported by MariaDB Corporation's [MariaDB TX subscription](#). However, a MariaDB TX subscription with MariaDB Corporation is not required to use the MariaDB Package Repository.

The distributions currently supported by the script include:

- Red Hat Enterprise Linux (RHEL) 7 and 8
- CentOS 7
- Debian 9 (Stretch), 10 (Buster), 11 (Bullseye)
- Ubuntu 18.04 LTS (Bionic), and 20.04 LTS (Focal)
- SUSE Linux Enterprise Server (SLES) 12 and 15

To install MariaDB on distributions not supported by the MariaDB Package Repository setup script, please consider using MariaDB Foundation's [MariaDB Repository Configuration Tool](#). Some Linux distributions also include MariaDB [in their own repositories](#).

## Using the MariaDB Package Repository Setup Script

The script can be executed in the following way:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash
```

The script will have to set up package repository configuration files, so it will need to be executed as root.

The script will also install the GPG public keys used to verify the signature of MariaDB software packages. If you want to avoid that, then you can use the `--skip-key-import` option.

If the script tries to create the repository configuration file and one with that name already exists, then the script will rename the existing file with an extension in the format ".old\_[0-9]+", which would make the OS's package manager ignore the file. You can safely remove those files after you have confirmed that the updated repository configuration file works..

If you want to see the repository configuration file that would be created without actually doing so, then you can use the

```
--write-to-stdout
```

option. This also prevents the need to run the script as root,

If you want to download the script, rather than executing it, then you can do so in the following way:

```
curl -LO https://r.mariadb.com/downloads/mariadb_repo_setup
```

## Options

To provide options to the script, you must tell bash to expect them by executing bash with the options

```
-s --
```

, for example:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --help
```

Option	Description
<code>--help</code>	Display a usage message and exit.
<code>--mariadb-server-version=&lt;version&gt;</code>	Override the default MariaDB Server version. By default, the script will use 'mariadb-10.5'.
<code>--mariadb-maxscale-version=&lt;version&gt;</code>	Override the default MariaDB MaxScale version. By default, the script will use 'latest'.
<code>--os-type=&lt;type&gt;</code>	Override detection of OS type. Acceptable values include debian , ubuntu , rhel , and sles .
<code>--os-version=&lt;version&gt;</code>	Override detection of OS version. Acceptable values depend on the OS type you specify.
<code>--skip-key-import</code>	Skip importing GPG signing keys.
<code>--skip-maxscale</code>	Skip the 'MaxScale' repository.
<code>--skip-server</code>	Skip the 'MariaDB Server' repository.
<code>--skip-tools</code>	Skip the 'Tools' repository.

--skip-check-installed	Skip tests for required prerequisites for this script.
--write-to-stdout	Write output to stdout instead of to the OS's repository configuration file. This will also skip importing GPG public keys and updating the package cache on platforms where that behavior exists.

#### --mariadb-server-version

By default, the script will configure your system to install from the repository of the latest GA version of MariaDB. That is currently [MariaDB 10.5](#). If a new major GA release occurs and you would like to upgrade to it, then you will need to either manually edit the repository configuration file to point to the new version, or run the MariaDB Package Repository setup script again.

The script can also configure your system to install from the repository of a different version of MariaDB if you use the

```
--mariadb-server-version  
option.
```

The string

```
mariadb-
```

has to be prepended to the version number. For example, to configure your system to install from the repository of [MariaDB 10.3](#), that would be:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --mariadb-server-version="mariadb-10.6"
```

The following MariaDB versions are currently supported:

- mariadb-10.2
- mariadb-10.3
- mariadb-10.4
- mariadb-10.5
- mariadb-10.6
- mariadb-10.7

If you want to pin the repository of a specific minor release, such as [MariaDB 10.3.9](#), then you can also specify the minor release. For example,

```
mariadb-10.6.4
```

This may be helpful if you want to avoid upgrades. However, avoiding upgrades is not recommended, since minor releases can contain important bug fixes and fixes for security vulnerabilities.

#### --mariadb-maxscale-version

By default, the script will configure your system to install from the repository of the latest GA version of MariaDB MaxScale.

If you would like to pin the repository to a specific version of MariaDB MaxScale then you will need to either manually edit the repository configuration file to point to the desired version, or use the

```
--mariadb-maxscale-version  
option.
```

For example, to configure your system to install from the repository of MariaDB MaxScale 6.1, that would be:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --mariadb-maxscale-version="6.1"
```

The following MariaDB MaxScale versions are currently supported:

- MaxScale 1.4
- MaxScale 2.0
- MaxScale 2.1
- MaxScale 2.2

- MaxScale 2.3
- MaxScale 2.4
- MaxScale 2.5
- MaxScale 6.1
- MaxScale 6.2

The special identifiers

```
latest  
(for the latest GA release) and  
beta  
(for the latest beta release) are also supported. By default the  
mariadb_repo_setup  
script uses  
latest  
as the version.
```

```
--os-type  
and  
--os-version
```

If you want to run this script on an unsupported OS that you believe to be package-compatible with an OS that is supported, then you can use the

```
--os-type  
and  
--os-version  
options to override the script's OS detection. If you use either option, then you must use both options.
```

The supported values for

```
--os-type  
are:
```

- rhel
- debian
- ubuntu
- sles

If you use a non-supported value, then the script will fail, just as it would fail if you ran the script on an unsupported OS.

The supported values for

```
--os-version  
are entirely dependent on the OS type.
```

For Red Hat Enterprise Linux (RHEL) and CentOS,

```
7  
and  
8  
are valid options.
```

For Debian and Ubuntu, the version must be specified as the codename of the specific release. For example, Debian 9 must be specified as

```
stretch  
, and Ubuntu 18.04 must be specified as  
bionic
```

These options can be useful if your distribution is a fork of another distribution. As an example, Linux Mint 8.1 is based on and is fully compatible with Ubuntu 16.04 LTS (Xenial). Therefore, If you are using Linux Mint 8.1, then you can configure your system to install from the repository of Ubuntu 16.04 LTS (Xenial). If you would like to do that, then you can do so by specifying

```
--os-type=ubuntu  
and  
--os-version=xenial  
to the MariaDB Package Repository setup script.
```

For example, to manually set the

```
--os-type  
and  
--os-version
```

to RHEL 8 you could do:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --os-type=rhel --os-version=8
```

--write-to-stdout

The

--write-to-stdout

option will prevent the script from modifying anything on the system. The repository configuration will not be written to the repository configuration file. Instead, it will be printed to standard output. That allows the configuration to be reviewed, redirected elsewhere, consumed by another script, or used in some other way.

The

--write-to-stdout

option automatically enables

--skip-key-import

For example:

```
curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --write-to-stdout
```

## Platform-Specific Behavior

### Platform-Specific Behavior on RHEL and CentOS

On Red Hat Enterprise Linux (RHEL) and CentOS, the MariaDB Package Repository setup script performs the following tasks:

- Creates a repository configuration file at

/etc/yum.repos.d/mariadb.repo

- Imports the GPG public key used to verify the signature of MariaDB software packages with

```
rpm --import  
from  
downloads.mariadb.com
```

### Platform-Specific Behavior on Debian and Ubuntu

On Debian and Ubuntu, the MariaDB Package Repository setup script performs the following tasks:

- Creates a repository configuration file at

/etc/apt/sources.list.d/mariadb.list

- Creates a package preferences file at

/etc/apt/preferences.d/mariadb-enterprise.pref

, which gives packages from MariaDB repositories a higher priority than packages from OS and other repositories, which can help avoid conflicts. It looks like the following:

```
Package: *\nPin: origin downloads.mariadb.com\nPin-Priority: 1000
```

- Imports the GPG public key used to verify the signature of MariaDB software package with

```
apt-key  
from the  
keyserver.ubuntu.com  
key server.
```

- Updates the package cache with package definitions from the MariaDB Package Repository with

apt-get update

### Platform-Specific Behavior on SLES

On SUSE Linux Enterprise Server (SLES), the MariaDB Package Repository setup script performs the following tasks:

- Creates a repository configuration file at

/etc/zypp/repos.d/mariadb.repo

- Imports the GPG public key used to verify the signature of MariaDB software packages with

```
rpm --import  
from  
downloads.mariadb.com
```

.

## Installing Packages with the MariaDB Package Repository

After setting up the MariaDB Package Repository, you can install the software packages in the supported repositories.

### Installing Packages on RHEL and CentOS

To install MariaDB on Red Hat Enterprise Linux (RHEL) and CentOS, see the instructions at [Installing MariaDB Packages with YUM](#). For example:

```
sudo yum install MariaDB-server MariaDB-client MariaDB-backup
```

To install MariaDB MaxScale on Red Hat Enterprise Linux (RHEL) and CentOS, see the instructions at [MariaDB MaxScale Installation Guide](#). For example:

```
sudo yum install maxscale
```

### Installing Packages on Debian and Ubuntu

To install MariaDB on Debian and Ubuntu, see the instructions at [Installing MariaDB Packages with APT](#). For example:

```
sudo apt-get install mariadb-server mariadb-client mariadb-backup
```

To install MariaDB MaxScale on Debian and Ubuntu, see the instructions at [MariaDB MaxScale Installation Guide](#). For example:

```
sudo apt-get install maxscale
```

### Installing Packages on SLES

To install MariaDB on SUSE Linux Enterprise Server (SLES), see the instructions at [Installing MariaDB Packages with ZYpp](#). For example:

```
sudo zypper install MariaDB-server MariaDB-client MariaDB-backup
```

To install MariaDB MaxScale on SUSE Linux Enterprise Server (SLES), see the instructions at [MariaDB MaxScale Installation Guide](#). For example:

```
sudo zypper install maxscale
```

## Versions

Version	sha256sum
2022-02-08	b9e90cde27affc2a44f9fc60e302ccfcacf71f4ae02071f30d570e6048c28597
2022-01-18	c330d2755e18e48c3bba300a2898b0fc8ad2d3326d50b64e02fe65c67b454599

### 2.1.3 Upgrading MariaDB

#### 2.1.3.1 Upgrading Between Major MariaDB Versions

## Contents

1. Requirements for Doing an Upgrade Between Major Versions
2. Recommended Steps
  1. Step by step instructions for upgrades
3. Work Done by `mysql_upgrade`
4. Post Upgrade Work
5. If Something Goes Wrong
  1. Disaster Recovery
6. Downgrading
7. See Also

MariaDB is designed to allow easy upgrades. You should be able to trivially upgrade from ANY earlier MariaDB version to the latest one (for example [MariaDB 5.5 .x](#) to [MariaDB 10.5 .x](#)), usually in a few seconds. This is also mainly true for any MySQL version < 8.0 to [MariaDB 10.4](#) and up.

Upgrades are normally easy because:

- All MariaDB table data files are backward compatible
- The MariaDB connection protocol is backward compatible. You don't normally need to upgrade any of your old clients to be able to connect to a newer MariaDB version.
- The MariaDB slave can be of any newer version than the master.

MariaDB Corporation regularly runs tests to check that one can upgrade from [MariaDB 5.5](#) to the latest MariaDB version without any trouble. All older versions should work too (as long as the storage engines you were using are still around).

Note that if you are using [MariaDB Galera Cluster](#), you have to follow the [Galera upgrading instructions](#) !

## Requirements for Doing an Upgrade Between Major Versions

- Go through the individual version upgrade notes (listed below) to look for any major changes or configuration options that have changed.
- Ensure that the target MariaDB version supports the storage engines you are using. For example, in 10.5 [TokuDB](#) is not supported.
- Backup the database (just in case). At least, take a copy of the

```
mysql
      data directory with mysqldump --add-drop-table mysql as most of the upgrade changes are done there (adding new fields and new system tables etc).
```
- Ensure that the `innodb_fast_shutdown` variable is not 2 (fast crash shutdown) or 3. The default of this variable is 1. The safest option for upgrades is 0, but the shutdown time may be notably larger with 0 than for 1 as there are a lot more cleanups done for 0.
- `innodb_force_recovery` must be less than

```
3
.
.
.
• Cleanly shutdown of the server. This is necessary because even if data files are compatible between versions, recovery logs may not be.
```

Note that rpms don't support upgrading between major versions, only minor like 10.4.1 to 10.4.2. If you are using rpms, you should de-install the old MariaDB rpms and install the new MariaDB rpms before running `mysql_upgrade`. Note that when installing the new rpms, `mysql_upgrade` may be run automatically. There is no problem with running `mysql_upgrade` many times.

## Recommended Steps

- If you have a [master-slave setup](#), first upgrade one slave and when you have verified that the slave works well, upgrade the rest of the slaves (if any). Then [upgrade one slave to master](#), upgrade the master, and change the master to a slave.
- If you don't have a master-slave setup, then [take a backup](#), [shutdown MariaDB](#) and do the upgrade.

## Step by step instructions for upgrades

- Upgrade MariaDB binaries and libraries, preferably without starting MariaDB.
- If the MariaDB server process, `mysqld` or `mariadb` was not started as part of the upgrade, start it by executing

```
mysqld --skip-grant-tables
.
.
.
This may produce some warnings about some system tables not being up to date, but you can ignore these for now as mysql_upgrade will fix that.
```
- Run `mysql_upgrade`
- Restart MariaDB server.

## Work Done by `mysql_upgrade`

The main work done when upgrading is done by running `mysql_upgrade`. The main things it does are:

- Updating the system tables in the

```
mysql
      database to the newest version. This is very quick.
```
- `mysql_upgrade` also runs `mysqlcheck --check-upgrade` to check if there have been any collation changes between the major versions. This recreates indexes in old tables that are using any of the changed collations. This can take a bit of time if there are a lot of tables or there are many

tables which used the changed collation. The last time a collation changed was in MariaDB/MySQL 5.1.23.

## Post Upgrade Work

Check the [MariaDB error log](#) for any problems during upgrade. If there are any warnings in the log files, do you best to get rid of them!

The common warnings/errors are:

- Using obsolete options. If this is the case, remove them from your [my.cnf files](#).
- Check the manual for [new features](#) that have been added since your last MariaDB version.
- Test that your application works as before. The main difference from before is that because of optimizer improvements your application should work better than before, but in some rare cases the optimizer may get something wrong. In this case, you can try to use [explain](#), [optimizer trace](#) or [optimizer\\_switch](#) to fix the queries.

## If Something Goes Wrong

- First, check the [MariaDB error log](#) to see if you are using configure options that are not supported anymore.
- Check the upgrade notices for the MariaDB release that you are upgrading to.
- File an issue in the [MariaDB bug tracker](#) so that we know about the issue and can provide a fix to make upgrades even better.
- Add a comment to this manual entry for how we can improve it.

## Disaster Recovery

In the unlikely event something goes wrong, you can try the following:

- Remove the InnoDB tables from the

```
mysql
      data directory. They are:
      o gtid_slave_pos
      o innodb_table_stats
      o innodb_index_stats
      o transaction_registry
```

- Move the

```
mysql
      data directory to
      mysql-old
      and run mysql\_install\_db to generate a new one.
```

- After the above, you have to add back your old users.
- When done, delete the

```
mysql-old
      data directory.
```

## Downgrading

MariaDB server is not designed for downgrading. That said, in most cases, as long as you haven't run any [ALTER TABLE](#) or [CREATE TABLE](#) statements and you have a [mysqldump](#) of your old

```
mysql
      database , you should be able to downgrade to your previous version by doing the following:
```

- Do a clean shutdown. For this special case you have to set [innodb\\_fast\\_shutdown](#) to 0, before taking down the new MariaDB server, to ensure there are no redo or undo logs that need to be applied on the downgraded server.
- Delete the tables in the

```
mysql
      database (if you didn't use the option
      --add-drop-table
      to mysqldump )
```

- Delete the new MariaDB installation
- Install the old MariaDB version
- Start the server with [mysqld --skip-grant-tables](#)
- Install the old

```
mysql
      database
      FLUSH PRIVILEGES
```

## See Also

- [Upgrading from MySQL to MariaDB](#)
- [Upgrading from MariaDB 10.5 to MariaDB 10.6](#)

- [Upgrading from MariaDB 10.4 to MariaDB 10.5](#)
- [Upgrading from MariaDB 10.3 to MariaDB 10.4](#)
- [Upgrading from MariaDB 10.2 to MariaDB 10.3](#)
- [Upgrading from MariaDB 10.1 to MariaDB 10.2](#)
- [Upgrading from MariaDB 10.0 to MariaDB 10.1](#)
- [Upgrading from MariaDB 5.5 to MariaDB 10.0](#)
- [Galera upgrading instructions](#)
- [innodb\\_fast\\_shutdown](#)

## 2.1.3.2 Upgrading Between Minor Versions on Linux

For Windows, see [Upgrading MariaDB on Windows instead](#).

For MariaDB Galera Cluster, see [Upgrading Between Minor Versions with Galera Cluster instead](#).

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

To upgrade between minor versions of MariaDB on Linux/Unix (for example from [MariaDB 10.3.12](#) to [MariaDB 10.3.13](#)), the following procedure is suggested:

1. [Stop MariaDB](#).
2. Uninstall the old version of MariaDB.
3. Install the new version of MariaDB.
  - On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.
4. Make any desired changes to configuration options in [option files](#), such as  
`my.cnf`

5. [Start MariaDB](#).
6. Run

`mysql_upgrade`

◦

`mysql_upgrade`  
does two things:

1. Ensures that the system tables in the

`mysq`

`1`

database are fully compatible with the new version.

2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB.

◦ In most cases this should be a fast operation (depending of course on the number of tables).

To upgrade between major versions, see the following:

- [Upgrading from MariaDB 10.4 to MariaDB 10.5](#)
- [Upgrading from MariaDB 10.3 to MariaDB 10.4](#)
- [Upgrading from MariaDB 10.2 to MariaDB 10.3](#)
- [Upgrading from MariaDB 10.1 to MariaDB 10.2](#)
- [Upgrading from MariaDB 10.0 to MariaDB 10.1](#)
- [Upgrading from MariaDB 5.5 to MariaDB 10.0](#)

## 2.1.3.3 Upgrading from MariaDB 10.6 to MariaDB 10.7

### Contents

1. [How to Upgrade](#)
2. [Incompatible Changes Between 10.6 and 10.7](#)
  1. [Compression](#)
  2. [Options That Have Been Removed or Renamed](#)
3. [See Also](#)

## How to Upgrade

For Windows, see [Upgrading MariaDB on Windows](#).

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs [MariaDB 10.7](#). For example,
  - On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
2. [Stop MariaDB](#).
3. Uninstall the old version of MariaDB.
  - On Debian, Ubuntu, and other similar Linux distributions, execute the following:

```
sudo apt-get remove mariadb-server
```
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:

```
sudo yum remove MariaDB-server
```
  - On SLES, OpenSUSE, and other similar Linux distributions, execute the following:

```
sudo zypper remove MariaDB-server
```
4. Install the new version of MariaDB.
  - On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.
5. Make any desired changes to configuration options in [option files](#), such as  
`my.cnf`  
. This includes removing any options that are no longer supported.
6. [Start MariaDB](#).
7. Run [mariadb-upgrade](#).
  - `mariadb-upgrade`  
does two things:
    1. Ensures that the system tables in the `mysql` database are fully compatible with the new version.
    2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

## Incompatible Changes Between 10.6 and 10.7

On most servers upgrading from 10.6 should be painless. However, there are some things that have changed which could affect an upgrade:

### Compression

If a non-zlib compression algorithm was used in [InnoDB](#) or [Mroonga](#) before upgrading to 10.7, those tables will be unreadable until the appropriate compression library is installed. See [Compression Plugins#Upgrading](#).

### Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your [option files](#):

Option	Reason
<code>wsrep_replicate_myisam</code>	Use <code>wsrep_mode</code> instead.
<code>wsrep_strict_ddl</code>	Use <code>wsrep_mode</code> instead.

### See Also

- The features in MariaDB 10.7
- Upgrading from MariaDB 10.5 to MariaDB 10.6
- Upgrading from MariaDB 10.4 to MariaDB 10.5
- Upgrading from MariaDB 10.3 to MariaDB 10.4

## 2.1.3.4 Upgrading from MariaDB 10.5 to MariaDB 10.6

### Contents

1. [How to Upgrade](#)
2. [Incompatible Changes Between 10.5 and 10.6](#)
  1. [Reserved Word](#)
  2. [InnoDB COMPRESSED Row Format](#)
  3. [Character Sets](#)
  4. [Options That Have Changed Default Values](#)
  5. [Options That Have Been Removed or Renamed](#)
  6. [Deprecated Options](#)
3. [Major New Features To Consider](#)
4. [See Also](#)

### How to Upgrade

For Windows, see [Upgrading MariaDB on Windows](#).

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs [MariaDB 10.6](#). For example,
  - On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
2. [Stop MariaDB](#).
3. Uninstall the old version of MariaDB.
  - On Debian, Ubuntu, and other similar Linux distributions, execute the following:
 

```
sudo apt-get remove mariadb-server
```
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:
 

```
sudo yum remove MariaDB-server
```
  - On SLES, OpenSUSE, and other similar Linux distributions, execute the following:
 

```
sudo zypper remove MariaDB-server
```
4. Install the new version of MariaDB.
  - On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.
5. Make any desired changes to configuration options in [option files](#), such as
  - `my.cnf`
  - . This includes removing any options that are no longer supported.
6. [Start MariaDB](#).
7. Run [mariadb-upgrade](#).
  - ```
mariadb-upgrade
```

 does two things:
    1. Ensures that the system tables in the `mysql` database are fully compatible with the new version.
    2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

### Incompatible Changes Between 10.5 and 10.6

On most servers upgrading from 10.5 should be painless. However, there are some things that have changed which could affect an upgrade:

## Reserved Word

- New reserved word : OFFSET. This can no longer be used as an identifier without being quoted.

## InnoDB COMPRESSED Row Format

From MariaDB 10.6.0 , tables that are of the COMPRESSED row format are read-only by default. This is the first step towards removing write support and deprecating the feature.

Set the innodb\_read\_only\_compressed variable to

OFF

to make the tables writable.

## Character Sets

From MariaDB 10.6 , the

utf8

character set (and related collations) is by default an alias for

utf8mb3

rather than the other way around. It can be set to imply

utf8mb4

by changing the value of the old\_mode system variable.

## Options That Have Changed Default Values

| Option                               | Old default value | New default value |
|--------------------------------------|-------------------|-------------------|
| character_set_client                 | utf8              | utf8mb3           |
| character_set_connection             | utf8              | utf8mb3           |
| character_set_results                | utf8              | utf8mb3           |
| character_set_system                 | utf8              | utf8mb3           |
| innodb_flush_method                  | fsync             | O_DIRECT          |
| [[server-system-variables/#old_mode] | old_mode]         | Empty             |
|                                      |                   | UTF8_IS_UTF8MB3   |

## Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your option files :

| Option                                      | Reason                                                                                                                                          |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| innodb_adaptive_max_sleep_delay             |                                                                                                                                                 |
| innodb_background_scrub_data_check_interval |                                                                                                                                                 |
| innodb_background_scrub_data_compressed     |                                                                                                                                                 |
| innodb_background_scrub_data_interval       |                                                                                                                                                 |
| innodb_background_scrub_data_uncompressed   |                                                                                                                                                 |
| innodb_buffer_pool_instances                |                                                                                                                                                 |
| innodb_checksum_algorithm                   | The variable is still present, but the *innodb and *none options have been removed as the crc32 algorithm only is supported from MariaDB 10.6 . |
| innodb_commit_concurrency                   |                                                                                                                                                 |
| innodb_concurrency_tickets                  |                                                                                                                                                 |
| innodb_file_format                          |                                                                                                                                                 |
| innodb_large_prefix                         |                                                                                                                                                 |
| innodb_lock_schedule_algorithm              |                                                                                                                                                 |
| innodb_log_checksums                        |                                                                                                                                                 |

|                             |
|-----------------------------|
| innodb_log_compressed_pages |
| innodb_log_files_in_group   |
| innodb_log_optimize_ddl     |
| innodb_page_cleaners        |
| innodb_replication_delay    |
| innodb_scrub_log            |
| innodb_scrub_log_speed      |
| innodb_sync_array_size      |
| innodb_thread_concurrency   |
| innodb_thread_sleep_delay   |
| innodb_undo_logs            |

## Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

| Option                 | Reason                                  |
|------------------------|-----------------------------------------|
| wsrep_replicate_myisam | Use <a href="#">wsrep_mode</a> instead. |
| wsrep_strict_ddl       | Use <a href="#">wsrep_mode</a> instead. |

## Major New Features To Consider

- See also [System Variables Added in MariaDB 10.6](#).

## See Also

- [The features in MariaDB 10.6](#)
- [Upgrading from MariaDB 10.4 to MariaDB 10.5](#)
- [Upgrading from MariaDB 10.3 to MariaDB 10.4](#)
- [Upgrading from MariaDB 10.2 to MariaDB 10.3](#)

## 2.1.3.5 Upgrading from MariaDB 10.4 to MariaDB 10.5

### Contents

1. [How to Upgrade](#)
2. [Incompatible Changes Between 10.4 and 10.5](#)
  1. [Binary name changes](#)
  2. [GRANT PRIVILEGE changes](#)
  3. [Options That Have Changed Default Values](#)
  4. [Options That Have Been Removed or Renamed](#)
  5. [Deprecated Options](#)
3. [Major New Features To Consider](#)
4. [See Also](#)

## How to Upgrade

For Windows, see [Upgrading MariaDB on Windows](#) instead.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs [MariaDB 10.5](#). For example,
  - On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.

- On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
2. [Stop MariaDB](#).
3. Uninstall the old version of MariaDB.
- On Debian, Ubuntu, and other similar Linux distributions, execute the following:
- ```
sudo apt-get remove mariadb-server
```
- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:
- ```
sudo yum remove MariaDB-server
```
- On SLES, OpenSUSE, and other similar Linux distributions, execute the following:
- ```
sudo zypper remove MariaDB-server
```
4. Install the new version of MariaDB.
- On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.
5. Make any desired changes to configuration options in [option files](#), such as
- ```
my.cnf
```
- . This includes removing any options that are no longer supported.
6. [Start MariaDB](#).
7. Run [mysql\\_upgrade](#).
- 

```
mysql_upgrade
does two things:
1. Ensures that the system tables in the# mysql database are fully compatible with the new version.
2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .
```

## Incompatible Changes Between 10.4 and 10.5

On most servers upgrading from 10.4 should be painless. However, there are some things that have changed which could affect an upgrade:

### Binary name changes

All binaries previously beginning with mysql now begin with mariadb, with symlinks for the corresponding mysql command.

Usually that shouldn't cause any changed behavior, but when starting the MariaDB server via [systemd](#), or via the [mysqld\\_safe](#) script symlink, the server process will now always be started as

```
mariadb
, not
mysqld
```

So anything looking for the

```
mysqld
name in the system process list, like e.g. monitoring solutions, now needs for
mariadb
instead when the server / service is not started directly, but via
mysqld_safe
or as a system service.
```

### GRANT PRIVILEGE changes

A number of statements changed the privileges that they require. The old privileges were historically inappropriately chosen in the upstream. 10.5.2 fixes this problem. Note, these changes are incompatible to previous versions. A number of GRANT commands might be needed after upgrade.

- SHOW BINLOG EVENTS  
now requires the  
BINLOG MONITOR  
privilege (required  
REPLICATION SLAVE  
prior to 10.5.2).
- SHOW SLAVE HOSTS  
now requires the  
REPLICATION MASTER ADMIN  
privilege (required

REPLICATION SLAVE  
prior to 10.5.2).

- SHOW SLAVE STATUS  
now requires the  
REPLICATION SLAVE ADMIN  
or the  
SUPER  
privilege (required  
REPLICATION CLIENT  
or  
SUPER  
prior to 10.5.2).

- SHOW RELAYLOG EVENTS  
now requires the  
REPLICATION SLAVE ADMIN  
privilege (required  
REPLICATION SLAVE  
prior to 10.5.2).

## Options That Have Changed Default Values

| Option                                | Old default value | New default value |
|---------------------------------------|-------------------|-------------------|
| innodb_adaptive_hash_index            | ON                | OFF               |
| innodb_checksum_algorithm             | crc32             | full_crc32        |
| innodb_log_optimize_ddl               | ON                | OFF               |
| slave_parallel_mode                   | conservative      | optimistic        |
| performance_schema_max_cond_classes   | 80                | 90                |
| performance_schema_max_file_classes   | 50                | 80                |
| performance_schema_max_mutex_classes  | 200               | 210               |
| performance_schema_max_rwlock_classes | 40                | 50                |
| performance_schema_setup_actors_size  | 100               | -1                |
| performance_schema_setup_objects_size | 100               | -1                |

## Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your [option files](#) :

| Option                         | Reason                                                                                                |
|--------------------------------|-------------------------------------------------------------------------------------------------------|
| innodb_checksums               | Deprecated and functionality replaced by <a href="#">innodb_checksum_algorithms</a> in MariaDB 10.0 . |
| idle_flush_pct                 | Has had no effect since merging InnoDB 5.7 from mysql-5.7.9 ( <a href="#">MariaDB 10.2.2</a> ).       |
| innodb_locks_unsafe_for_binlog | Deprecated in MariaDB 10.0 . Use <a href="#">READ COMMITTED</a> transaction isolation level instead.  |
| innodb_rollback_segments       | Deprecated and replaced by <a href="#">innodb_undo_logs</a> in MariaDB 10.0 .                         |
| innodb_stats_sample_pages      | Deprecated in MariaDB 10.0 . Use <a href="#">innodb_stats_transient_sample_pages</a> instead.         |
| max_long_data_size             | Deprecated and replaced by <a href="#">max_allowed_packet</a> in MariaDB 5.5 .                        |
| multi_range_count              | Deprecated and has had no effect since MariaDB 5.3 .                                                  |
| thread_concurrency             | Deprecated and has had no effect since MariaDB 5.5 .                                                  |
| timed_mutexes                  | Deprecated and has had no effect since MariaDB 5.5 .                                                  |

## Deprecated Options

The following options have been deprecated. They have not yet been removed, but will be in a future version, and should ideally no longer be used.

| Option                                      | Reason                                           |
|---------------------------------------------|--------------------------------------------------|
| innodb_adaptive_max_sleep_delay             | No need for thread throttling any more.          |
| innodb_background_scrub_data_check_interval | Problematic ‘background scrubbing’ code removed. |
| innodb_background_scrub_data_interval       | Problematic ‘background scrubbing’ code removed. |

|                                                        |                                                                                                        |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <code>innodb_background_scrub_data_compressed</code>   | Problematic ‘background scrubbing’ code removed.                                                       |
| <code>innodb_background_scrub_data_uncompressed</code> | Problematic ‘background scrubbing’ code removed.                                                       |
| <code>innodb_buffer_pool_instances</code>              | Having more than one buffer pool is no longer necessary.                                               |
| <code>innodb_commit_concurrency</code>                 | No need for thread throttling any more.                                                                |
| <code>innodb_concurrency_tickets</code>                | No need for thread throttling any more.                                                                |
| <code>innodb_log_files_in_group</code>                 | Redo log was unnecessarily split into multiple files. Limited to 1 from <a href="#">MariaDB 10.5</a> . |
| <code>innodb_log_optimize_ddl</code>                   | Prohibited optimizations.                                                                              |
| <code>innodb_page_cleaners</code>                      | Having more than one page cleaner task no longer necessary.                                            |
| <code>innodb_replication_delay</code>                  | No need for thread throttling any more.                                                                |
| <code>innodb_scrub_log</code>                          | Never really worked as intended, redo log format is being redone.                                      |
| <code>innodb_scrub_log_speed</code>                    | Never really worked as intended, redo log format is being redone.                                      |
| <code>innodb_thread_concurrency</code>                 | No need for thread throttling any more.                                                                |
| <code>innodb_thread_sleep_delay</code>                 | No need for thread throttling any more.                                                                |
| <code>innodb_undo_logs</code>                          | It always makes sense to use the maximum number of rollback segments.                                  |
| <code>large_page_size</code>                           | Unused since multiple page size support was added.                                                     |

## Major New Features To Consider

You might consider using the following major new features in [MariaDB 10.5](#):

- The [S3 storage engine](#) allows one to archive MariaDB tables in Amazon S3, or any third-party public or private cloud that implements S3 API.
- [ColumnStore](#) columnar storage engine.
- See also [System Variables Added in MariaDB 10.5](#).

## See Also

- [The features in MariaDB 10.5](#)
- [Upgrading from MariaDB 10.3 to MariaDB 10.4](#)
- [Upgrading from MariaDB 10.2 to MariaDB 10.3](#)
- [Upgrading from MariaDB 10.1 to MariaDB 10.2](#)

## 2.1.3.6 Upgrading from MariaDB 10.3 to MariaDB 10.4

### Contents

1. [How to Upgrade](#)
2. [Incompatible Changes Between 10.3 and 10.4](#)
  1. [Options That Have Changed Default Values](#)
  2. [Options That Have Been Removed or Renamed](#)
  3. [Authentication and TLS](#)
3. [Major New Features To Consider](#)
4. [See Also](#)

## How to Upgrade

For Windows, see [Upgrading MariaDB on Windows instead](#).

For MariaDB Galera Cluster, see [Upgrading from MariaDB 10.3 to MariaDB 10.4 with Galera Cluster instead](#).

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system’s package manager installs [MariaDB 10.4](#). For example,

- On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
2. [Stop MariaDB](#).
  3. Uninstall the old version of MariaDB.
    - On Debian, Ubuntu, and other similar Linux distributions, execute the following:

```
sudo apt-get remove mariadb-server
```

    - On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:

```
sudo yum remove MariaDB-server
```

    - On SLES, OpenSUSE, and other similar Linux distributions, execute the following:

```
sudo zypper remove MariaDB-server
```
  4. Install the new version of MariaDB.
    - On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
    - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
    - On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.
  5. Make any desired changes to configuration options in [option files](#), such as `my.cnf`
    - . This includes removing any options that are no longer supported.
  6. [Start MariaDB](#).
  7. Run

`mysql_upgrade`

- `mysql_upgrade`  
does two things:
1. Ensures that the system tables in the

```
mysq
1
database are fully compatible with the new version.
```

2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

## Incompatible Changes Between 10.3 and 10.4

On most servers upgrading from 10.3 should be painless. However, there are some things that have changed which could affect an upgrade:

### Options That Have Changed Default Values

| Option                                      | Old default value | New default value                             |
|---------------------------------------------|-------------------|-----------------------------------------------|
| <code>slave_transaction_retry_errors</code> | 1213,1205         | 1158,1159,1160,1161,1205,1213,1429,2013,12701 |
| <code>wsrep_debug</code>                    | OFF               | NONE                                          |
| <code>wsrep_load_data_splitting</code>      | ON                | OFF                                           |

### Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your [option files](#):

| Option | Reason |
|--------|--------|
|--------|--------|

### Authentication and TLS

- See [Authentication from MariaDB 10.4](#) for an overview of the changes.
- The [unix\\_socket authentication plugin](#) is now default on Unix-like systems.
- TLSv1.0 is disabled by default in [MariaDB 10.4](#) . See [tls\\_version](#) and [TLS Protocol Versions](#) .

## Major New Features To Consider

You might consider using the following major new features in MariaDB 10.4 :

- Galera has been upgraded from Galera 3 to Galera 4.
- System-versioning extended with support for application-time periods .
- User password expiry
- Account Locking
- See also [System Variables Added in MariaDB 10.4](#) .

## See Also

- [The features in MariaDB 10.4](#)
- [Upgrading from MariaDB 10.3 to MariaDB 10.4 with Galera Cluster](#)
- [Upgrading from MariaDB 10.2 to MariaDB 10.3](#)
- [Upgrading from MariaDB 10.1 to MariaDB 10.2](#)
- [Upgrading from MariaDB 10.0 to MariaDB 10.1](#)

## 2.1.3.7 Upgrading from MariaDB 10.2 to MariaDB 10.3

### Contents

1. [How to Upgrade](#)
2. [Incompatible Changes Between 10.2 and 10.3](#)
  1. [Options That Have Changed Default Values](#)
  2. [Options That Have Been Removed or Renamed](#)
  3. [Reserved Words](#)
  4. [SQL\\_MODE=ORACLE](#)
  5. [Functions](#)
  6. [mysqldump](#)
  7. [MariaDB Backup and Percona XtraBackup](#)
  8. [Privileges](#)
3. [Major New Features To Consider](#)
4. [See Also](#)

## How to Upgrade

For Windows, see [Upgrading MariaDB on Windows instead](#).

For MariaDB Galera Cluster, see [Upgrading from MariaDB 10.2 to MariaDB 10.3 with Galera Cluster instead](#).

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#) .

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.3 . For example,
  - On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
2. Stop MariaDB . The server should be cleanly shut down, with no incomplete transactions remaining. `innodb_fast_shutdown` must be set to
  - 0
  - or
  - 1
  - and `innodb_force_recovery` must be less than
  - 3
3. Uninstall the old version of MariaDB.
  - On Debian, Ubuntu, and other similar Linux distributions, execute the following:

```
sudo apt-get remove mariadb-server
```
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:

```
sudo yum remove MariaDB-server
```

- On SLES, OpenSUSE, and other similar Linux distributions, execute the following:

```
sudo zypper remove MariaDB-server
```

#### 4. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.

#### 5. Make any desired changes to configuration options in [option files](#), such as

```
my.cnf
```

. This includes removing any options that are no longer supported.

#### 6. [Start MariaDB](#).

#### 7. Run

```
mysql_upgrade
```

◦

```
mysql_upgrade
```

does two things:

1. Ensures that the system tables in the

```
mysq
```

```
l
```

database are fully compatible with the new version.

2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

## Incompatible Changes Between 10.2 and 10.3

On most servers upgrading from 10.2 should be painless. However, there are some things that have changed which could affect an upgrade:

### Options That Have Changed Default Values

| Option                               | Old default value | New default value                 |
|--------------------------------------|-------------------|-----------------------------------|
| innodb_flush_method                  | (empty)           | fsync                             |
| innodb_spin_wait_delay               | 6                 | 4                                 |
| performance_schema_max_stage_classes | 150               | 160                               |
| plugin_maturity                      | unknown           | One less than the server maturity |

### Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your [option files](#) :

| Option                           | Reason                                                                                                                                                                        |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| innodb_buffer_pool_populate      | Used in XtraDB-only                                                                                                                                                           |
| innodb_cleaner_lsn_age_factor    | Used in XtraDB-only                                                                                                                                                           |
| innodb_corrupt_table_action      | Used in XtraDB-only                                                                                                                                                           |
| innodb_empty_free_list_algorithm | Used in XtraDB-only                                                                                                                                                           |
| innodb_fake_changes              | Used in XtraDB-only                                                                                                                                                           |
| innodb_file_format               | The <a href="#">InnoDB file format</a> is now Barracuda, and the old Antelope file format is no longer supported.                                                             |
| innodb_file_format_check         | No longer necessary as the Antelope <a href="#">InnoDB file format</a> is no longer supported.                                                                                |
| innodb_file_format_max           | No longer necessary as the Antelope <a href="#">InnoDB file format</a> is no longer supported.                                                                                |
| innodb_foreground_preflush       | Used in XtraDB-only                                                                                                                                                           |
| innodb_instrument_semaphores     |                                                                                                                                                                               |
| innodb_kill_idle_transaction     | Used in XtraDB-only                                                                                                                                                           |
| innodb_large_prefix              | Large index key prefixes were made default from <a href="#">MariaDB 10.2</a> , and limiting tables to small prefixes is no longer permitted in <a href="#">MariaDB 10.3</a> . |

|                                                        |                                                                                                                                                          |
|--------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>innodb_locking_fake_changes</code>               | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_log_arch_dir</code>                       | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_log_arch_expire_sec</code>                | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_log_archive</code>                        | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_log_block_size</code>                     | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_log_checksum_algorithm</code>             | Translated to <code>innodb_log_checksums</code> (NONE to OFF, everything else to ON); only existed to allow easier upgrade from earlier XtraDB versions. |
| <code>innodb_mtflush_threads</code>                    | Replaced by the <code>innodb_page_cleaners</code> system variable.                                                                                       |
| <code>innodb_sched_priority_cleaner</code>             | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_show_locks_held</code>                    | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_show_verbose_locks</code>                 | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_support_xa</code>                         | XA transactions are always supported.                                                                                                                    |
| <code>innodb_use_fallocate</code>                      |                                                                                                                                                          |
| <code>innodb_use_global_flush_log_at_trx_commit</code> | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_use_mtflush</code>                        | Replaced by the <code>innodb_page_cleaners</code> system variable.                                                                                       |
| <code>innodb_use_stacktrace</code>                     | Used in XtraDB-only                                                                                                                                      |
| <code>innodb_use_trim</code>                           |                                                                                                                                                          |

## Reserved Words

- New `reserved words` : EXCEPT and INTERSECT. These can no longer be used as `identifiers` without being quoted.

## SQL\_MODE=ORACLE

- `MariaDB 10.3` has introduced major new Oracle compatibility features. If you upgrade and are using this setting, please check the [changes carefully](#).

## Functions

- As a result of implementing Table Value Constructors, the `VALUES function` has been renamed to `VALUE()`.
- Functions that used to only return 64-bit now can return 32-bit results ( [MDEV-12619](#) ). This could cause incompatibilities with strongly-typed clients.

## mysqldump

- `mysqldump` in `MariaDB 10.3` includes logic to cater for the `mysql.transaction_registry table`.  
`mysqldump` from an earlier MariaDB release cannot be used on `MariaDB 10.3` and beyond.

## MariaDB Backup and Percona XtraBackup

- `Percona XtraBackup` is not compatible with `MariaDB 10.3` . Installations currently using XtraBackup should upgrade to `MariaDB Backup` before upgrading to `MariaDB 10.3` .

## Privileges

- If a user has the `SUPER privilege` but not the `DELETE HISTORY` privilege, running `mysql_upgrade` will grant `DELETE HISTORY` as well.

## Major New Features To Consider

You might consider using the following major new features in `MariaDB 10.3` :

- `System-versioned tables`
- `Sequences`
- See also [System Variables Added in MariaDB 10.3](#) .

## See Also

- The features in MariaDB 10.3
- Upgrading from MariaDB 10.2 to MariaDB 10.3 with Galera Cluster
- Upgrading from MariaDB 10.1 to MariaDB 10.2
- Upgrading from MariaDB 10.0 to MariaDB 10.1

## 2.1.3.8 Upgrading from MariaDB 10.1 to MariaDB 10.2

### Contents

1. [How to Upgrade](#)
2. [Incompatible Changes Between 10.1 and 10.2](#)
  1. [InnoDB Instead of XtraDB](#)
  2. [Options That Have Changed Default Values](#)
  3. [Options That Have Been Removed or Renamed](#)
  4. [Reserved Words](#)
  5. [Tokudb](#)
  6. [Replication](#)
  7. [SQL Mode](#)
  8. [Auto\\_increment](#)
  9. [TLS](#)
3. [Major New Features To Consider](#)
4. [See Also](#)

### How to Upgrade

For Windows, see [Upgrading MariaDB on Windows instead](#).

For MariaDB Galera Cluster, see [Upgrading from MariaDB 10.1 to MariaDB 10.2 with Galera Cluster instead](#).

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

The suggested upgrade procedure is:

1. Modify the repository configuration, so the system's package manager installs [MariaDB 10.2](#). For example,
  - On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
2. Set

```
innodb_fast_shutdown
to
0
. It can be changed dynamically with
```

```
SET GLOBAL
```

```
. For example:
```

```
SET GLOBAL innodb_fast_shutdown=0;
```

- This step is not necessary when upgrading to [MariaDB 10.2.5](#) or later. Omitting it can make the upgrade process far faster. See [MDEV-12289](#) for more information.

3. [Stop MariaDB](#).

4. Uninstall the old version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following:

```
sudo apt-get remove mariadb-server
```

- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:

```
sudo yum remove MariaDB-server
```

- On SLES, OpenSUSE, and other similar Linux distributions, execute the following:

```
sudo zypper remove MariaDB-server
```

5. Install the new version of MariaDB.

- On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.

6. Make any desired changes to configuration options in [option files](#), such as

```
my.cnf
```

. This includes removing any options that are no longer supported.

7. [Start MariaDB](#).

8. Run

```
mysql_upgrade
```

◦

```
mysql_upgrade  
does two things:
```

1. Ensures that the system tables in the

```
mysq
```

```
1
```

database are fully compatible with the new version.

2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

## Incompatible Changes Between 10.1 and 10.2

On most servers upgrading from 10.1 should be painless. However, there are some things that have changed which could affect an upgrade:

### InnoDB Instead of XtraDB

MariaDB 10.2 uses [InnoDB](#) as the default storage engine, rather than XtraDB, used in [MariaDB 10.1](#) and before. See [Why does MariaDB 10.2 use InnoDB instead of XtraDB?](#) In most cases this should have minimal effect as the latest InnoDB has incorporated most of the improvements made in earlier versions of XtraDB. Note that certain [XtraDB system variables](#) are now ignored (although they still exist so as to permit easy upgrading).

### Options That Have Changed Default Values

In particular, take note of the changes to [innodb\\_strict\\_mode](#), [sql\\_mode](#), [binlog\\_format](#), [binlog\\_checksum](#) and [innodb\\_checksum\\_algorithm](#).

| Option                                              | Old default value | New default value |
|-----------------------------------------------------|-------------------|-------------------|
| <a href="#">aria_recover(_options)</a>              | NORMAL            | BACKUP, QUICK     |
| <a href="#">binlog_annotation_row_events</a>        | OFF               | ON                |
| <a href="#">binlog_checksum</a>                     | NONE              | CRC32             |
| <a href="#">binlog_format</a>                       | STATEMENT         | MIXED             |
| <a href="#">group_concat_max_len</a>                | 1024              | 1048576           |
| <a href="#">innodb_autoinc_lock_mode</a>            | 1                 | 2                 |
| <a href="#">innodb_buffer_pool_dump_at_shutdown</a> | OFF               | ON                |
| <a href="#">innodb_buffer_pool_dump_pct</a>         | 100               | 25                |
| <a href="#">innodb_buffer_pool_instances</a>        | 8                 | Varies            |
| <a href="#">innodb_buffer_pool_load_at_startup</a>  | OFF               | ON                |
| <a href="#">innodb_checksum_algorithm</a>           | innodb            | crc32             |
| <a href="#">innodb_file_format</a>                  | Antelope          | Barracuda         |
| <a href="#">innodb_large_prefix</a>                 | OFF               | ON                |
| <a href="#">innodb_lock_schedule_algorithm</a>      | VATS              | FCFS              |
| <a href="#">innodb_log_compressed_pages</a>         | OFF               | ON                |
| <a href="#">innodb_max_dirty_pages_pct_lwm</a>      | 0.001000          | 0                 |

|                                 |                                                   |                                                                                                 |
|---------------------------------|---------------------------------------------------|-------------------------------------------------------------------------------------------------|
| innodb_max_undo_log_size        | 1073741824                                        | 10485760                                                                                        |
| innodb_purge_threads            | 1                                                 | 4                                                                                               |
| innodb_strict_mode              | OFF                                               | ON                                                                                              |
| innodb_undo_directory           | .                                                 | NULL                                                                                            |
| innodb_use_atomic_writes        | OFF                                               | ON                                                                                              |
| innodb_use_trim                 | OFF                                               | ON                                                                                              |
| lock_wait_timeout               | 31536000                                          | 86400                                                                                           |
| log_slow_admin_statements       | OFF                                               | ON                                                                                              |
| log_slow_slave_statements       | OFF                                               | ON                                                                                              |
| log_warnings                    | 1                                                 | 2                                                                                               |
| max_allowed_packet              | 4M                                                | 16M                                                                                             |
| max_long_data_size              | 4M                                                | 16M                                                                                             |
| myisam_recover_options          | NORMAL                                            | BACKUP, QUICK                                                                                   |
| optimizer_switch                | See <a href="#">Optimizer Switch</a> for details. |                                                                                                 |
| replicate_annotation_row_events | OFF                                               | ON                                                                                              |
| server_id                       | 0                                                 | 1                                                                                               |
| slave_net_timeout               | 3600                                              | 60                                                                                              |
| sql_mode                        | NO_AUTO_CREATE_USER,<br>NO_ENGINE_SUBSTITUTION    | STRICT_TRANS_TABLES, ERROR_FOR_DIVISION_BY_ZERO,<br>NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION |
| thread_cache_size               | 0                                                 | Auto                                                                                            |
| thread_pool_max_threads         | 1000                                              | 65536                                                                                           |
| thread_stack                    | 295936                                            | 299008                                                                                          |

## Options That Have Been Removed or Renamed

The following options should be removed or renamed if you use them in your [option files](#) :

| Option                          | Reason                                                                                            |
|---------------------------------|---------------------------------------------------------------------------------------------------|
| aria_recover                    | Renamed to <a href="#">aria_recover_options</a> to match <a href="#">myisam_recover_options</a> . |
| innodb_additional_mem_pool_size | Deprecated in <a href="#">MariaDB 10.0</a> .                                                      |
| innodb_api_bk_commit_interval   | Memcache never implemented in MariaDB.                                                            |
| innodb_api_disable_rowlock      | Memcache never implemented in MariaDB.                                                            |
| innodb_api_enable_binlog        | Memcache never implemented in MariaDB.                                                            |
| innodb_api_enable_mdl           | Memcache never implemented in MariaDB.                                                            |
| innodb_api_trx_level            | Memcache never implemented in MariaDB.                                                            |
| innodb_use_sys_malloc           | Deprecated in <a href="#">MariaDB 10.0</a> .                                                      |

## Reserved Words

New [reserved words](#) : OVER, RECURSIVE and ROWS. These can no longer be used as [identifiers](#) without being quoted.

## TokuDB

[TokuDB](#) has been split into a separate package, mariadb-plugin-tokudb.

## Replication

[Replication](#) from legacy MySQL servers may require setting [binlog\\_checksum](#) to NONE.

## SQL Mode

[SQL\\_MODE](#) has been changed; in particular, NOT NULL fields with no default will no longer fall back to a dummy value for inserts which do not specify a value for that field.

## Auto\_increment

[Auto\\_increment](#) columns are no longer permitted in [CHECK constraints](#), [DEFAULT value expressions](#) and [virtual columns](#). They were permitted in earlier versions, but did not work correctly.

## TLS

Starting with [MariaDB 10.2](#), when the user specifies the

--ssl

option with a [client or utility](#), the [client or utility](#) will not verify the server certificate by default. In order to verify the server certificate, the user must specify the

--ssl-verify-server-cert

option to the [client or utility](#). For more information, see the [list of options](#) for the

`mysql`

client.

## Major New Features To Consider

You might consider using the following major new features in [MariaDB 10.2](#):

- [Window Functions](#)
- [mysqlbinlog](#) now supports continuous binary log backups
- [Recursive Common Table Expressions](#)
- [JSON functions](#)
- See also [System Variables Added in MariaDB 10.2](#).

## See Also

- [The features in MariaDB 10.2](#)
- [Upgrading from MariaDB 10.1 to MariaDB 10.2 with Galera Cluster](#)
- [Upgrading from MariaDB 10.0 to MariaDB 10.1](#)
- [Upgrading from MariaDB 5.5 to MariaDB 10.0](#)

## 2.1.3.9 Upgrading MariaDB on Windows

### Contents

1. [Minor upgrades](#)
2. [General information on upgrade and version coexistence](#)
3. [General recommendations](#)
4. [Upgrade Wizard](#)
5. [mysql\\_upgrade\\_service](#)
6. [Migration to 64 bit MariaDB from 32 bit](#)
7. [Upgrading ZIP-based installations.](#)

## Minor upgrades

To install a minor upgrade, e.g 10.1.27 on top of existing 10.1.26, with MSI, just download the 10.1.27 MSI and start it. It will do everything that needs to be done for minor upgrade automatically - shutdown MariaDB service(s), replace executables and DLLs, and start service(s) again.

The rest of the article is dedicated to \*major\* upgrades, e.g 10.1.x to 10.2.y.

## General information on upgrade and version coexistence

This section assumes MSI installations.

First, check everything listed in the Incompatibilities section of the article relating to the version you are upgrading, for example, [Upgrading from MariaDB 10.1 to MariaDB 10.2](#), to make sure you are prepared for the upgrade.

MariaDB (and also MySQL) allows different versions of the product to co-exist on the same machine, as long as these versions are different either in major or minor version numbers. For example, it is possible to have say [MariaDB 5.1.51](#) and 5.2.6 to be installed on the same machine.

However only a single instance of 5.2 can exist. If for example 5.2.7 is installed on a machine where 5.2.6 is already installed, the installer will just replace 5.2.6 executables with 5.2.7 ones.

Now imagine, that both 5.1 and 5.2 are installed on the same machine and we want to upgrade the database instance running on 5.1 to the new version. In this case special tools are required. Traditionally,

`mysql_upgrade`

is used to accomplish this. On Windows, the [MySQL upgrade](#) is a complicated multiple-step manual process.

Since [MariaDB 5.2.6](#), the Windows distribution includes tools that simplify migration between different versions and also allow migration between MySQL and MariaDB.

**Note**. Automatic upgrades are only possible for DB instances that run as a Windows service.

## General recommendations

**Important:** Ignore any statement that tells you to "*just uninstall MySQL and install MariaDB*". This does not work on Windows, never has, and never will. Keep your MySQL installed until after the database had been converted.

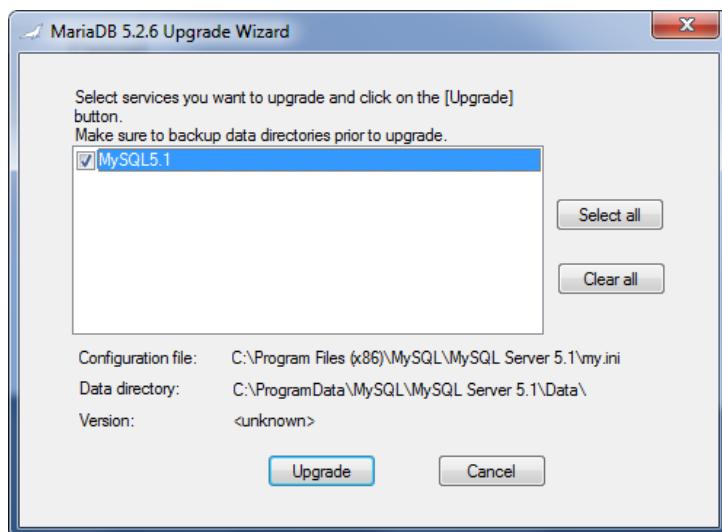
The following install/upgrade sequence is recommended in case of "major" upgrades, like going from 5.3 to 5.5

- Install new version, while still retaining the old one
- Upgrade services one by one, like described later in the document (e.g with `mysql_upgrade_service`). It is recommended to have services cleanly shut down before the upgrade.
- Uninstall old version when previous step is done.

**Note**. This recommendation differs from the procedure on Unixes, where the upgrade sequence is "uninstall old version, install new version"

## Upgrade Wizard

This is a GUI tool that is typically invoked at the end of a MariaDB installation if upgradable services are found. The UI allows you to select instances you want to upgrade.



## mysql\_upgrade\_service

This is a command line tool that performs upgrades. The tool requires full administrative privileges (it has to start and stop services).

Example usage:

```
mysql_upgrade_service --service=MySQL
```

`mysql_upgrade_service` accepts a single parameter — the name of the MySQL or MariaDB service. It performs all the steps to convert a MariaDB/MySQL instance running as the service to the current version.

## Migration to 64 bit MariaDB from 32 bit

Earlier we said that only single instance of "MariaDB <major>.<minor>" version can be installed on the same machine. This was almost correct, because MariaDB MSI installations allow 32 and 64-bit versions to be installed on the same machine, and in this case it is possible to have two instances of say 5.2 installed at the same time, an x86 one and an x64 one. One can use the x64 Upgrade wizard to upgrade an instance running as a 32-bit process to run as 64-bit.

## Upgrading ZIP-based installations.

Both UpgradeWizard and mysql\_upgrade\_service can also be used to upgrade database instances that were installed with the [ZIP installation](#).

## 2.1.3.10 Upgrading Galera Cluster

### 2.1.3.10.1 Upgrading Between Minor Versions with Galera Cluster

#### Performing a Rolling Upgrade

The following steps can be used to perform a rolling upgrade between minor versions of MariaDB (for example from [MariaDB 10.3.12](#) to [MariaDB 10.3.13](#)) when Galera Cluster is being used. In a rolling upgrade, each node is upgraded individually, so the cluster is always operational. There is no downtime from the application's perspective.

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

For each node, perform the following steps:

1. [Stop MariaDB](#).
2. Install the new version of MariaDB and the Galera wsrep provider.
  - On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.
3. Make any desired changes to configuration options in [option files](#), such as  
`my.cnf`  
. This includes removing any system variables or options that are no longer supported.
4. [Start MariaDB](#).
5. Run

`mysql_upgrade`

with the  
`--skip-write-binlog`  
option.

◦

`mysql_upgrade`  
does two things:  
1. Ensures that the system tables in the

`mysq`

1  
database are fully compatible with the new version.

2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB.

When this process is done for one node, move onto the next node.

Note that when upgrading the Galera wsrep provider, sometimes the Galera protocol version can change. The Galera wsrep provider should not start using the new protocol version until all cluster nodes have been upgraded to the new version, so this is not generally an issue during a rolling upgrade. However, this can cause issues if you restart a non-upgraded node in a cluster where the rest of the nodes have been upgraded.

### 2.1.3.10.2 Upgrading from MariaDB 10.3 to MariaDB 10.4 with Galera Cluster

MariaDB starting with [10.1](#)

Since [MariaDB 10.1](#), the [MySQL-wsrep](#) patch has been merged into MariaDB Server. Therefore, in [MariaDB 10.1](#) and above, the functionality of MariaDB Galera Cluster can be obtained by installing the standard MariaDB Server packages and the Galera wsrep provider library package.

Beginning in [MariaDB 10.1](#), [Galera Cluster](#) ships with the MariaDB Server. Upgrading a Galera Cluster node is very similar to upgrading a server from [MariaDB 10.3](#) to [MariaDB 10.4](#). For more information on that process as well as incompatibilities between versions, see the [Upgrade Guide](#).

#### Performing a Rolling Upgrade

The following steps can be used to perform a rolling upgrade from [MariaDB 10.3](#) to [MariaDB 10.4](#) when using Galera Cluster. In a rolling upgrade, each

node is upgraded individually, so the cluster is always operational. There is no downtime from the application's perspective.

First, before you get started:

1. First, take a look at [Upgrading from MariaDB 10.3 to MariaDB 10.4](#) to see what has changed between the major versions.
  1. Check whether any system variables or options have been changed or removed. Make sure that your server's configuration is compatible with the new MariaDB version before upgrading.
  2. Check whether replication has changed in the new MariaDB version in any way that could cause issues while the cluster contains upgraded and non-upgraded nodes.
  3. Check whether any new features have been added to the new MariaDB version. If a new feature in the new MariaDB version cannot be replicated to the old MariaDB version, then do not use that feature until all cluster nodes have been upgraded to the new MariaDB version.
2. Next, make sure that the Galera version numbers are compatible.
  1. If you are upgrading from the most recent [MariaDB 10.3](#) release to [MariaDB 10.4](#), then the versions will be compatible. [MariaDB 10.3](#) uses Galera 3 (i.e. Galera wsrep provider versions 25.3.x), and [MariaDB 10.4](#) uses Galera 4 (i.e. Galera wsrep provider versions 26.4.x). This means that upgrading to [MariaDB 10.4](#) also upgrades the system to Galera 4. However, Galera 3 and Galera 4 should be compatible for the purposes of a rolling upgrade, as long as you are using Galera 26.4.2 or later.
  2. See [What is MariaDB Galera Cluster?: Galera wsrep provider Versions](#) for information on which MariaDB releases uses which Galera wsrep provider versions.
3. Ideally, you want to have a large enough gcache to avoid a [State Snapshot Transfer \(SST\)](#) during the rolling upgrade. The gcache size can be configured by setting

```
gcache.size
```

For example:

```
wsrep_provider_options="gcache.size=2G"
```

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

Then, for each node, perform the following steps:

1. Modify the repository configuration, so the system's package manager installs [MariaDB 10.4](#). For example,
  - On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
2. If you use a load balancing proxy such as MaxScale or HAProxy, make sure to drain the server from the pool so it does not receive any new connections.
3. [Stop MariaDB](#).
4. Uninstall the old version of MariaDB and the Galera wsrep provider.
  - On Debian, Ubuntu, and other similar Linux distributions, execute the following:

```
sudo apt-get remove mariadb-server galera
```
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:

```
sudo yum remove MariaDB-server galera
```
  - On SLES, OpenSUSE, and other similar Linux distributions, execute the following:

```
sudo zypper remove MariaDB-server galera
```

5. Install the new version of MariaDB and the Galera wsrep provider.
  - On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.

6. Make any desired changes to configuration options in [option files](#), such as

```
my.cnf
. This includes removing any system variables or options that are no longer supported.
```

7. On Linux distributions that use

```
systemd
you may need to increase the service startup timeout as the default timeout of 90 seconds may not be sufficient. See Systemd: Configuring the Systemd Service Timeout for more information.
```

8. [Start MariaDB](#).

9. Run

```
mysql_upgrade
```

with the

```
--skip-write-binlog  
option.  
◦  
mysql_upgrade  
does two things:  
1. Ensures that the system tables in the
```

```
mysql
```

```
1  
database are fully compatible with the new version.
```

```
2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .
```

When this process is done for one node, move onto the next node.

Note that when upgrading the Galera wsrep provider, sometimes the Galera protocol version can change. The Galera wsrep provider should not start using the new protocol version until all cluster nodes have been upgraded to the new version, so this is not generally an issue during a rolling upgrade. However, this can cause issues if you restart a non-upgraded node in a cluster where the rest of the nodes have been upgraded.

## 2.1.3.10.3 Upgrading from MariaDB 10.2 to MariaDB 10.3 with Galera Cluster

MariaDB starting with [10.1](#)

Since [MariaDB 10.1](#), the MySQL-wsrep patch has been merged into MariaDB Server. Therefore, in [MariaDB 10.1](#) and above, the functionality of MariaDB Galera Cluster can be obtained by installing the standard MariaDB Server packages and the Galera wsrep provider library package.

Beginning in [MariaDB 10.1](#), Galera Cluster ships with the MariaDB Server. Upgrading a Galera Cluster node is very similar to upgrading a server from [MariaDB 10.2](#) to [MariaDB 10.3](#). For more information on that process as well as incompatibilities between versions, see the [Upgrade Guide](#).

## Performing a Rolling Upgrade

The following steps can be used to perform a rolling upgrade from [MariaDB 10.2](#) to [MariaDB 10.3](#) when using Galera Cluster. In a rolling upgrade, each node is upgraded individually, so the cluster is always operational. There is no downtime from the application's perspective.

First, before you get started:

1. First, take a look at [Upgrading from MariaDB 10.2 to MariaDB 10.3](#) to see what has changed between the major versions.
  1. Check whether any system variables or options have been changed or removed. Make sure that your server's configuration is compatible with the new MariaDB version before upgrading.
  2. Check whether replication has changed in the new MariaDB version in any way that could cause issues while the cluster contains upgraded and non-upgraded nodes.
  3. Check whether any new features have been added to the new MariaDB version. If a new feature in the new MariaDB version cannot be replicated to the old MariaDB version, then do not use that feature until all cluster nodes have been upgraded to the new MariaDB version.
2. Next, make sure that the Galera version numbers are compatible.
  1. If you are upgrading from the most recent [MariaDB 10.2](#) release to [MariaDB 10.3](#), then the versions will be compatible. Both [MariaDB 10.2](#) and [MariaDB 10.3](#) use Galera 3 (i.e. Galera wsrep provider versions 25.3.x), so they should be compatible.
  2. See [What is MariaDB Galera Cluster?: Galera wsrep provider Versions](#) for information on which MariaDB releases uses which Galera wsrep provider versions.
3. Ideally, you want to have a large enough gcache to avoid a [State Snapshot Transfer \(SST\)](#) during the rolling upgrade. The gcache size can be configured by setting

```
gcache.size
```

For example:

```
wsrep_provider_options="gcache.size=2G"
```

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#).

Then, for each node, perform the following steps:

1. Modify the repository configuration, so the system's package manager installs [MariaDB 10.3](#). For example,
  - On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.

- On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
- If you use a load balancing proxy such as MaxScale or HAProxy, make sure to drain the server from the pool so it does not receive any new connections.
  - [Stop MariaDB](#).
  - Uninstall the old version of MariaDB and the Galera wsrep provider.
    - On Debian, Ubuntu, and other similar Linux distributions, execute the following:

```
sudo apt-get remove mariadb-server galera
```

- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:

```
sudo yum remove MariaDB-server galera
```

- On SLES, OpenSUSE, and other similar Linux distributions, execute the following:

```
sudo zypper remove MariaDB-server galera
```

- Install the new version of MariaDB and the Galera wsrep provider.

- On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.

- Make any desired changes to configuration options in [option files](#), such as

`my.cnf`

. This includes removing any system variables or options that are no longer supported.

- On Linux distributions that use

`systemd`

you may need to increase the service startup timeout as the default timeout of 90 seconds may not be sufficient. See [Systemd: Configuring the Systemd Service Timeout](#) for more information.

- [Start MariaDB](#).

- Run

`mysql_upgrade`

with the

`--skip-write-binlog`  
option.

- 

`mysql_upgrade`  
does two things:

- Ensures that the system tables in the

`mysq`

`l`

database are fully compatible with the new version.

- Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

When this process is done for one node, move onto the next node.

Note that when upgrading the Galera wsrep provider, sometimes the Galera protocol version can change. The Galera wsrep provider should not start using the new protocol version until all cluster nodes have been upgraded to the new version, so this is not generally an issue during a rolling upgrade. However, this can cause issues if you restart a non-upgraded node in a cluster where the rest of the nodes have been upgraded.

## 2.1.3.10.4 Upgrading from MariaDB 10.1 to MariaDB 10.2 with Galera Cluster

MariaDB starting with [10.1](#)

Since [MariaDB 10.1](#), the [MySQL-wsrep](#) patch has been merged into MariaDB Server. Therefore, in [MariaDB 10.1](#) and above, the functionality of MariaDB Galera Cluster can be obtained by installing the standard MariaDB Server packages and the Galera wsrep provider library package.

## Contents

1. Performing a Rolling Upgrade
  1. Additional details
    1. Checking Status of the State Transfer
    1. State Transfers that Provide Access to the Server
    2. State Transfers that Require the Server to be Down

Beginning in MariaDB 10.1 , Galera Cluster ships with the MariaDB Server. Upgrading a Galera Cluster node is very similar to upgrading a server from MariaDB 10.1 to MariaDB 10.2 . For more information on that process as well as incompatibilities between versions, see the [Upgrade Guide](#) .

## Performing a Rolling Upgrade

The following steps can be used to perform a rolling upgrade from MariaDB 10.1 to MariaDB 10.2 when using Galera Cluster. In a rolling upgrade, each node is upgraded individually, so the cluster is always operational. There is no downtime from the application's perspective.

First, before you get started:

1. First, take a look at [Upgrading from MariaDB 10.1 to MariaDB 10.2](#) to see what has changed between the major versions.
  1. Check whether any system variables or options have been changed or removed. Make sure that your server's configuration is compatible with the new MariaDB version before upgrading.
  2. Check whether replication has changed in the new MariaDB version in any way that could cause issues while the cluster contains upgraded and non-upgraded nodes.
  3. Check whether any new features have been added to the new MariaDB version. If a new feature in the new MariaDB version cannot be replicated to the old MariaDB version, then do not use that feature until all cluster nodes have been upgraded to the new MariaDB version.
2. Next, make sure that the Galera version numbers are compatible.
  1. If you are upgrading from the most recent MariaDB 10.1 release to MariaDB 10.2 , then the versions will be compatible. Both MariaDB 10.1 and MariaDB 10.2 use Galera 3 (i.e. Galera wsrep provider versions 25.3.x), so they should be compatible.
  2. See [What is MariaDB Galera Cluster?: Galera wsrep provider Versions](#) for information on which MariaDB releases uses which Galera wsrep provider versions.
3. Ideally, you want to have a large enough gcache to avoid a [State Snapshot Transfer \(SST\)](#) during the rolling upgrade. The gcache size can be configured by setting

`gcache.size`

For example:

```
wsrep_provider_options="gcache.size=2G"
```

Before you upgrade, it would be best to take a backup of your database. This is always a good idea to do before an upgrade. We would recommend [Mariabackup](#) .

Then, for each node, perform the following steps:

1. Modify the repository configuration, so the system's package manager installs MariaDB 10.2 . For example,
  - On Debian, Ubuntu, and other similar Linux distributions, see [Updating the MariaDB APT repository to a New Major Release](#) for more information.
  - On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Updating the MariaDB YUM repository to a New Major Release](#) for more information.
  - On SLES, OpenSUSE, and other similar Linux distributions, see [Updating the MariaDB ZYpp repository to a New Major Release](#) for more information.
2. If you use a load balancing proxy such as MaxScale or HAProxy, make sure to drain the server from the pool so it does not receive any new connections.
3. Set

`innodb_fast_shutdown`

```
to  
0  
. It can be changed dynamically with
```

`SET GLOBAL`

. For example:

```
SET GLOBAL innodb_fast_shutdown=0;
```

◦ This step is not necessary when upgrading to MariaDB 10.2.5 or later. Omitting it can make the upgrade process far faster. See [MDEV-1025](#).

[12289](#) for more information.

4. [Stop MariaDB](#) .

5. Uninstall the old version of MariaDB and the Galera wsrep provider.

- On Debian, Ubuntu, and other similar Linux distributions, execute the following:

```
sudo apt-get remove mariadb-server galera
```

- On RHEL, CentOS, Fedora, and other similar Linux distributions, execute the following:

```
sudo yum remove MariaDB-server galera
```

- On SLES, OpenSUSE, and other similar Linux distributions, execute the following:

```
sudo zypper remove MariaDB-server galera
```

6. Install the new version of MariaDB and the Galera wsrep provider.

- On Debian, Ubuntu, and other similar Linux distributions, see [Installing MariaDB Packages with APT](#) for more information.
- On RHEL, CentOS, Fedora, and other similar Linux distributions, see [Installing MariaDB Packages with YUM](#) for more information.
- On SLES, OpenSUSE, and other similar Linux distributions, see [Installing MariaDB Packages with ZYpp](#) for more information.

7. Make any desired changes to configuration options in [option files](#) , such as

`my.cnf`

. This includes removing any system variables or options that are no longer supported.

- In order to use Galera Cluster without problems in [MariaDB 10.2](#) , the

`innodb_lock_schedule_algorithm`

system variable must be set to

`FCFS`

. In [MariaDB 10.2.12](#) and later, this system variable is automatically set to this value when Galera Cluster is enabled. In [MariaDB 10.2.11](#) and before, this system variable must be set to this value manually. See [MDEV-12837](#) for more information.

8. On Linux distributions that use

`systemd`

you may need to increase the service startup timeout as the default timeout of 90 seconds may not be sufficient. See [Systemd: Configuring the Systemd Service Timeout](#) for more information.

9. [Start MariaDB](#) .

10. Run

`mysql_upgrade`

with the

`--skip-write-binlog`  
option.

- 

`mysql_upgrade`  
does two things:

1. Ensures that the system tables in the

`mysq`

`1`

database are fully compatible with the new version.

2. Does a very quick check of all tables and marks them as compatible with the new version of MariaDB .

When this process is done for one node, move onto the next node.

Note that when upgrading the Galera wsrep provider, sometimes the Galera protocol version can change. The Galera wsrep provider should not start using the new protocol version until all cluster nodes have been upgraded to the new version, so this is not generally an issue during a rolling upgrade. However, this can cause issues if you restart a non-upgraded node in a cluster where the rest of the nodes have been upgraded.

## Additional details

### Checking Status of the State Transfer

When a node rejoins the cluster after being upgraded, it may have to perform a state transfer, such as an [Incremental State Transfer \(IST\)](#) or a [State Snapshot Transfer\(SST\)](#) . It is recommended to ensure that the node's state transfer is complete before upgrading the next node in the cluster.

### State Transfers that Provide Access to the Server

If the node is synchronizing with the cluster by performing a state transfer that allows access to the server, such as an [Incremental State Transfer \(IST\)](#) or a [State Snapshot Transfer\(SST\)](#) that uses the

```
mysqldump
```

SST method, then you can check the status of the state transfer by connecting to the server through the

```
mysql
```

client, then checking the

```
wsrep_local_state_uuid
```

and

```
wsrep_cluster_state_uuid
```

status variables. When they equal each other, the node is in sync with the cluster.

```
select if((SELECT VARIABLE_VALUE AS "uuid"
FROM information_schema.GLOBAL_STATUS
WHERE VARIABLE_NAME = "wsrep_cluster_state_uuid")=(SELECT VARIABLE_VALUE AS "uuid"
FROM information_schema.GLOBAL_STATUS
WHERE VARIABLE_NAME = "wsrep_local_state_uuid"), "Synced", "Not Synced") as "Cluster Status";
+-----+
| Cluster Status |
+-----+
| Synced         |
+-----+
```

When the local and cluster UUID's come into sync, the node is again online and functioning as a part of the cluster.

#### State Transfers that Require the Server to be Down

Some state transfers require the server to be unavailable, such as all [State Snapshot Transfer\(SST\)](#) methods other than

```
mysqldump
```

, so

```
mysql
```

client access is unavailable while the state transfer is happening. In those cases, you may have to monitor the progress of the state transfer in the [error log](#). You'll know when the SST is complete when the joiner node changes its state to

```
SYNCED
```

. For example:

```
2018-08-30 14:44:15 140694729484032 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 210248566)
```

## 2.1.3.11 Upgrading from MySQL to MariaDB

### 2.1.3.11.1 Upgrading from MySQL to MariaDB

#### Contents

1. [Upgrading on Windows](#)
2. [Upgrading my.cnf](#)
3. [Other Things to Think About](#)
4. [See Also](#)

For [all practical purposes](#), you can view MariaDB as an upgrade of MySQL:

- Before upgrading, please [check if there are any known incompatibilities](#) between your MySQL release and the MariaDB release you want to move to.
- In particular, note that the [JSON type](#) in MariaDB is a LONGTEXT, while in MySQL it's a binary type. See [Making MariaDB understand MySQL JSON](#).
- If you are using MySQL 8.0 or above, you have to use [mysqldump](#) to move your database to MariaDB.
- For upgrading from very old MySQL versions, see [Upgrading to MariaDB from MySQL 5.0 \(or older version\)](#).
- Within the same base version (for example MySQL 5.5 -> [MariaDB 5.5](#), MySQL 5.6 -> [MariaDB 10.0](#) and MySQL 5.7 -> [MariaDB 10.2](#)) you can in most cases just uninstall MySQL and install MariaDB and you are good to go. There is no need to dump and restore databases. As with any upgrade, we recommend making a backup of your data beforehand.
- You should run

```
mysql_upgrade
```

(just as you would with MySQL) to finish the upgrade. This is needed to ensure that your mysql privilege and event tables are updated with the new fields MariaDB uses. Note that if you use a MariaDB package,

```
mysql_upgrade
```

is usually run automatically.

- All your old clients and connectors (PHP, Perl, Python, Java, etc.) will work unchanged (no need to recompile). This works because MariaDB and MySQL use the same client protocol and the client libraries are binary compatible. You can also use your old MySQL connector packages with MariaDB if you want.

## Upgrading on Windows

On Windows, you should not uninstall MySQL and install MariaDB, this would not work, the existing database will not be found.

Thus On Windows, just install MariaDB and use the upgrade wizard which is part of installer package and is launched by MSI installer. Or, in case you prefer command line, use

```
mysql_upgrade_service <service_name>
on the command line.
```

## Upgrading my.cnf

All the options in your original MySQL

```
my.cnf
file should work fine for MariaDB.
```

However as MariaDB has more features than MySQL, there is a few things that you should consider changing in your

```
my.cnf
file.
```

- MariaDB uses by default the [Aria storage engine](#) for internal temporary files instead of MyISAM. If you have a lot of temporary files, you should add and set

```
aria-pagecache-buffer-size
```

to the same value as you have for

```
key-buffer-size
```

- If you don't use MyISAM tables, you can set

```
key-buffer-size
```

to a very low value, like 64K.

- If using [MariaDB 10.1](#) or earlier, and your applications often connect and disconnect to MariaDB, you should set up

```
thread-cache-size
```

to the number of concurrent queries threads you are typically running. This is important in MariaDB as we are using the [jemalloc](#) memory allocator. [jemalloc](#) usually has better performance when running many threads compared to other memory allocators, except if you create and destroy a lot of threads, in which case it will spend a lot of resources trying to manage thread specific storage. Having a thread cache will fix this problem.

- If you have a LOT of connections (> 100) that mostly run short running queries, you should consider using the [thread pool](#). For example using :

```
thread_handling=pool-of-threads
```

and

```
thread_pool_size=128
```

could give a notable performance boost in this case. Where the

```
thread_pool_size
```

should be about

```
2 * number of cores on your machine
```

## Other Things to Think About

- Views with definition

```
ALGORITHM=MERGE
```

or

```
ALGORITHM=TEMPTABLE
```

got accidentally swapped between MariaDB and MySQL. You have to re-create views created with either of these definitions (see [MDEV-6916](#)).

- MariaDB has LGPL versions of the [C connector](#) and [Java Client](#). If you are shipping an application that supports MariaDB or MySQL, you should consider using these!

- You should consider trying out the [MyRocks storage engine](#) or some of the other [new storage engines](#) that MariaDB provides.

## See Also

- MariaDB has a lot of [new features](#) that you should know about.
- [MariaDB versus MySQL - Compatibility](#)
- [Migrating to MariaDB](#)
- You can find general upgrading informations on the [MariaDB installation page](#).
- There is a [Screencast for upgrading MySQL to MariaDB](#).
- [Upgrading to MariaDB in Debian 9](#)

## 2.1.3.11.2 Moving from MySQL to MariaDB in Debian 9

MariaDB 10.1 is now the default mysql server in Debian 9 "Stretch". This page provides information on this change and instructions to help with upgrading your Debian 8 "Jessie" version of MySQL or MariaDB to [MariaDB 10.1](#) in Debian 9 "Stretch".

### Contents

1. [Background information](#)
2. [Before you upgrade](#)
  1. [Backup before you begin](#)
  2. [Changed, renamed, and removed options](#)
    1. [Options with changed default values](#)
    2. [Options that have been removed or renamed](#)
  3. [Suggested upgrade procedure for replication](#)
  4. [Other resources to consult before beginning your upgrade](#)
3. [Upgrading to MariaDB 10.1 from MySQL 5.5](#)
4. [Upgrading to MariaDB 10.1 from an older version of MariaDB](#)
5. [MariaDB Galera Cluster](#)
6. [Configuration options for advanced database users](#)
7. [Secure passwordless root accounts only on new installs](#)
8. [See also](#)
9. [Comments and suggestions](#)
10. [Notes](#)

## Background information

The version of MySQL in Debian 8 "Jessie" is 5.5. When installing, most users will install the

```
mysql-server
package, which depends on the
mysql-server-5.5 package
. In Debian 9 "Stretch" the
mysql-server
package depends on a new package called
default-mysql-server
. This package in turn depends on
mariadb-server-10.1
. There is no
default-mysql-server
package in Jessie.
```

In both Jessie and Stretch there is also a

```
mariadb-server
package which is a MariaDB-specific analog to the
mysql-server
package. In Jessie this package depends on
mariadb-server-10.0
and in Stretch this package depends on
mariadb-server-10.1
(the same as the
default-mysql-server
package).
```

So, the main repository difference in Debian 9 "Stretch" is that when you install the

```
mysql-server
package on Stretch you will get MariaDB 10.1 instead of MySQL, like you would with previous versions of Debian. Note that
```

```
mysql-server  
is just an empty transitional meta-package and users are encouraged to install MariaDB using the actual package  
mariadb-server
```

All apps and tools, such as the popular LAMP stack, in the repositories that depend on the mysql-server package will continue to work using MariaDB as the database. For new installs there is nothing different that needs to be done when installing the mysql-server or mariadb-server packages.

## Before you upgrade

If you are currently running MySQL 5.5 on Debian 8 "Jessie" and are planning an upgrade to [MariaDB 10.1](#) on Debian 9 "Stretch", there are some things to keep in mind:

### Backup before you begin

This is a major upgrade, and so complete database backups are strongly suggested before you begin. [MariaDB 10.1](#) is compatible on disk and wire with MySQL 5.5, and the MariaDB developer team has done extensive development and testing to make upgrades as painless and trouble-free as possible. Even so, it's always a good idea to do regular backups, especially before an upgrade. As the database has to shutdown anyway for the upgrade, this is a good opportunity to do a backup!

### Changed, renamed, and removed options

Some default values have been changed, some have been renamed, and others have been removed between MySQL 5.5 and [MariaDB 10.1](#). The following sections detail them.

#### Options with changed default values

Most of the following options have increased a bit in value to give better performance. They should not use much additional memory, but some of them do use a bit more disk space.

| Option                                   | Old default value | New default value |
|------------------------------------------|-------------------|-------------------|
| <code>aria-sort-buffer-size</code>       | 128M              | 256M              |
| <code>back_log</code>                    | 50                | 150               |
| <code>innodb-concurrency-tickets</code>  | 500               | 5000              |
| <code>innodb-log-file-size</code>        | 5M                | 48M               |
| <code>innodb_log_compressed_pages</code> | ON                | OFF               |
| <code>innodb-old-blocks-time</code>      | 0                 | 1000              |

|                         |      |                                            |
|-------------------------|------|--------------------------------------------|
| innodb-open-files       | 300  | 400<br>[2]                                 |
| innodb-purge-batch-size | 20   | 300                                        |
| innodb-undo-logs        | ON   | 20                                         |
| join_buffer_size        | 128K | 256K                                       |
| max_allowed_packet      | 1M   | 4M                                         |
| max-connect-errors      | 10   | 100                                        |
| max-relay-log-size      | 0    | 1024M                                      |
| myisam-sort-buffer-size | 8M   | 128M                                       |
| optimizer-switch        | ...  | Added<br>extended_keys=on, exists_to_in=on |
| query_alloc_block_size  | 8192 | 16384                                      |
| query_cache_size        | 0    | 1M                                         |
| query_cache_type        | ON   | OFF                                        |

|                                      |       |                                                              |
|--------------------------------------|-------|--------------------------------------------------------------|
| <code>query_prealloc_size</code>     | 8192  | 24576                                                        |
| <code>secure_auth</code>             | OFF   | ON                                                           |
| <code>sql_log_bin</code>             |       | No longer affects replication of events in a Galera cluster. |
| <code>sql_mode</code>                | empty | <code>NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION</code>     |
| <code>sync_master_info</code>        | 0     | 10000                                                        |
| <code>sync_relay_log</code>          | 0     | 10000                                                        |
| <code>sync_relay_log_info</code>     | 0     | 10000                                                        |
| <code>table_open_cache</code>        | 400   | 2000                                                         |
| <code>thread_pool_max_threads</code> | 500   | 1000                                                         |

### Options that have been removed or renamed

The following options should be removed or renamed if you use them in your config files:

| Option                                       | Reason                                                                                |
|----------------------------------------------|---------------------------------------------------------------------------------------|
| <code>engine-condition-pushdown</code>       | Replaced with<br><br><code>set optimizer_switch='engine_condition_pushdown=on'</code> |
| <code>innodb-adaptive-flushing-method</code> | Removed by <a href="#">XtraDB</a>                                                     |
| <code>innodb-autoextend-increment</code>     | Removed by <a href="#">XtraDB</a>                                                     |

|                                       |                                                      |
|---------------------------------------|------------------------------------------------------|
| innodb-blocking-buffer-pool-restore   | Removed by <a href="#">XtraDB</a>                    |
| innodb-buffer-pool-pages              | Removed by <a href="#">XtraDB</a>                    |
| innodb-buffer-pool-pages-blob         | Removed by <a href="#">XtraDB</a>                    |
| innodb-buffer-pool-pages-index        | Removed by <a href="#">XtraDB</a>                    |
| innodb-buffer-pool-restore-at-startup | Removed by <a href="#">XtraDB</a>                    |
| innodb-buffer-pool-shm-checksum       | Removed by <a href="#">XtraDB</a>                    |
| innodb-buffer-pool-shm-key            | Removed by <a href="#">XtraDB</a>                    |
| innodb-checkpoint-age-target          | Removed by <a href="#">XtraDB</a>                    |
| innodb-dict-size-limit                | Removed by <a href="#">XtraDB</a>                    |
| innodb-doublewrite-file               | Removed by <a href="#">XtraDB</a>                    |
| innodb-fast-checksum                  | Renamed to <a href="#">innodb-checksum-algorithm</a> |
| innodb-flush-neighbor-pages           | Renamed to <a href="#">innodb-flush-neighbors</a>    |
| innodb-ibuf-accel-rate                | Removed by <a href="#">XtraDB</a>                    |
| innodb-ibuf-active-contract           | Removed by <a href="#">XtraDB</a>                    |
| innodb-ibuf-max-size                  | Removed by <a href="#">XtraDB</a>                    |
| innodb-import-table-from-xtrabackup   | Removed by <a href="#">XtraDB</a>                    |
| innodb-index-stats                    | Removed by <a href="#">XtraDB</a>                    |
| innodb-lazy-drop-table                | Removed by <a href="#">XtraDB</a>                    |
| innodb-merge-sort-block-size          | Removed by <a href="#">XtraDB</a>                    |

|                                       |                                        |
|---------------------------------------|----------------------------------------|
| innodb-persistent-stats-root-page     | Removed by <a href="#">XtraDB</a>      |
| innodb-read-ahead                     | Removed by <a href="#">XtraDB</a>      |
| innodb-recovery-stats                 | Removed by <a href="#">XtraDB</a>      |
| innodb-recovery-update-relay-log      | Removed by <a href="#">XtraDB</a>      |
| innodb-stats-auto-update              | Renamed to<br>innodb-stats-auto-recalc |
| innodb-stats-update-need-lock         | Removed by <a href="#">XtraDB</a>      |
| innodb-sys-stats                      | Removed by <a href="#">XtraDB</a>      |
| innodb-table-stats                    | Removed by <a href="#">XtraDB</a>      |
| innodb-thread-concurrency-timer-based | Removed by <a href="#">XtraDB</a>      |
| innodb-use-sys-stats-table            | Removed by <a href="#">XtraDB</a>      |
| rpl_recovery_rank                     | Unused in 10.0+                        |
| xtradb-admin-command                  | Removed by <a href="#">XtraDB</a>      |

## Suggested upgrade procedure for replication

If you have a [master-slave setup](#), the normal procedure is to first upgrade your slaves to MariaDB, then move one of your slaves to be the master and then upgrade your original master. In this scenario you can upgrade from MySQL to MariaDB or upgrade later to a new version of MariaDB without any downtime.

## Other resources to consult before beginning your upgrade

It may also be useful to check out the [Upgrading MariaDB](#) section. It contains several articles on upgrading from MySQL to MariaDB and from one version of MariaDB to another. For upgrade purposes, MySQL 5.5 and [MariaDB 5.5](#) are very similar. In particular, see the [Upgrading from MariaDB 5.5 to MariaDB 10.0](#) and [Upgrading from MariaDB 10.0 to MariaDB 10.1](#) articles.

If you need help with upgrading or setting up replication, you can always [contact the MariaDB corporation](#) to find experts to help you with this.

## Upgrading to MariaDB 10.1 from MySQL 5.5

The suggested upgrade procedure is:

1. Set `innodb_fast_shutdown` to
  - 0
  - . This is to ensure that if you make a backup as part of the upgrade, all data is written to the InnoDB data files, which simplifies any restore in the future.
2. Shutdown MySQL 5.5
3. Take a [backup](#)
  - when the server is shut down is the perfect time to take a backup of your databases

- store a copy of the backup on external media or a different machine for safety
4. Perform the upgrade from Debian 8 to Debian 9
  5. During the upgrade, the [mysql\\_upgrade](#) script will be run automatically; this script does two things:
    1. Upgrades the permission tables in the
 

```
mysql
database with some new fields
```
    2. Does a very quick check of all tables and marks them as compatible with [MariaDB 10.1](#)
      - In most cases this should be a fast operation (depending of course on the number of tables)
  6. Add new options to [my.cnf](#) to enable features
    - If you change
 

```
my.cnf
then you need to restart
mysqld
with e.g.
sudo service mysql restart
or
sudo service mariadb restart
```
    -

## Upgrading to [MariaDB 10.1](#) from an older version of MariaDB

If you have installed [MariaDB 5.5](#) or [MariaDB 10.0](#) on your Debian 8 "Jessie" machine from the MariaDB repositories you will need to upgrade to [MariaDB 10.1](#) when upgrading to Debian 9 "Stretch". You can choose to continue using the MariaDB repositories or move to using the Debian repositories.

If you want to continue using the MariaDB repositories edit the MariaDB entry in your sources.list and change every instance of 5.5 or 10.0 to 10.1. Then upgrade as suggested [above](#).

If you want to move to using [MariaDB 10.1](#) from the Debian repositories, delete or comment out the MariaDB entries in your sources.list file. Then upgrade as suggested [above](#).

If you are already using [MariaDB 10.1](#) on your Debian 8 "Jessie" machine, you can choose to continue to use the MariaDB repositories or move to using the Debian repositories as with [MariaDB 5.5](#) and 10.0. In either case, the upgrade will at most be just a minor upgrade from one version of [MariaDB 10.1](#) to a newer version. In the case that you are already on the current version of MariaDB that exists in the Debian repositories or a newer one) MariaDB will not be upgraded during the system upgrade but will be upgraded when future versions of MariaDB are released.

You should always perform a compete backup of your data prior to performing any major system upgrade, even if MariaDB itself is not being upgraded!

## MariaDB Galera Cluster

If you have been using MariaDB Galera Cluster 5.5 or 10.0 on Debian 8 "Jessie" it is worth mentioning that [Galera Cluster](#) is included by default in [MariaDB 10.1](#), there is no longer a need to install a separate

```
mariadb-galera-server
package.
```

## Configuration options for advanced database users

To get better performance from MariaDB used in production environments, here are some suggested additions to [your configuration file](#) which in Debian is at

```
/etc/mysql/mariadb.d/my.cnf
:
```

```
[[mysqld]]
# Cache for disk based temporary files
aria_pagecache_buffer_size=128M
# If you are not using MyISAM tables directly (most people are using InnoDB)
key_buffer_size=64K
```

The reason for the above change is that MariaDB is using the newer [Aria](#) storage engine for disk based temporary files instead of MyISAM. The main benefit of Aria is that it can cache both indexes and rows and thus gives better performance than MyISAM for large queries.

## Secure passwordless root accounts only on new installs

Unlike the old MySQL packages in Debian, [MariaDB 10.0](#) onwards in Debian uses unix socket authentication on new installs to avoid root password management issues and thus be more secure and easier to use with provision systems of the cloud age.

This only affects new installs. Upgrades from old versions will continue to use whatever authentication and user accounts already existed. This is however good to know, because it can affect upgrades of dependant systems, typically e.g. require users to rewrite their Ansible scripts and similar tasks. The new feature is much easier than the old, so adjusting for it requires little work.

## See also

- Differences in MariaDB in Debian (and Ubuntu)
- Configuring MariaDB for optimal performance
- New features in MariaDB you should consider using
- What is MariaDB 10.1
- General instructions for upgrading from MySQL to MariaDB

## Comments and suggestions

If you have comments or suggestions on things we can add or change to improve this page. Please add them as comments below.

## Notes

1. ↑ The

```
innodb-open-files  
variable defaults to the value of  
table-open-cache  
(  
400  
is the default) if it is set to any value less than  
10  
so long as  
innodb-file-per-table  
is set to  
1  
or  
TRUE  
(the default). If  
innodb_file_per_table  
is set to  
0  
or  
FALSE  
and  
innodb-open-files  
is set to a value less than  
10  
, the default is  
300
```

### 2.1.3.11.3 Screencast for Upgrading MySQL to MariaDB

There is a [screencast](#) for upgrading from MySQL 5.1.55 to MariaDB. Watch this example to see how easy this process is. It really is just a "drop in replacement" to MySQL.

## 2.1.4 Downgrading between Major Versions of MariaDB

Downgrading MariaDB is not supported. The only reliable way to downgrade is to [restore from a full backup](#) made before upgrading, and start the old version of MariaDB.

Some people have reported successfully downgrading, but there are many possible things that can go wrong, and downgrading is not tested in any way by the MariaDB developers.

Between major releases, there are often substantial changes, even if none of the new features are used. For example, both [MariaDB 10.2](#) and [MariaDB 10.3](#) introduce new versions of the redo log.

Even within minor releases, there may be problems, for example [MariaDB 10.1.21](#) fixed a file format incompatibility bug that prevents a downgrade to earlier [MariaDB 10.1](#) releases.

## 2.1.5 Compiling MariaDB From Source

### 2.1.6

#### 2.1.6.1 Starting and Stopping MariaDB

## Contents

- 1. Service Managers
  - 1. [Systemd](#)
  - 2. [SysVinit](#)
  - 3. [launchd](#)
  - 4. [Upstart](#)
- 2. Starting the Server Process Manually
  - 1. [mysqld](#)
  - 2. [mysqld\\_safe](#)
  - 3. [mysqld\\_multi](#)
  - 4. [mysql.server](#)

There are several different methods to start or stop the MariaDB Server process. There are two primary categories that most of these methods fall into: starting the process with the help of a service manager, and starting the process manually.

## Service Managers

[sysVinit](#) and [systemd](#) are the most common Linux service managers. [launchd](#) is used in MacOS X. [Upstart](#) is a less common service manager.

### Systemd

RHEL/CentOS 7 and above, Debian 8 Jessie and above, and Ubuntu 15.04 and above use [systemd](#) by default.

For information on how to start and stop MariaDB with this service manager, see [systemd: Interacting with the MariaDB Server Process](#).

### SysVinit

RHEL/CentOS 6 and below, and Debian 7 Wheezy and below use [sysVinit](#) by default.

For information on how to start and stop MariaDB with this service manager, see [sysVinit: Interacting with the MariaDB Server Process](#).

### launchd

[launchd](#) is used in MacOS X.

### Upstart

Ubuntu 14.10 and below use Upstart by default.

## Starting the Server Process Manually

### mysqld

[mysqld](#) is the actual MariaDB Server binary. It can be started manually on its own.

### mysqld\_safe

[mysqld\\_safe](#) is a wrapper that can be used to start the [mysqld](#) server process. The script has some built-in safeguards, such as automatically restarting the server process if it dies. See [mysqld\\_safe](#) for more information.

### mysqld\_multi

[mysqld\\_multi](#) is a wrapper that can be used to start the [mysqld](#) server process if you plan to run multiple server processes on the same host. See [mysqld\\_multi](#) for more information.

### mysql.server

[mysql.server](#) is a wrapper that works as a standard [sysVinit](#) script. However, it can be used independently of [sysVinit](#) as a regular `sh` script. The script starts the [mysqld](#) server process by first changing its current working directory to the MariaDB install directory and then starting [mysqld\\_safe](#). The script requires the standard [sysVinit](#) arguments, such as  
`start`  
`,`  
`stop`  
`, and`  
`status`  
. See [mysql.server](#) for more information.

## 2.1.6.2 Configuring MariaDB with Option Files

### Contents

1. [Global Options Related to Option Files](#)
2. [Default Option File Locations](#)
  1. Default Option File Locations on Linux, Unix, Mac
  2. Default Option File Locations on Windows
  3. Default Option File Hierarchy
3. [Custom Option File Locations](#)
4. [Option File Syntax](#)
5. [Option Groups](#)
  1. Server Option Groups
  2. Client Option Groups
  3. Tool-Specific Option Groups
  4. Custom Option Group Suffixes
6. [Including Option Files](#)
7. [Including Option File Directories](#)
8. [Checking Program Options](#)
9. [MySQL 5.6 Obfuscated Authentication Credential Option File](#)
10. [Option Prefixes](#)
11. [Options](#)
  1. MariaDB Server Options
  2. MariaDB Client Options
12. [Example Option Files](#)
  1. [Example Minimal Option File](#)
  2. [Example Hybrid Option File](#)
13. [See Also](#)

You can configure MariaDB to run the way you want by configuring the server with MariaDB's option files. The default MariaDB option file is called

`my.cnf`

on Unix-like operating systems and

`my.ini`

on Windows. Depending on how you've [installed](#) MariaDB, the default option file may be in a number of places, or it may not exist at all.

## Global Options Related to Option Files

The following options relate to how MariaDB handles option files. These options can be used with most of MariaDB's command-line tools, not just `mysqld`. They must be given as the first argument on the command-line:

| Option                                                   | Description                                                                         |
|----------------------------------------------------------|-------------------------------------------------------------------------------------|
| <code>--print-defaults</code>                            | Read options from option files, print all option values, and then exit the program. |
| <code>--no-defaults</code>                               | Don't read options from any option file.                                            |
| <code>--defaults-file</code><br><code>=path</code>       | Only read options from the given option file.                                       |
| <code>--defaults-extra-file</code><br><code>=path</code> | Read this extra option file after all other option files are read.                  |

|                                            |                                                                                                 |
|--------------------------------------------|-------------------------------------------------------------------------------------------------|
| <pre>--defaults-group-suffix =suffix</pre> | <p>In addition to the default option groups, also read option groups with the given suffix.</p> |
|--------------------------------------------|-------------------------------------------------------------------------------------------------|

## Default Option File Locations

MariaDB reads option files from many different directories by default. See the sections below to find out which directories are checked for which system.

For an exact list of option files read on your system by a specific program, you can execute:

```
$program --help --verbose
```

For example:

```
$ mysqld --help --verbose
mysqld Ver 10.3.13-MariaDB-log for Linux on x86_64 (MariaDB Server)
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Starts the MariaDB database server.

Usage: mysqld [OPTIONS]

Default options are read from the following files in the given order:
/etc/my.cnf ~/.my.cnf
The following groups are read: mysqld server mysqld-10.3 mariadb mariadb-10.3 client-server galera
....
```

The option files are each scanned once, in the order given by

- `--help` --verbose
- . The effect of the configuration options are as if they would have been given as command line options in the order they are found.

## Default Option File Locations on Linux, Unix, Mac

On Linux, Unix, or Mac OS X, the default option file is called

- `my.cnf`
- . MariaDB looks for the MariaDB option file in the locations and orders listed below.

The locations are dependent on whether the

`DEFAULT_SYSCONFDIR`

`cmake`

option was defined when MariaDB was built. This option is usually defined as  
`/etc`  
when building [RPM packages](#), but it is usually not defined when building [DEB packages](#) or [binary tarballs](#).

- When the

`DEFAULT_SYSCONFDIR`

`cmake`

option was **not** defined, MariaDB looks for the MariaDB option file in the following locations in the following order:

| Location                           | Scope  |
|------------------------------------|--------|
| <code>/etc/my.cnf</code>           | Global |
| <code>/etc/mysql/my.cnf</code>     | Global |
| <code>\$MARIADB_HOME/my.cnf</code> | Server |

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| \$MYSQL_HOME/my.cnf | Server                                                   |
| defaults-extra-file | File specified with<br>--defaults-extra-file<br>, if any |
| ~/.my.cnf           | User                                                     |

- When the

DEFAULT\_SYSConFDIR

cmake

option was defined, MariaDB looks for the MariaDB option file in the following locations in the following order:

| Location                  | Scope                                                    |
|---------------------------|----------------------------------------------------------|
| DEFAULT_SYSConFDIR/my.cnf | Global                                                   |
| \$MARIADB_HOME/my.cnf     | Server (from <a href="#">MariaDB 10.6</a> )              |
| \$MYSQL_HOME/my.cnf       | Server                                                   |
| defaults-extra-file       | File specified with<br>--defaults-extra-file<br>, if any |
| ~/.my.cnf                 | User                                                     |

- MARIADB\_HOME  
(from [MariaDB 10.6](#)) or  
MYSQL\_HOME  
is the [environment variable](#) containing the path to the directory holding the server-specific  
my.cnf  
file. If  
MYSQL\_HOME  
is not set, and the server is started with [mysqld\\_safe](#) ,  
MYSQL\_HOME  
is set as follows:
  - If there is a  
my.cnf  
file in the MariaDB data directory, but not in the MariaDB base directory,  
MYSQL\_HOME  
is set to the MariaDB data directory.
  - Else,  
MYSQL\_HOME  
is set to the MariaDB base directory.
- Note that if  
MARIADB\_HOME  
is set (from [MariaDB 10.6](#) ),  
MYSQL\_HOME  
will not be used, even if set.

## Default Option File Locations on Windows

On Windows, the option file can be called either

my.ini

or

my.cnf

. MariaDB looks for the MariaDB option file in the following locations in the following order:

| Location                        | Scope                                                            |
|---------------------------------|------------------------------------------------------------------|
| System Windows Directory\my.ini | Global                                                           |
| System Windows Directory\my.cnf | Global                                                           |
| Windows Directory\my.ini        | Global                                                           |
| Windows Directory\my.cnf        | Global                                                           |
| C:\my.ini                       | Global                                                           |
| C:\my.cnf                       | Global                                                           |
| INSTALLDIR\my.ini               | Server                                                           |
| INSTALLDIR\my.cnf               | Server                                                           |
| INSTALLDIR\data\my.ini          | Server                                                           |
| INSTALLDIR\data\my.cnf          | Server                                                           |
| %MARIADB_HOME%\my.ini           | Server (from <a href="#">MariaDB 10.6</a> )                      |
| %MARIADB_HOME%\my.cnf           | Server (from <a href="#">MariaDB 10.6</a> )                      |
| %MYSQL_HOME%\my.ini             | Server                                                           |
| %MYSQL_HOME%\my.cnf             | Server                                                           |
| defaults-extra-file             | File specified with<br><br>--defaults-extra-file<br><br>, if any |

- The

System Windows Directory  
is the directory returned by the

[GetSystemWindowsDirectory](#)

function. The value is usually

C:\Windows

. To find its specific value on your system, open

`cmd.exe`

and execute:

```
echo %WINDIR%
```

- The

Windows Directory  
is the directory returned by the

`GetWindowsDirectory`

function. The value may be a private  
Windows Directory  
for the application, or it may be the same as the  
System Windows Directory  
returned by the

`GetSystemWindowsDirectory`

function.

- `INSTALLDIR`  
is the parent directory of the directory where  
`mysqld.exe`  
is located. For example, if  
`mysqld.exe`  
is in  
`C:\Program Files\`

`MariaDB 10.3`

`\bin`  
, then  
`INSTALLDIR`  
would be  
`C:\Program Files\`

`MariaDB 10.3`

- `MARIADB_HOME`  
(from [MariaDB 10.6](#)) or  
`MYSQL_HOME`  
is the [environment variable](#) containing the path to the directory holding the server-specific  
`my.cnf`  
file.

- Note that if

`MARIADB_HOME`  
is set (from [MariaDB 10.6](#)),  
`MYSQL_HOME`  
will not be used, even if set.

## Default Option File Hierarchy

MariaDB will look in all of the above locations, in order, even if it has already found an option file, and it's possible for more than one option file to exist. For example, you could have an option file in

`/etc/my.cnf`  
with global settings for all servers, and then you could another option file in  
`~/.my.cnf`  
(i.e. your user account's home directory) which will specify additional settings (or override previously specified setting) that are specific only to that user.

Option files are usually optional. However, if the

`--defaults-file`

option is set, and if the file does not exist, then MariaDB will raise an error. If the

`--defaults-file`

option is set, then MariaDB will *only* read the option file referred to by this option.

If an option or system variable is not explicitly set, then it will be set to its default value. See [Server System Variables](#) for a full list of all server system variables and their default values.

## Custom Option File Locations

MariaDB can be configured to read options from custom options files with the following command-line arguments. These command-line arguments can be used with most of MariaDB's command-line tools, not just

`mysqld`

. They must be given as the first argument on the command-line:

| Option                                                   | Description                                                        |
|----------------------------------------------------------|--------------------------------------------------------------------|
| <code>--defaults-file</code><br><code>=path</code>       | Only read options from the given option file.                      |
| <code>--defaults-extra-file</code><br><code>=path</code> | Read this extra option file after all other option files are read. |

## Option File Syntax

The syntax of the MariaDB option files are:

- Lines starting with # are comments.
- Empty lines are ignored.
- Option groups use the syntax
  - [group-name]
  - . See the [Option Groups](#) section below for more information on available option groups.
- The same option group can appear multiple times.
- The
  - `!include`  
directive can be used to include other option files. See the [Including Option Files](#) section below for more information on this syntax.
  - The
    - `!includedir`  
directive can be used to include all  
.cnf  
files (and potentially  
.ini  
files) in a given directory. The option files within the directory are read in alphabetical order. See the [Including Option File Directories](#) section below for more information on this syntax.
- Dashes (-)
  - 
  - ) and underscores (\_)
  - 
  - ) in options are interchangeable.
- Double quotes can be used to quote values
- - `\n`
  - ,
  - `\r`
  - ,
  - `\t`
  - ,
  - `\b`
  - ,
  - `\s`
  - ,
  - `\"`
  - ,

\',  
, and  
\\\\'  
are recognized as character escapes for new line, carriage return, tab, backspace, space, double quote, single quote, and backslash respectively.

- Certain option prefixes are supported. See the [Option Prefixes](#) section below for information about available option prefixes.
- See the [Options](#) section below for information about available options.

## Option Groups

A MariaDB program can read options from one or many option groups. For an exact list of option groups read on your system by a specific program, you can execute:

```
$program --help --verbose
```

For example:

```
$ mysqld --help --verbose
mysqld Ver 10.3.13-MariaDB-log for Linux on x86_64 (MariaDB Server)
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Starts the MariaDB database server.

Usage: mysqld [OPTIONS]

Default options are read from the following files in the given order:
/etc/my.cnf ~/.my.cnf
The following groups are read: mysqld server mysqld-10.3 mariadb mariadb-10.3 client-server galera
...
```

## Server Option Groups

MariaDB programs reads server options from the following server option groups:

| Group           | Description                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [client-server] | Options read by all MariaDB <a href="#">client programs</a> and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients. |
| [server]        | Options read by MariaDB Server.                                                                                                                                                          |
| [mysqld]        | Options read by<br>mysqld<br>, which includes both MariaDB Server and MySQL Server.                                                                                                      |
| [mysqld-X.Y]    | Options read by a specific version of<br>mysqld<br>, which includes both MariaDB Server and MySQL Server. For example,<br>[mysqld-10.4]<br>.                                             |
| [mariadb]       | Options read by MariaDB Server.                                                                                                                                                          |
| [mariadb-X.Y]   | Options read by a specific version of MariaDB Server. For example,<br>[mariadb-10.4]<br>.                                                                                                |
| [mariadb-ndb]   | Options read by MariaDB Server. Available starting with <a href="#">MariaDB 10.4.6</a> .                                                                                                 |

|               |                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [mariadb-X.Y] | Options read by a specific version of MariaDB Server. For example,<br>[mariadb-10.4]<br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                                                                                                                                                                                                                                |
| [galera]      | Options read by MariaDB Server, but only if it is compiled with <a href="#">Galera Cluster</a> support. In <a href="#">MariaDB 10.1</a> and later, all builds on Linux are compiled with <a href="#">Galera Cluster</a> support. When using one of these builds, options from this option group are read even if the <a href="#">Galera Cluster</a> functionality is not enabled. |

X.Y in the examples above refer to the base (major.minor) version of the server. For example, [MariaDB 10.3.10](#) would read from

[mariadb-10.3]  
. By using the  
mariadb-X.Y

syntax, one can create option files that have MariaDB-only options in the MariaDB-specific option groups. That would allow the option file to work for both MariaDB and MySQL.

## Client Option Groups

MariaDB programs reads client options from the following option groups:

| Group            | Description                                                                                                                                                                              |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [client]         | Options read by all MariaDB and MySQL <a href="#">client programs</a> , which includes both MariaDB and MySQL clients. For example,<br>mysqldump                                         |
| [client-server]  | Options read by all MariaDB <a href="#">client programs</a> and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients. |
| [client-mariadb] | Options read by all MariaDB <a href="#">client programs</a> .                                                                                                                            |

## Tool-Specific Option Groups

Many MariaDB tools reads options from their own option groups as well. Many of them are listed below:

| Group          | Description                                                                                                                                 |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| [mysqld_safe]  | Options read by<br><a href="#">mysqld_safe</a><br>, which includes both MariaDB Server and MySQL Server.                                    |
| [safe_mysqld]  | Options read by<br><a href="#">mysqld_safe</a><br>, which includes both MariaDB Server and MySQL Server.                                    |
| [mariadb_safe] | Options read by<br><a href="#">mysqld_safe</a><br>from MariaDB Server.                                                                      |
| [mariadb-safe] | Options read by<br><a href="#">mysqld_safe</a><br>from MariaDB Server. Available starting with <a href="#">MariaDB 10.4.6</a> .             |
| [mariabackup]  | Options read by <a href="#">Mariabackup</a> . Available starting with <a href="#">MariaDB 10.1.31</a> and <a href="#">MariaDB 10.2.13</a> . |
| [xtrabackup]   | Options read by <a href="#">Mariabackup</a> and <a href="#">Percona XtraBackup</a> .                                                        |

|                   |                                                                                                                        |
|-------------------|------------------------------------------------------------------------------------------------------------------------|
| [mysql_upgrade]   | Options read by<br><br>mysql_upgrade<br><br>, which includes both MariaDB Server and MySQL Server.                     |
| [mariadb-upgrade] | Options read by<br><br>mysql_upgrade<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                 |
| [sst]             | Specific options read by the <a href="#">mariabackup SST method</a> and the <a href="#">xtrabackup-v2 SST method</a> . |
| [mysql]           | Options read by<br><br>mysql<br><br>, which includes both MariaDB Server and MySQL Server.                             |
| [mariadb-client]  | Options read by<br><br>mysql<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                         |
| [mysqldump]       | Options read by<br><br>mysqldump<br><br>, which includes both MariaDB Server and MySQL Server.                         |
| [mariadb-dump]    | Options read by<br><br>mysqldump<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                     |
| [mysqlimport]     | Options read by<br><br>mysqlimport<br><br>, which includes both MariaDB Server and MySQL Server.                       |
| [mariadb-import]  | Options read by<br><br>mysqlimport<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                   |
| [mysqlbinlog]     | Options read by<br><br>mysqlbinlog<br><br>, which includes both MariaDB Server and MySQL Server.                       |
| [mariadb-binlog]  | Options read by<br><br>mysqlbinlog<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                   |
| [mysqladmin]      | Options read by<br><br>mysqladmin<br><br>, which includes both MariaDB Server and MySQL Server.                        |
| [mariadb-admin]   | Options read by<br><br>mysqladmin<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                    |

|                 |                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------|
| [mysqlshow]     | Options read by<br><br>mysqlshow<br><br>, which includes both MariaDB Server and MySQL Server.                          |
| [mariadb-show]  | Options read by<br><br>mysqlshow<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                      |
| [mysqlcheck]    | Options read by<br><br>mysqlcheck<br><br>, which includes both MariaDB Server and MySQL Server.                         |
| [mariadb-check] | Options read by<br><br>mysqlcheck<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                     |
| [mysqlslap]     | Options read by<br><br>mysqlslap<br><br>, which includes both MariaDB Server and MySQL Server.                          |
| [mariadb-slap]  | Options read by<br><br>mysqlslap<br><br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                      |
| [odbc]          | Options read by <a href="#">MariaDB Connector/ODBC</a> , but only if the<br><br>USE_MCNF<br><br>parameter has been set. |

## Custom Option Group Suffixes

MariaDB can be configured to read options from option groups with a custom suffix by providing the following command-line argument. This command-line argument can be used with most of MariaDB's command-line tools, not just

`mysqld`

. It must be given as the first argument on the command-line:

| Option                                 | Description                                                                              |
|----------------------------------------|------------------------------------------------------------------------------------------|
| --defaults-group-suffix<br><br>=suffix | In addition to the default option groups, also read option groups with the given suffix. |

The default group suffix can also be specified via the

`MYSQL_GROUP_SUFFIX`  
[environment variable](#).

## Including Option Files

It is possible to include additional option files from another option file. For example, to include  
`/etc/mysql/dbserver1.cnf`  
, an option file could contain:

```
[mariadb]
...
!include /etc/mysql/dbserver1.cnf
```

## Including Option File Directories

It is also possible to include all option files in a directory from another option file. For example, to include all option files in

```
/etc/my.cnf.d/
, an option file could contain:
```

```
[mariadb]
...
!includedir /etc/my.cnf.d/
```

The option files within the directory are read in alphabetical order.

All option file names must end in

```
.cnf
on Unix-like operating systems. On Windows, all option file names must end in
.cnf
or
.ini
.
```

## Checking Program Options

You can check which options a given program is going to use by using the

```
--print-defaults
```

command-line argument:

| Option           | Description                                                                         |
|------------------|-------------------------------------------------------------------------------------|
| --print-defaults | Read options from option files, print all option values, and then exit the program. |

This command-line argument can be used with most of MariaDB's command-line tools, not just

```
mysqld
```

. It must be given as the first argument on the command-line. For example:

```
$ mysqldump --print-defaults
mysqldump would have been started with the following arguments:
--ssl_cert=/etc/my.cnf.d/certificates/client-cert.pem --ssl_key=/etc/my.cnf.d/certificates/client-key.pem --ssl_ca=/etc/my.cnf.d/certificates/ca.pem
```

You can also check which options a given program is going to use by using the

```
my_print_defaults
```

utility and providing the names of the option groups that the program reads. For example:

```
$ my_print_defaults mysqldump client client-server client-mariadb
--ssl_cert=/etc/my.cnf.d/certificates/client-cert.pem
--ssl_key=/etc/my.cnf.d/certificates/client-key.pem
--ssl_ca=/etc/my.cnf.d/certificates/ca.pem
--ssl-verify-server-cert
--max_allowed_packet=1GB
```

The

```
my_print_defaults
```

utility's

```
--mysqld  
command-line option provides a shortcut to refer to all of the server option groups :
```

```
$ my_print_defaults --mysqld  
--log_bin=mariadb-bin  
--log_slave_updates=ON  
--ssl_cert=/etc/my.cnf.d/certificates/server-cert.pem  
--ssl_key=/etc/my.cnf.d/certificates/server-key.pem  
--ssl_ca=/etc/my.cnf.d/certificates/ca.pem
```

## MySQL 5.6 Obfuscated Authentication Credential Option File

MySQL 5.6 and above support an obfuscated authentication credential option file called

.mylogin.cnf

that is created with

[mysql\\_config\\_editor](#)

MariaDB does not support this. The passwords in MySQL's

.mylogin.cnf

are only obfuscated, rather than encrypted, so the feature does not really add much from a security perspective. It is more likely to give users a false sense of security, rather than to seriously protect them.

## Option Prefixes

MariaDB supports certain prefixes that can be used with options. The supported option prefixes are:

| Option Prefix                      | Description                                                                                                                     |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <code>auto</code> <code>set</code> | Sets the option value automatically. Only supported for certain options. Available in <a href="#">MariaDB 10.1.7</a> and later. |
| <code>dis</code> <code>able</code> | For all boolean options, disables the setting (equivalent to setting it to<br>0<br>). Same as<br><code>skip</code>              |
| <code>en</code> <code>able</code>  | For all boolean options, enables the setting (equivalent to setting it to<br>1<br>).                                            |
| <code>lo</code> <code>ose</code>   | Don't produce an error if the option doesn't exist.                                                                             |
| <code>ma</code> <code>ximum</code> | Sets the maximum value for the option.                                                                                          |
| <code>sk</code> <code>ip</code>    | For all boolean options, disables the setting (equivalent to setting it to<br>0<br>). Same as<br><code>disable</code>           |

For example:

```
[mariadb]
...
# determine a good value for open_files_limit automatically
auto_set_open_files_limit

# disable the unix socket plugin
disable_unix_socket

# enable the slow query log
enable_slow_query_log

# don't produce an error if these options don't exist
loose_file_key_management_filename = /etc/mysql/encryption/keyfile.enc
loose_file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
loose_file_key_management_encryption_algorithm = AES_CTR

# set max_allowed_packet to maximum value
maximum_max_allowed_packet

# disable external locking for MyISAM
skip_external_locking
```

## Options

Dashes (-

) and underscores (\_

\_ ) in options are interchangeable.

If an option is not explicitly set, then the server or client will simply use the default value for that option.

## MariaDB Server Options

MariaDB Server options can be set in [server option groups](#).

For a list of options that can be set for MariaDB Server, see the list of options available for

`mysqld`

Most of the [server system variables](#) can also be set in MariaDB's option file.

## MariaDB Client Options

MariaDB client options can be set in [client option groups](#).

See the specific page for each [client program](#) to determine what options are available for that program.

## Example Option Files

Most MariaDB installations include a sample MariaDB option file called

`my-default.cnf`

. On older releases, you would have also found the following option files:

- `my-small.cnf`
- `my-medium.cnf`
- `my-large.cnf`
- `my-huge.cnf`

However, these option files are now very dated for modern servers, so they were removed in [MariaDB 10.3.1](#).

In source distributions, the sample option files are usually found in the  
`support-files`

directory, and in other distributions, the option files are usually found in the share/mysql directory that is relative to the MariaDB base installation directory.

You can copy one of these sample MariaDB option files and use it as the basis for building your server's primary MariaDB option file.

## Example Minimal Option File

The following is a minimal my.cnf file that you can use to test MariaDB.

```
[client-server]
# Uncomment these if you want to use a nonstandard connection to MariaDB
#socket=/tmp/mysql.sock
#port=3306

# This will be passed to all MariaDB clients
[client]
#password=my_password

# The MariaDB server
[mysqld]
# Directory where you want to put your data
data=/usr/local/mysql/var
# Directory for the errmsg.sys file in the language you want to use
language=/usr/local/share/mysql/english

# This is the prefix name to be used for all log, error and replication files
log basename=mysqld

# Enable logging by default to help find problems
general-log
log-slow-queries
```

## Example Hybrid Option File

The following is an extract of an option file that one can use if one wants to work with both MySQL and MariaDB.

```

# Example mysql config file.

[client-server]
socket=/tmp/mysql-dbug.sock
port=3307

# This will be passed to all mysql clients
[client]
password=my_password

# Here are entries for some specific programs
# The following values assume you have at least 32M ram

# The MySQL server
[mysqld]
temp-pool
key_buffer_size=16M
datadir=/my/mysqldata
loose-innodb_file_per_table

[mariadb]
datadir=/my/data
default-storage-engine=aria
loose-mutex-deadlock-detector
max-connections=20

[mariadb-5.5]
language=/my/maria-5.5/sql/share/english/
socket=/tmp/mysql-dbug.sock
port=3307

[mariadb-10.1]
language=/my/maria-10.1/sql/share/english/
socket=/tmp/mysql2-dbug.sock

[mysqldump]
quick
max_allowed_packet=16M

[mysql]
no-auto-rehash
loose-abort-source-on-error

```

## See Also

- [Configuring MariaDB Connector/C with Option Files](#)
- [Troubleshooting Connection Issues](#)
- [Configuring MariaDB for Remote Client Access](#)
- [MySQL 5.6: Security through Complacency?](#)

## 2.1.6.3 mysqld Configuration Files and Groups

For all about configuring mysqld, see [Configuring MariaDB with Option Files](#).

## 2.1.6.4 mysqld Options

MariaDB starting with [10.4.6](#)

From [MariaDB 10.4.6](#),

```

mariadb
is a symlink to
mysqld
.
```

MariaDB starting with [10.5.2](#)

From [MariaDB 10.5.2](#),

```

mariadb
is the name of the binary, with
mysqld
a symlink.
```

# Contents

- 1. Option Prefixes
  - 1. `--auto-set-`\*
  - 2. `--disable-`\*
  - 3. `--enable-`\*
  - 4. `--loose-`\*
  - 5. `--maximum-`\*
  - 6. `--skip-`\*
- 2. Option File Options
  - 1. `--defaults-extra-file`
  - 2. `--defaults-file`
  - 3. `--defaults-group-suffix`
  - 4. `--no-defaults`
  - 5. `--print-defaults`
- 3. Compatibility Options
  - 1. `-a, --ansi`
  - 2. `--new`
  - 3. `--old`
  - 4. `--old-alter-table`
  - 5. `--old-mode`
  - 6. `--old-passwords`
  - 7. `--old-style-user-limits`
  - 8. `--safe-mode`
  - 9. `--show-old-temporals`
  - 10. `--skip-new`
- 11. Compatibility Options and System Variables
- 4. Locale Options
  - 1. `--character-set-client-handshake`
  - 2. `--character-set-filesystem`
  - 3. `--character-set-server`
  - 4. `--character-sets-dir`
  - 5. `--collation-server`
  - 6. `--default-character-set`
  - 7. `--default-time-zone`
  - 8. `--default-week-format`
  - 9. `--language`
  - 10. `--lc-messages`
  - 11. `--lc-messages-dir`
  - 12. `--lc-time-names`
- 13. Locale Options and System Variables
- 5. Windows Options
  - 1. `--console`
  - 2. `--named-pipe`
  - 3. `--install`
  - 4. `--install-manual`
  - 5. `--remove`
  - 6. `--slow-start-timeout`
  - 7. `--standalone`
- 8. Windows Options and System Variables
- 6. Replication and Binary Logging Options
  - 1. `--abort-slave-event-count`
  - 2. `--auto-increment-increment`
  - 3. `--auto-increment-offset`
  - 4. `--binlog-annotate-row-events`
  - 5. `--binlog-cache-size`
  - 6. `--binlog-checksum`
  - 7. `--binlog-commit-wait-count`
  - 8. `--binlog-commit-wait-usec`
  - 9. `--binlog-direct-non-transactional-updates`
  - 10. `--binlog-do-db`
  - 11. `--binlog-expire-logs-seconds`
  - 12. `--binlog-file-cache-size`
  - 13. `--binlog-format`
  - 14. `--binlog-ignore-db`
  - 15. `--binlog-optimize-thread-scheduling`
  - 16. `--binlog-row-event-max-size`
  - 17. `--binlog-row-image`
  - 18. `--binlog-row-metadata`
  - 19. `--binlog-stmt-cache-size`

20. `--default-master-connection`  
21. `--disconnect-slave-event-count`  
22. `--flashback`  
23. `--gtid-cleanup-batch-size`  
24. `--gtid-domain-id`  
25. `--gtid-ignore-duplicates`  
26. `--gtid-strict-mode`  
27. `--init-rpl-role`  
28. `--init-slave`  
29. `--log-basename`  
30. `--log-bin`  
31. `--log-bin-compress`  
32. `--log-bin-compress-min-len`  
33. `--log-bin-index`  
34. `--log-bin-trust-function-creators`  
35. `--log-bin-trust-routine-creators`  
36. `--log-slave-updates`  
37. `--master-host`  
38. `--master-info-file`  
39. `--master-password`  
40. `--master-port`  
41. `--master-retry-count`  
42. `--master-ssl`  
43. `--master-ssl-ca`  
44. `--master-ssl-capath`  
45. `--master-ssl-cert`  
46. `--master-ssl-cipher`  
47. `--master-ssl-key`  
48. `--master-user`  
49. `--master-verify-checksum`  
50. `--max-binlog-cache-size`  
51. `--max-binlog-dump-events`  
52. `--max-binlog-size`  
53. `--max-binlog-stmt-cache-size`  
54. `--max-relay-log-size`  
55. `--read-binlog-speed-limit`  
56. `--relay-log`  
57. `--relay-log-index`  
58. `--relay-log-info-file`  
59. `--relay-log-purge`  
60. `--relay-log-recovery`  
61. `--relay-log-space-limit`  
62. `--replicate-annotate-row-events`  
63. `--replicate-do-db`  
64. `--replicate-do-table`  
65. `--replicate-events-marked-for-skip`  
66. `--replicate-ignore-db`  
67. `--replicate-ignore-table`  
68. `--replicate-rewrite-db`  
69. `--replicate-same-server-id`  
70. `--replicate-wild-do-table`  
71. `--replicate-wild-ignore-table`  
72. `--report-host`  
73. `--report-password`  
74. `--report-port`  
75. `--report-user`  
76. `--rpl-recovery-rank`  
77. `--server-id`  
78. `--slave-ddl-exec-mode`  
79. `--slave-compressed-protocol`  
80. `--slave-domain-parallel-threads`  
81. `--slave-exec-mode`  
82. `--slave-load-tmpdir`  
83. `--slave-max-allowed-packet`  
84. `--slave-net-timeout`  
85. `--slave-parallel-threads`  
86. `--slave-parallel-max-queued`  
87. `--slave-run-triggers-for-rbr`  
88. `--slave-skip-errors`  
89. `--slave-sql-verify-checksum`  
90. `--slave-transaction-retries`  
91. `--slave-transaction-retry-errors`  
92. `--slave-transaction-retry-interval`

- 93. `--slave-type-conversions`
- 94. `--sporadic-binlog-dump-fail`
- 95. `--sync-binlog`
- 96. `--sync-master-info`
- 97. `--sync-relay-log`
- 98. `--sync-relay-log-info`
- 99. `--sysdate-is-now`
- 100. Replication and Binary Logging Options and System Variables
- 101. Semisynchronous Replication Options and System Variables
  - 1. `rpl-semi-sync-master-enabled`
  - 2. `rpl-semi-sync-master-timeout`
  - 3. `rpl-semi-sync-master-trace-level`
  - 4. `rpl-semi-sync-master-wait-no-slave`
  - 5. `rpl-semi-sync-master-wait-point`
  - 6. `rpl-semi-sync-slave-delay-master`
  - 7. `rpl-semi-sync-slave-kill-conn-timeout`
  - 8. `rpl-semi-sync-slave-enabled`
  - 9. `rpl-semi-sync-slave-trace-level`
- 7. Optimizer Options
  - 1. `--alter-algorithm`
  - 2. `--analyze-sample-percentage`
  - 3. `--big-tables`
  - 4. `--bulk-insert-buffer-size`
  - 5. `--expensive-subquery-limit`
  - 6. `--join-buffer-size`
  - 7. `--join-buffer-space-limit`
  - 8. `--join-cache-level`
  - 9. `--max-heap-table-size`
  - 10. `--max-join-size`
  - 11. `--max-seeks-for-key`
  - 12. `--max-sort-length`
  - 13. `--mrr-buffer-size`
  - 14. `--optimizer-max-sel-arg-weight`
  - 15. `--optimizer-prune-level`
  - 16. `--optimizer-search-depth`
  - 17. `--optimizer-selectivity-sampling-limit`
  - 18. `--optimizer-switch`
  - 19. `-optimizer-trace`
  - 20. `-optimizer-trace-max-mem-size`
  - 21. `--optimizer-use-condition-selectivity`
  - 22. `--query-alloc-block-size`
  - 23. `--query-prealloc-size`
  - 24. `--range-alloc-block-size`
  - 25. `--read-buffer-size`
  - 26. `--record-buffer`
  - 27. `--rowid-merge-buff-size`
  - 28. `--table-cache`
  - 29. `--table-definition-cache`
  - 30. `--table-open-cache`
  - 31. `--table-open-cache-instances`
  - 32. `--tmp-disk-table-size`
  - 33. `--tmp-memory-table-size`
  - 34. `--tmp-table-size`
  - 35. `--use-stat-tables`
- 36. Optimizer Options and System Variables
- 8. Storage Engine Options
  - 1. `--skip-bdb`
  - 2. MyISAM Storage Engine Options
    - 1. `--concurrent-insert`
    - 2. `--delayed-insert-limit`
    - 3. `--delayed-insert-timeout`
    - 4. `--delayed-queue-size`
    - 5. `--external-locking`
    - 6. `--keep-files-on-create`
    - 7. `--key-buffer-size`
    - 8. `--key-cache-age-threshold`
    - 9. `--key-cache-block-size`
    - 10. `--key-cache-division-limit`
    - 11. `--key-cache-file-back-size`

11. `--key-cache-me-hash-size`
12. `--key-cache-segments`
13. `--log-isam`
14. `--myisam-block-size`
15. `--myisam-data-pointer-size`
16. `--myisam-max-sort-file-size`
17. `--myisam-mmap-size`
18. `--myisam-recover-options`
19. `--myisam-repair-threads`
20. `--myisam-sort-buffer-size`
21. `--myisam-stats-method`
22. `--myisam-use-mmap`
23. [MyISAM Storage Engine Options and System Variables](#)
3. [InnoDB Storage Engine Options](#)
  1. `--ignore-built-in-innodb`
  2. `--innodb`
  3. `--innodb-adaptive-checkpoint`
  4. `--innodb-adaptive-flushing`
  5. `--innodb-adaptive-flushing-lwm`
  6. `--innodb-adaptive-flushing-method`
  7. `--innodb-adaptive-hash-index`
  8. `--innodb-adaptive-hash-index-partitions`
  9. `--innodb-adaptive-hash-index-parts`
  10. `--innodb-adaptive-max-sleep-delay`
  11. `--innodb-additional-mem-pool-size`
  12. `--innodb-api-bk-commit-interval`
  13. `--innodb-api-disable-rowlock`
  14. `--innodb-api-enable-binlog`
  15. `--innodb-api-enable-mdl`
  16. `--innodb-api-trx-level`
  17. `--innodb-auto-lru-dump`
  18. `--innodb-autoextend-increment`
  19. `--innodb-autocomp-lock-mode`
  20. `--innodb-background-scrub-data-check-interval`
  21. `--innodb-background-scrub-data-compressed`
  22. `--innodb-background-scrub-data-interval`
  23. `--innodb-background-scrub-data-uncompressed`
  24. `--innodb-blocking-buffer-pool-restore`
  25. `--innodb-buf-dump-status-frequency`
  26. `--innodb-buffer-pool-chunk-size`
  27. `--innodb-buffer-pool-dump-at-shutdown`
  28. `--innodb-buffer-pool-dump-now`
  29. `--innodb-buffer-pool-dump-pct`
  30. `--innodb-buffer-pool-evict`
  31. `--innodb-buffer-pool-filename`
  32. `--innodb-buffer-pool-instances`
  33. `--innodb-buffer-pool-load-abort`
  34. `--innodb-buffer-pool-load-at-startup`
  35. `--innodb-buffer-pool-load-now`
  36. `--innodb-buffer-pool-load-pages-abort`
  37. `--innodb-buffer-pool-populate`
  38. `--innodb-buffer-pool-restore-at-startup`
  39. `--innodb-buffer-pool-shm-checksum`
  40. `--innodb-buffer-pool-shm-key`
  41. `--innodb-buffer-pool-size`
  42. `--innodb-change-buffer-max-size`
  43. `--innodb-change-buffering`
  44. `--innodb-change-buffering-debug`
  45. `--innodb-checkpoint-age-target`
  46. `--innodb-checksum-algorithm`
  47. `--innodb-checksums`
  48. `--innodb-cleaner-lsn-age-factor`
  49. `--innodb-cmp`

50. --innodb-cmp-per-index-enabled  
51. --innodb-cmp-reset  
52. --innodb-cmpmem  
53. --innodb-cmpmem-reset  
54. --innodb-commit-concurrency  
55. --innodb-compression-algorithm  
56. --innodb-compression-failure-threshold-pct  
57. --innodb-compression-level  
58. --innodb-compression-pad-pct-max  
59. --innodb-concurrency-tickets  
60. --innodb-corrupt-table-action  
61. --innodb-data-file-path  
62. --innodb-data-home-dir  
63. --innodb-deadlock-detect  
64. --innodb-deadlock-report  
65. --innodb-default-encryption-key-id  
66. --innodb-default-page-encryption-key  
67. --innodb-default-row-format  
68. --innodb-defragment  
69. --innodb-defragment-fill-factor  
70. --innodb-defragment-fill-factor-n-recs  
71. --innodb-defragment-frequency  
72. --innodb-defragment-n-pages  
73. --innodb-defragment-stats-accuracy  
74. --innodb-dict-size-limit  
75. --innodb-disable-sort-file-cache  
76. --innodb-doublewrite  
77. --innodb-doublewrite-file  
78. --innodb-empty-free-list-algorithm  
79. --innodb-enable-unsafe-group-commit  
80. --innodb-encrypt-log  
81. --innodb-encrypt-tables  
82. --innodb-encrypt-temporary-tables  
83. --innodb-encryption-rotate-key-age  
84. --innodb-encryption-rotation-iops  
85. --innodb-encryption-threads  
86. --innodb-extra-rsegments  
87. --innodb-extra-undoslots  
88. --innodb-fake-changes  
89. --innodb-fast-checksum  
90. --innodb-fast-shutdown  
91. --innodb-fatal-semaphore-wait-threshold  
92. --innodb-file-format  
93. --innodb-file-format-check  
94. --innodb-file-format-max  
95. --innodb-file-io-threads  
96. --innodb-file-per-table  
97. --innodb-filill-factor  
98. --innodb-flush-log-at-trx-commi  
99. --innodb-flush-method  
100. --innodb-flush-neighbor-pages  
101. --innodb-flush-neighbors  
102. --innodb-flush-sync  
103. --innodb-flushing-avg-loops  
104. --innodb-force-load-corrupted  
105. --innodb-force-primary-key  
106. --innodb-force-recovery  
107. --innodb-foreground-preflush  
108. --innodb-ft-aux-table  
109. --innodb-ft-cache-size  
110. --innodb-ft-enable-diag-print  
111. --innodb-ft-enable-stopword  
112. --innodb-ft-max-token-size  
113. --innodb-ft-min-token-size  
114. --innodb-ft-num-word-optimize  
115. --innodb-ft-result-cache-limit  
116. --innodb-ft-server-stopword-table  
117. --innodb\_ft\_sort\_nll\_degree

117. --innodb-rr-sort-pct-degree  
118. --innodb-ft-total-cache-size  
119. --innodb-ft-user-stopword-table  
120. --innodb-ibuf-accel-rate  
121. --innodb-ibuf-active-contract  
122. --innodb-ibuf-max-size  
123. --innodb-idle-flush-pct  
124. --innodb-immediate-scrub-data-uncompressed  
125. --innodb-import-table-from-xtrabackup  
126. --innodb-index-stats  
127. --innodb-instant-alter-column-allowed  
128. --innodb-instrument-semaphores  
129. --innodb-io-capacity  
130. --innodb-io-capacity-max  
131. --innodb-large-prefix  
132. --innodb-lazy-drop-table  
133. --innodb-lock-schedule-algorithm  
134. --innodb-lock-wait-timeout  
135. --innodb-lock-waits  
136. --innodb-locking-fake-changes  
137. --innodb-locks  
138. --innodb-locks-unsafe-for-binlog  
139. --innodb-log-arch-dir  
140. --innodb-log-arch-expire-sec  
141. --innodb-log-archive  
142. --innodb-log-block-size  
143. --innodb-log-buffer-size  
144. --innodb-log-checksum-algorithm  
145. --innodb-log-checksums  
146. --innodb-log-compressed-pages  
147. --innodb-log-file-size  
148. --innodb-log-files-in-group  
149. --innodb-log-group-home-dir  
150. --innodb-log-optimize-ddl  
151. --innodb-log-write-ahead-size  
152. --innodb-lru-flush-size  
153. --innodb-lru-scan-depth  
154. --innodb-max-bitmap-file-size  
155. --innodb-max-changed-pages  
156. --innodb-max-dirty-pages-pct  
157. --innodb-max-dirty-pages-pct-lwm  
158. --innodb-max-purge-lag  
159. --innodb-max-purge-lag-delay  
160. --innodb-max-purge-lag-wait  
161. --innodb-max-undo-log-size  
162. --innodb-merge-sort-block-size  
163. --innodb-mirrored-log-groups  
164. --innodb-monitor-disable  
165. --innodb-monitor-enable  
166. --innodb-monitor-reset  
167. --innodb-monitor-reset-all  
168. --innodb-mtflush-threads  
169. --innodb-numa-interleave  
170. --innodb-old-blocks-pct  
171. --innodb-old-blocks-time  
172. --innodb-online-alter-log-max-size  
173. --innodb-open-files  
174. --innodb-optimize-fulltext-only  
175. --innodb-page-cleaners  
176. --innodb-page-size  
177. --innodb-pass-corrupt-table  
178. --innodb-prefix-index-cluster-optimization  
179. --innodb-print-all-deadlocks  
180. --innodb-purge-batch-size  
181. --innodb-purge-rseg-truncate-frequency  
182. --innodb-purge-threads  
183. --innodb-random-read-ahead  
184. --innodb-read-ahead

185. `--innodb-read-ahead-threshold`  
186. `--innodb-read-io-threads`  
187. `--innodb-read-only`  
188. `--innodb-recovery-update-relay-log`  
189. `--innodb-replication-delay`  
190. `--innodb-rollback-on-timeout`  
191. `--innodb-rollback-segments`  
192. `--innodb-rseg`  
193. `--innodb-sched-priority-cleaner`  
194. `--innodb-safe-truncate`  
195. `--innodb-scrub-log`  
196. `--innodb-scrub-log-interval`  
197. `--innodb-scrub-log-speed`  
198. `--innodb-show-locks-held`  
199. `--innodb-show-verbose-locks`  
200. `--innodb-sort-buffer-size`  
201. `--innodb-spin-wait-delay`  
202. `--innodb-stats-auto-recalc`  
203. `--innodb-stats-auto-update`  
204. `--innodb-stats-include-delete-marked`  
205. `--innodb-stats-method`  
206. `--innodb-stats-modified-counter`  
207. `--innodb-stats-on-metadata`  
208. `--innodb-stats-persistent`  
209. `--innodb-stats-persistent-sample-pages`  
210. `--innodb-stats-sample-pages`  
211. `--innodb-stats-traditional`  
212. `--innodb-stats-transient-sample-pages`  
213. `--innodb-stats-update-need-lock`  
214. `--innodb-status-file`  
215. `--innodb-status-output`  
216. `--innodb-status-output-locks`  
217. `--innodb-strict-mode`  
218. `--innodb-support-xa`  
219. `--innodb-sync-array-size`  
220. `--innodb-sync-spin-loops`  
221. `--innodb-sys-indexes`  
222. `--innodb-sys-stats`  
223. `--innodb-sys-tables`  
224. `--innodb-table-locks`  
225. `--innodb-table-stats`  
226. `--innodb-temp-data-file-path`  
227. `--innodb-thread-concurrency`  
228. `--innodb-thread-concurrency-timer-based`  
229. `--innodb-thread-sleep-delay`  
230. `--innodb-tmpdir`  
231. `--innodb-track-changed-pages`  
232. `--innodb-track-redo-log-now`  
233. `--innodb-trx`  
234. `--innodb-undo-directory`  
235. `--innodb-undo-log-truncate`  
236. `--innodb-undo-logs`  
237. `--innodb-undo-tablespaces`  
238. `--innodb-use-atomic-writes`  
239. `--innodb-use-fallocate`  
240. `--innodb-use-global-flush-log-at-trx-commit`  
241. `--innodb-use-mtflush`  
242. `--innodb-use-native-aio`  
243. `--innodb-use-purge-thread`  
244. `--innodb-use-stacktrace`  
245. `--innodb-use-sys-malloc`  
246. `--innodb-use-sys-stats-table`  
247. `--innodb-use-trim`  
248. `--innodb-write-io-threads`  
249. `--skip-innodb`  
250. `--skip-innodb-checksums`  
251. `--skip-innodb-doublewrite`  
252. [InnoDB Storage Engine Options and Configuration](#)

- System Variables**
- 4. **Aria Storage Engine Options**
  - 1. `--aria-block-size`
  - 2. `--aria-checkpoint-interval`
  - 3. `--aria-checkpoint-log-activity`
  - 4. `--aria-encrypt-tables`
  - 5. `--aria-force-start-after-recovery-failures`
  - 6. `--aria-group-commit`
  - 7. `--aria-group-commit-interval`
  - 8. `--aria-log-dir-path`
  - 9. `--aria-log-file-size`
  - 10. `--aria-log-purge-type`
  - 11. `--aria-max-sort-file-size`
  - 12. `--aria-page-checksum`
  - 13. `--aria-pagecache-age-threshold`
  - 14. `--aria-pagecache-buffer-size`
  - 15. `--aria-pagecache-division-limit`
  - 16. `--aria-pagecache-file-hash-size`
  - 17. `--aria-recover`
  - 18. `--aria-recover-options`
  - 19. `--aria-repair-threads`
  - 20. `--aria-sort-buffer-size`
  - 21. `--aria-stats-method`
  - 22. `--aria-sync-log-dir`
  - 23. `--aria-used-for-temp-tables`
  - 24. `--deadlock-search-depth-long`
  - 25. `--deadlock-search-depth-short`
  - 26. `--deadlock-timeout-long`
  - 27. `--deadlock-timeout-short`
  - 28. **Aria Storage Engine Options and System Variables**
- 5. **MyRocks Storage Engine Options**
- 6. **S3 Storage Engine Options**
  - 1. `--s3-access-key`
  - 2. `--s3-block-size`
  - 3. `--s3-bucket`
  - 4. `--s3-debug`
  - 5. `--s3-host-name`
  - 6. `--s3-pagecache-age-threshold`
  - 7. `--s3-pagecache-buffer-size`
  - 8. `--s3-pagecache-division-limit`
  - 9. `--s3-pagecache-file-hash-size`
  - 10. `--s3-port`
  - 11. `--s3-protocol-version`
  - 12. `--s3-region`
  - 13. `--s3-secret-key`
  - 14. `--s3-slave-ignore-updates`
  - 15. `--s3-use-http`
- 7. **CONNECT Storage Engine Options**
  - 1. `--connect-class-path`
  - 2. `--connect-cond-push`
  - 3. `--connect-conv-size`
  - 4. `--connect-default-depth`
  - 5. `--connect-default-prec`
  - 6. `--connect-enable-mongo`
  - 7. `--connect-exact-info`
  - 8. `--connect-force-bson`
  - 9. `--connect-idx-map`
  - 10. `--connect-java-wrapper`
  - 11. `--connect-json-all-path`
  - 12. `--connect-json-grp-size`
  - 13. `--connect-json-null`
  - 14. `--connect-jvm-path`
  - 15. `--connect-type-conv`
  - 16. `--connect-use-tempfile`
  - 17. `--connect-work-size`
  - 18. `--connect-xtrace`
- 19. **CONNECT Storage Engine Options and System Variables**
- 8. **Spider Storage Engine Options**
- 9. **Mroonga Storage Engine Options**
- 10. **TokuDB Storage Engine Options**

- 10. [TOKUDB Storage Engine Options](#)
- 9. [Performance Schema Options](#)
  - 1. [--performance-schema](#)
  - 2. [--performance-schema-accounts-size](#)
  - 3. [--performance-schema-consumer-events-stages-current](#)
  - 4. [--performance-schema-consumer-events-stages-history](#)
  - 5. [--performance-schema-consumer-events-stages-history-long](#)
  - 6. [--performance-schema-consumer-events-statements-current](#)
  - 7. [--performance-schema-consumer-events-statements-history](#)
  - 8. [--performance-schema-consumer-events-statements-history-long](#)
  - 9. [--performance-schema-consumer-events-waits-current](#)
  - 10. [--performance-schema-consumer-events-waits-history](#)
  - 11. [--performance-schema-consumer-events-waits-history-long](#)
  - 12. [--performance-schema-consumer-global-instrumentation](#)
  - 13. [--performance-schema-consumer-statements-digest](#)
  - 14. [--performance-schema-consumer-thread-instrumentation](#)
  - 15. [--performance-schema-digests-size](#)
  - 16. [--performance-schema-events-stages-history-long-size](#)
  - 17. [--performance-schema-events-stages-history-size](#)
  - 18. [--performance-schema-events-statements-history-long-size](#)
  - 19. [--performance-schema-events-statements-history-size](#)
  - 20. [--performance-schema-events-transactions-history-long-size](#)
  - 21. [--performance-schema-events-transactions-history-size](#)
  - 22. [--performance-schema-events-waits-history-long-size](#)
  - 23. [--performance-schema-events-waits-history-size](#)
  - 24. [--performance-schema-hosts-size](#)
  - 25. [--performance-schema-max-cond-classes](#)
  - 26. [--performance-schema-max-cond-instances](#)
  - 27. [--performance-schema-max-digest-length](#)
  - 28. [--performance-schema-max-file-classes](#)
  - 29. [--performance-schema-max-file-handles](#)
  - 30. [--performance-schema-max-file-instances](#)
  - 31. [--performance-schema-max-index-stat](#)
  - 32. [--performance-schema-max-memory-classes](#)
  - 33. [--performance-schema-max-metadata-locks](#)
  - 34. [--performance-schema-max-mutex-classes](#)
  - 35. [--performance-schema-max-mutex-instances](#)
  - 36. [--performance-schema-max-prepared-statement-instances](#)
  - 37. [--performance-schema-max-program-instances](#)
  - 38. [--performance-schema-max-sql-text-length](#)

39. `--performance-schema-max-rwlock-classes`
  40. `--performance-schema-max-rwlock-instances`
  41. `--performance-schema-max-socket-classes`
  42. `--performance-schema-max-socket-instances`
  43. `--performance-schema-max-stage-classes`
  44. `--performance-schema-max-statement-classes`
  45. `--performance-schema-max-statement-stack`
  46. `--performance-schema-max-table-handles`
  47. `--performance-schema-max-table-instances`
  48. `--performance-schema-max-table-lock-stat`
  49. `--performance-schema-max-thread-classes`
  50. `--performance-schema-max-thread-instances`
  51. `--performance-schema-session-connect-attrs-size`
  52. `--performance-schema-setup-actors-size`
  53. `--performance-schema-setup-objects-size`
  54. `--performance-schema-users-size`
  55. **Performance Schema Options and System Variables**
10. **Galera Cluster Options**
    1. `--wsrep-auto-increment-control`
    2. `--wsrep-causal-reads`
    3. `--wsrep-certify-nonPK`
    4. `--wsrep-cluster-address`
    5. `--wsrep-cluster-name`
    6. `--wsrep-convert-LOCK-to-trx`
    7. `--wsrep-data-home-dir`
    8. `--wsrep-dbug-option`
    9. `--wsrep-debug`
    10. `--wsrep-desync`
    11. `--wsrep-dirty-reads`
    12. `--wsrep-drupal-282555-workaround`
    13. `--wsrep-forced-binlog-format`
    14. `--wsrep-gtid-domain-id`
    15. `--wsrep-gtid-mode`
    16. `--wsrep-ignore-apply-errors`
    17. `--wsrep-load-data-splitting`
    18. `--wsrep-log-conflicts`
    19. `--wsrep-max-ws-rows`
    20. `--wsrep-max-ws-size`
    21. `--wsrep-mode`
    22. `--wsrep-mysql-replication-bundle`
    23. `--wsrep-new-cluster`
    24. `--wsrep-node-address`
    25. `--wsrep-node-incoming-address`
    26. `--wsrep-node-name`
    27. `--wsrep-notify-cmd`
    28. `--wsrep-on`
    29. `--wsrep-OSU-method`
    30. `--wsrep-provider`
    31. `--wsrep-provider-options`
    32. `--wsrep-recover`
    33. `--wsrep-reject_queries`
    34. `--wsrep-replicate-myisam`
    35. `--wsrep-restart-slave`
    36. `--wsrep-retry-autocommit`
    37. `--wsrep-slave-FK-checks`
    38. `--wsrep-slave-threads`
    39. `--wsrep-slave-LIK-checks`

- 35. [--wsrep-slave-or-checks](#)
  - 40. [--wsrep-sr-store](#)
  - 41. [--wsrep-sst-auth](#)
  - 42. [--wsrep-sst-donor](#)
  - 43. [--wsrep-sst-donor-rejects-queries](#)
  - 44. [--wsrep-sst-method](#)
  - 45. [--wsrep-sst-receive-address](#)
  - 46. [--wsrep-start-position](#)
  - 47. [--wsrep-strict-dll](#)
  - 48. [--wsrep-sync-wait](#)
  - 49. [--wsrep-trx-fragment-size](#)
  - 50. [--wsrep-trx-fragment-unit](#)
  - 51. [Galera Cluster Options and System Variables](#)
11. [Options When Debugging mysqld](#)
- 1. [--core-file](#)
  - 2. [--debug](#)
  - 3. [--debug-assert-if-crashed-table](#)
  - 4. [--debug-binlog-fsync-sleep](#)
  - 5. [--debug-crc-break](#)
  - 6. [--debug-flush](#)
  - 7. [--debug-no-thread-alarm](#)
  - 8. [--debug-no-sync](#)
  - 9. [--debug-sync-timeout](#)
  - 10. [--gdb](#)
  - 11. [--silent-startup](#)
  - 12. [--sync-sys](#)
  - 13. [--thread-alarm](#)
  - 14. [Debugging Options and System Variables](#)
12. [Other Options](#)
- 1. [--allow-suspicious-udfs](#)
  - 2. [--autocommit](#)
  - 3. [--automatic-sp-privileges](#)
  - 4. [--back-log](#)
  - 5. [--basedir](#)
  - 6. [--bind-address](#)
  - 7. [--bootstrap](#)
  - 8. [--check-constraint-checks](#)
  - 9. [--chroot](#)
  - 10. [--column-compression-threshold](#)
  - 11. [--column-compression-zlib-level](#)
  - 12. [--column-compression-zlib-strategy](#)
  - 13. [--column-compression-zlib-wrap](#)
  - 14. [--completion-type](#)
  - 15. [--connect-timeout](#)
  - 16. [--datadir](#)
  - 17. [--date-format](#)
  - 18. [--datetime-format](#)
  - 19. [--deadlock-search-depth-long](#)
  - 20. [--deadlock-search-depth-short](#)
  - 21. [--deadlock-timeout-long](#)
  - 22. [--deadlock-timeout-short](#)
  - 23. [--default-password-lifetime](#)
  - 24. [--default-regex-flags](#)
  - 25. [--default-storage-engine](#)
  - 26. [--default-table-type](#)
  - 27. [--default-tmp-storage-engine](#)
  - 28. [--delay-key-write](#)
  - 29. [--des-key-file](#)
  - 30. [--disconnect-on-expired-password](#)
  - 31. [--div-precision-increment](#)
  - 32. [--encrypt-binlog](#)
  - 33. [--encrypt-tmp-disk-tables](#)
  - 34. [--encrypt-tmp-files](#)
  - 35. [--encryption-algorithm](#)
  - 36. [--engine-condition-pushdown](#)
  - 37. [--eq-range-index-dive-limit](#)
  - 38. [--event-scheduler](#)
  - 39. [--exit-info](#)
  - 40. [--expire-logs-days](#)
  - 41. [--explicit-defaults-for-timestamp](#)
  - 42. [--extra-max-connections](#)

43. --extra-port  
44. --flush  
45. --flush-time  
46. --ft-boolean-syntax  
47. --ft-max-word-len  
48. --ft-min-word-len  
49. --ft-query-expansion-limit  
50. --ft-stopword-file  
51. --general-log  
52. --general-log-file  
53. --getopt-prefix-matching  
54. --group-concat-max-len  
55. --help  
56. --histogram-size  
57. --histogram-type  
58. --host-cache-size  
59. --idle-readonly-transaction-timeout  
60. --idle-transaction-timeout  
61. --idle-write-transaction-timeout  
62. --ignore-db-dirs  
63. --in-predicate-conversion-threshold  
64. --init-connect  
65. --init-file  
66. --interactive-timeout  
67. --large-pages  
68. --local-infile  
69. --lock-wait-timeout  
70. --log  
71. --log-disabled\_statements  
72. --log-error  
73. --log-output  
74. --log-queries-not-using-indexes  
75. --log-ddl-recovery  
76. --log-short-format  
77. --log-slow-admin-statements  
78. --log-slow-admin-statements  
79. --log-slow-file  
80. --log-slow-filter  
81. --log-slow-queries  
82. --log-slow-rate-limit  
83. --log-slow-slave-statements  
84. --log-slow-time  
85. --log-slow-verbosity  
86. --log-tc  
87. --log-tc-size  
88. --log-warnings  
89. --long-query-time  
90. --low-priority-updates  
91. --lower-case-table-names  
92. --master-connect-retry  
93. --max-allowed-packet  
94. --max-connections  
95. --max-connect-errors  
96. --max-delayed-threads  
97. --max-digest-length  
98. --max-error-count  
99. --max-length-for-sort-data  
100. --max-long-data-size  
101. --max-password-errors  
102. --max-prepared-stmt-count  
103. --max-recursive-iterations  
104. --max-rowid-filter-size  
105. --max-session-mem-used  
106. --max-sp-recursion-depth  
107. --max-statement-time  
108. --max-tmp-tables  
109. --max-user-connections  
110. --max-write-lock-count  
111. --memlock  
112. --metadata-locks-cache-size  
113. --metadata-locks-hash-instances  
114. --min-examined-row-limit  
115. --mrr-buffer-size

115. ~~--server-size~~  
116. ~~--multi-range-count~~  
117. ~~--mysql56-temporal-format~~  
118. ~~--ndb-use-copying-alter-table~~  
119. ~~--net-buffer-length~~  
120. ~~--net-read-timeout~~  
121. ~~--net-retry-count~~  
122. ~~--net-write-timeout~~  
123. ~~--one-thread~~  
124. ~~--open-files-limit~~  
125. ~~--pid-file~~  
126. ~~--plugin-load~~  
127. ~~--plugin-load-add~~  
128. ~~--plugin-dir~~  
129. ~~--plugin-maturity~~  
130. ~~--port~~  
131. ~~--port-open-timeout~~  
132. ~~--preload-buffer-size~~  
133. ~~--profiling-history-size~~  
134. ~~--progress-report-time~~  
135. ~~--proxy-protocol-networks~~  
136. ~~--query-cache-info~~  
137. ~~--query-cache-limit~~  
138. ~~--query-cache-min-res-unit~~  
139. ~~--query-cache-size~~  
140. ~~--query-cache-strip-comments~~  
141. ~~--query-cache-type~~  
142. ~~--query-cache-wlock-invalidate~~  
143. ~~--read-rnd-buffer-size~~  
144. ~~--read-only~~  
145. ~~--require-secure-transport~~  
146. ~~--safe-show-database~~  
147. ~~--safe-user-create~~  
148. ~~--safemalloc-mem-limit~~  
149. ~~--secure-auth~~  
150. ~~--secure-file-priv~~  
151. ~~--secure-timestamp~~  
152. ~~--session-track-schema~~  
153. ~~--session-track-state-change~~  
154. ~~--session-track-system-variables~~  
155. ~~--session-track-transaction-info~~  
156. ~~--show-slave-auth-info~~  
157. ~~--skip-automatic-sp-privileges~~  
158. ~~--skip-external-locking~~  
159. ~~--skip-grant-tables~~  
160. ~~--skip-host-cache~~  
161. ~~--skip-large-pages~~  
162. ~~--skip-log-error~~  
163. ~~--skip-name-resolve~~  
164. ~~--skip-networking~~  
165. ~~--skip-partition~~  
166. ~~--skip-show-database~~  
167. ~~--skip-slave-start~~  
168. ~~--skip-ssl~~  
169. ~~--skip-symlink~~  
170. ~~--skip-thread-priority~~  
171. ~~--slow-launch-time~~  
172. ~~--slow-query-log~~  
173. ~~--slow-query-log-file~~  
174. ~~--socket~~  
175. ~~--sort-buffer-size~~  
176. ~~--sql-bin-update-same~~  
177. ~~--sql-if-exists~~  
178. ~~--sql-mode~~  
179. ~~--ssl~~  
180. ~~--ssl-ca~~  
181. ~~--ssl-capath~~  
182. ~~--ssl-cert~~  
183. ~~--ssl-cipher~~  
184. ~~--ssl-crl~~  
185. ~~--ssl-crlpath~~  
186. ~~--ssl-key~~  
187. ~~--stack-trace~~

188. `--standard-compliant-cte`  
189. `--stored-program-cache`  
190. `--strict-password-validation`  
191. `--symbolic-links`  
192. `--sync-frm`  
193. `--system-versioning-alter-history`  
194. `--system-versioning-asof`  
195. `--system-versioning-innodb-algorithm-simple`  
196. `--table-lock-wait-timeout`  
197. `--tc-heuristic-recover`  
198. `--tcp-keepalive-interval`  
199. `--tcp-keepalive-probes`  
200. `--tcp-keepalive-time`  
201. `--tcp-nodelay`  
202. `--temp-pool`  
203. `--test-expect-abort`  
204. `--test-ignore-wrong-options`  
205. `--thread-cache-size`  
206. `--thread-concurrency`  
207. `--thread-handling`  
208. `--thread-pool-dedicated-listener`  
209. `--thread-pool-exact-stats`  
210. `--thread-pool-idle-timeout`  
211. `--thread-pool-max-threads`  
212. `--thread-pool-min-threads`  
213. `--thread-pool-prio-kickup-timer`  
214. `--thread-pool-priority`  
215. `--thread-pool-size`  
216. `--thread-pool-stall-limit`  
217. `--thread-stack`  
218. `--timed-mutexes`  
219. `--time-format`  
220. `--tls_version`  
221. `--tmpdir`  
222. `--transaction-isolation`  
223. `--transaction-alloc-block-size`  
224. `--transaction-prealloc-size`  
225. `--transaction-read-only`  
226. `--updatable-views-with-limit`  
227. `--user`  
228. `--userstat`  
229. `--verbose`  
230. `--version`  
231. `--wait-timeout`

### 13. Other Options and System Variables

### 14. Authentication Plugins - Options and System Variables

1. Authentication Plugin - ed25519
  1. `ed25519`
2. Authentication Plugin - gssapi
  1. `gssapi`
  2. `gssapi_keytab_path`
  3. `gssapi_principal_name`
  4. `gssapi_mech_name`
3. Authentication Plugin - named\_pipe
  1. `named_pipe`
4. Authentication Plugin - pam
  1. `pam`
  2. `pam_debug`
  3. `pam_use_cleartext_plugin`
  4. `pam_winbind_workaround`
5. Authentication Plugin - unix\_socket
  1. `unix_socket`

### 15. Encryption Plugins - Options and System Variables

1. Encryption Plugin - aws\_key\_management
  1. `aws_key_management`
  2. `aws_key_management_key_spec`
  3. `aws_key_management_log_level`
  4. `aws_key_management_master_key_id`
  5. `aws_key_management_mock`

- 6. `aws_key_management_region`
- 7. `aws_key_management_request_time`
- 8. `aws_key_management_rotate_key`
- 2. **Encryption Plugin -**
  - `file_key_management`
  - 1. `file_key_management`
  - 2. `file_key_management_encryption_alg`
  - 3. `file_key_management_filekey`
  - 4. `file_key_management_filename`
- 16. **Password Validation Plugins - Options and System Variables**
  - 1. **Password Validation Plugin -**
    - `simple_password_check`
    - 1. `simple_password_check`
    - 2. `simple_password_check_digits`
    - 3. `simple_password_check_letters_same_case`
    - 4. `simple_password_check_minimal_length`
    - 5. `simple_password_check_other_characters`
  - 2. **Password Validation Plugin -**
    - `cracklib_password_check`
    - 1. `cracklib_password_check`
    - 2. `cracklib_password_check_dictionary`
- 17. **Audit Plugins - Options and System Variables**
  - 1. **Audit Plugin - server\_audit**
    - 1. `server-audit`
    - 2. `server-audit-events`
    - 3. `server-audit-excl-users`
    - 4. `server-audit-file-path`
    - 5. `server-audit-file-rotate-now`
    - 6. `server-audit-file-rotate-size`
    - 7. `server-audit-file-rotations`
    - 8. `server-audit-incl-users`
    - 9. `server-audit-logging`
    - 10. `server-audit-mode`
    - 11. `server-audit-output-type`
    - 12. `server-audit-query-limit`
    - 13. `server-audit-syslog-facility`
    - 14. `server-audit-syslog-ident`
    - 15. `server-audit-syslog-info`
    - 16. `server-audit-syslog-priority`
  - 2. **Audit Plugin - SQL\_ERROR\_LOG**
    - 1. `sql_error_log`
    - 2. `sql_error_log_filename`
    - 3. `sql_error_log_filename`
    - 4. `sql_error_log_rate`
    - 5. `sql_error_log_rotate`
    - 6. `sql_error_log_rotations`
    - 7. `sql_error_log_size_limit`
  - 3. **Audit Plugin -**
    - `QUERY_RESPONSE_TIME_AUDIT`
    - 1. `query_response_time_audit`
- 18. **Daemon Plugins - Options and System Variables**
  - 1. **Daemon Plugin - handlersocket**
    - 1. `handlersocket-accept-balance`
    - 2. `handlersocket-address`
    - 3. `handlersocket-backlog`
    - 4. `handlersocket-epoll`
    - 5. `handlersocket-plain-secret`
    - 6. `handlersocket-plain-secret-wr`
    - 7. `handlersocket-port`
    - 8. `handlersocket-port-wr`
    - 9. `handlersocket-rcvbuf`
    - 10. `handlersocket-readsize`
    - 11. `handlersocket-sndbuf`
    - 12. `handlersocket-threads`
    - 13. `handlersocket-threads-wr`
    - 14. `handlersocket-timeout`
    - 15. `handlersocket-verbose`
    - 16. `handlersocket-wrlock-timeout`
- 19. **Information Schema Plugins - Options and System Variables**

- and System variables
1. Information Schema Plugin - DISKS
    1. disks
  2. Information Schema Plugin - feedback
    1. feedback
    2. feedback\_http\_proxy
    3. feedback\_send\_retry\_wait
    4. feedback\_send\_timeout
    5. feedback\_url
    6. feedback\_user\_info
  3. Information Schema Plugin - LOCALES
    1. locales
  4. Information Schema Plugin - METADATA\_LOCK\_INFO
    1. metadata\_lock\_info
  5. Information Schema Plugin - QUERY\_CACHE\_INFO
    1. query\_cache\_info
  6. Information Schema Plugin - QUERY\_RESPONSE\_TIME
    1. query\_response\_time
    2. query\_response\_time\_flush
    3. query\_response\_time\_range\_base
    4. query\_response\_time\_exec\_time\_debt
    5. query\_response\_time\_stats
  7. Information Schema Plugin - user\_variables
    1. user\_variables
  8. Information Schema Plugin - WSREP\_MEMBERSHIP
    1. wsrep\_membership
  9. Information Schema Plugin - WSREP\_STATUS
    1. wsrep\_status
20. Replication Plugins - Options and System Variables
1. Replication Plugin - rpl\_semi\_sync\_master
    1. rpl\_semi\_sync\_master
    2. rpl-semi-sync-master-enabled
    3. rpl-semi-sync-master-timeout
    4. rpl-semi-sync-master-trace-level
    5. rpl-semi-sync-master-wait-no-slave
    6. rpl-semi-sync-master-wait-point
  2. Replication Plugin - rpl\_semi\_sync\_slave
    1. rpl\_semi\_sync\_slave
    2. rpl-semi-sync-slave-delay-master
    3. rpl-semi-sync-slave-kill-conn-timeout
    4. rpl-semi-sync-slave-enabled
    5. rpl-semi-sync-slave-trace-level
21. Default Values

This page lists all of the options for

```
mysqld
/
mariadb
```

, ordered by topic. For a full alphabetical list of all mysqld options, as well as server and status variables, see [Full list of MariaDB options, system and status variables](#).

In many cases, the entry here is a summary, and links to the full description.

By convention, [server variables](#) have usually been specified with an underscore in the configuration files, and a dash on the command line. You can however specify underscores as dashes - they are interchangeable.

See [mysqld startup options](#) for which files and groups mysqld reads for it's default options.

## Option Prefixes

--auto-set--\*

- **Description:** Sets the option value automatically. Only supported for certain options. Available in [MariaDB 10.1.7](#) and later.

--disable--\*

- **Description:** For all boolean options, disables the setting (equivalent to setting it to

0  
). Same as  
--skip

--enable--\*

- **Description:** For all boolean options, enables the setting (equivalent to setting it to

1  
).

--loose--\*

- **Description:** Don't produce an error if the option doesn't exist.

--maximum--\*

- **Description:** Sets the maximum value for the option.

--skip--\*

- **Description:** For all boolean options, disables the setting (equivalent to setting it to

0  
). Same as  
--disable

## Option File Options

--defaults-extra-file

- **Commandline:**

--defaults-extra-file=name

- **Description:** Read this extra option file after all other option files are read.

◦ See [Configuring MariaDB with Option Files](#).

---

--defaults-file

- **Commandline:**

--defaults-file=name

- **Description:** Only read options from the given option file.

◦ See [Configuring MariaDB with Option Files](#).

---

```
--defaults-group-suffix
```

- **Commandline:**

```
--defaults-group-suffix=name
```

- **Description:** In addition to the default option groups, also read option groups with the given suffix.

- See [Configuring MariaDB with Option Files](#) .

---

```
--no-defaults
```

- **Commandline:**

```
--no-defaults
```

- **Description:** Don't read options from any option file.

- See [Configuring MariaDB with Option Files](#) .

---

```
--print-defaults
```

- **Commandline:**

```
--print-defaults
```

- **Description:** Read options from option files, print all option values, and then exit the program.

- See [Configuring MariaDB with Option Files](#) .

## Compatibility Options

The following options have been added to MariaDB to make it more compliant with other MariaDB and MySQL versions:

```
-a, --ansi
```

- **Description:** Use ANSI SQL syntax instead of MySQL syntax. This mode will also set [transaction isolation level serializable](#) .

---

```
--new
```

- **Description:** Use new functionality that will exist in next version of MariaDB. This function exists to make it easier to prepare for an upgrade. For version 5.1 this functions enables the LIST and RANGE partitions functions for ndbcluster.

---

```
--old-style-user-limits
```

- **Description:** Enable old-style user limits (before MySQL 5.0.3, user resources were counted per each user+host vs. per account).

---

```
--safe-mode
```

- **Description:** Disable some potential unsafe optimizations. For 5.2, [INSERT DELAYED](#) is disabled, [myisam\\_recover\\_options](#) is set to DEFAULT (automatically recover crashed MyISAM files) and the [query cache](#) is disabled. For [Aria](#) tables, disable bulk insert optimization to enable one to use [aria\\_read\\_log](#) to recover tables even if tables are deleted (good for testing recovery).

---

```
--skip-new
```

- **Description:** Disables `--new` in 5.2. In 5.1 used to disable some new potentially unsafe functions.
- 

## Compatibility Options and System Variables

- `--old`
- `--old-alter-table`
- `--old-mode`
- `--old-passwords`
- `--show-old-temporals`

## Locale Options

`--character-set-client-handshake`

- **Commandline:**

`--character-set-client-handshake`

- **Description:** Don't ignore client side character set value sent during handshake.
- 

`--default-character-set`

- **Commandline:**

`--default-character-set=name`

- **Description:** Still available as an option for setting the default character set for clients and their connections, it was deprecated and removed in MariaDB 10.2 as a server option. Use [character-set-server](#) instead.
- 

`--language`

- **Description:** This option can be used to set the server's language for error messages. This option can be specified either as a language name or as the path to the directory storing the language's [error message file](#). See [Server Locales](#) for a list of supported locales and their associated languages.

- This option is deprecated. Use the

`lc_messages`

and

`lc_messages_dir`

system variables instead.

- See [Setting the Language for Error Messages](#) for more information.
- 

## Locale Options and System Variables

- [character-set-filesystem](#)
- [character-set-client](#)
- [character-set-connection](#)
- [character-set-database](#)
- [character-set-filesystem](#)
- [character-set-results](#)
- [character-set-server](#)
- [character-set-system](#)
- [character-sets-dir](#)
- [collation-connection](#)
- [collation-database](#)
- [collation-server](#)
- [default-week-format](#)
- [default-time-zone](#)
- [lc-messages](#)

- [lc-messages-dir](#)
- [lc-time-names](#)

## Windows Options

`--console`

- **Description:** Windows-only option that keeps the console window open and for writing log messages to stderr and stdout. If specified together with [--log-error](#), the last option will take precedence.
- 

`--install`

- **Description:** Windows-only option that installs the

`mysqld`  
process as a Windows service.

- The Windows service created with this option [auto-starts](#). If you want a service that is [started on demand](#), then use the

`--install-manual`

option.

- This option takes a service name as an argument. If this option is provided without a service name, then the service name defaults to "MySQL".
  - This option is deprecated and may be removed in a future version. See [MDEV-19358](#) for more information.
- 

`--install-manual`

- **Description:** Windows-only option that installs the

`mysqld`  
process as a Windows service.

- The Windows service created with this option is [started on demand](#). If you want a service that [auto-starts](#), use the

`--install`

option.

- This option takes a service name as an argument. If this option is provided without a service name, then the service name defaults to "MySQL".
  - This option is deprecated and may be removed in a future version. See [MDEV-19358](#) for more information.
- 

`--remove`

- **Description:** Windows-only option that removes the Windows service created by the

`--install`

or

`--install-manual`

options.

- This option takes a service name as an argument. If this option is provided without a service name, then the service name defaults to "MySQL".
  - This option is deprecated and may be removed in a future version. See [MDEV-19358](#) for more information.
- 

`--slow-start-timeout`

- **Description:** Windows-only option that defines the maximum number of milliseconds that the service control manager should wait before trying to

kill the Windows service during startup. Defaults to  
15000

---

--standalone

- **Description:** Windows-only option that has no effect. Kept for compatibility reasons.
- 

## Windows Options and System Variables

The following options and system variables are related to using MariaDB on Windows:

- --named-pipe

## Replication and Binary Logging Options

The following options are related to [replication](#) and the [binary log](#) :

--abort-slave-event-count

- **Commandline:**  
--abort-slave-event-count=#
  - **Description:** Option used by mysql-test for debugging and testing of replication.
- 

--binlog-do-db

- **Commandline:**  
--binlog-do-db=name
  - **Description:** This option allows you to configure a [replication master](#) to write statements and transactions affecting databases that match a specified name into its [binary log](#). Since the filtered statements or transactions will not be present in the [binary log](#), its replication slaves will not be able to replicate them.
    - This option will **not** work with cross-database updates with [statement-based logging](#). See the [Statement-Based Logging](#) section for more information.
    - This option can **not** be set dynamically.
    - When setting it on the command-line or in a server [option group](#) in an [option file](#), the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times.
    - See [Replication Filters](#) for more information.
- 

--binlog-ignore-db

- **Commandline:**  
--binlog-ignore-db=name
- **Description:** This option allows you to configure a [replication master](#) to **not** write statements and transactions affecting databases that match a specified name into its [binary log](#). Since the filtered statements or transactions will not be present in the [binary log](#), its replication slaves will not be able to replicate them.
  - This option will **not** work with cross-database updates with [statement-based logging](#). See the [Statement-Based Logging](#) section for more information.
  - This option can **not** be set dynamically.
  - When setting it on the command-line or in a server [option group](#) in an [option file](#), the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times.
  - See [Replication Filters](#) for more information.

```
--binlog-row-event-max-size
```

- **Commandline:**

```
--binlog-row-event-max-size=#
```

- **Description:** The maximum size of a row-based [binary log](#) event in bytes. Rows will be grouped into events smaller than this size if possible. The value has to be a multiple of 256.

- **Default value**

◦

8192  
(>= [MariaDB 10.2.4](#))

◦

1024  
(<= [MariaDB 10.2.3](#))

---

```
--disconnect-slave-event-count
```

- **Commandline:**

```
--disconnect-slave-event-count=#
```

- **Description:** Option used by mysql-test for debugging and testing of replication.
- 

```
--flashback
```

- **Commandline:**

```
--flashback
```

- **Description:** Setup the server to use flashback. This enables the [binary log](#) and sets  
`binlog_format=ROW`

- **Introduced:** [MariaDB 10.2.4](#)
- 

```
--init-rpl-role
```

- **Commandline:**

```
--init-rpl-role=name
```

- **Description:** Set the replication role.
- 

```
--log-basename
```

- **Commandline:**

```
--log-basename=name
```

- **Description:** Basename for all log files and the .pid file. This sets all log file names at once (in 'datadir') and is normally the only option you need for specifying log files. This is especially recommended to be set if you are using [replication](#) as it ensures that your log file names are not dependent on your host name. Sets names for [log-bin](#) , [log-bin-index](#) , [relay-log](#) , [relay-log-index](#) , [general-log-file](#) ,

```
--log-slow-query-log-file
```

,

```
--log-error-file
```

, and [pid-file](#) .

- **Introduced:** [MariaDB 5.2](#)
- 

```
--log-bin-trust-routine-creators
```

- **Commandline:**

```
--log-bin-trust-routine-creators
```

- **Description:** Deprecated, use [log-bin-trust-function-creators](#) .
- 

```
--master-host
```

- **Commandline:**

```
--master-host=name
```

- **Description:** Master hostname or IP address for replication. If not set, the slave thread will not be started. Note that the setting of master-host will be ignored if there exists a valid master.info file.
- 

```
--master-info-file
```

- **Commandline:**

```
--master-info-file=name
```

- **Description:** Name and location of the file where the

MASTER\_LOG\_FILE

and

MASTER\_LOG\_POS

options (i.e. the [binary log](#) position on the master) and most other

[CHANGE MASTER](#)

options are written. The [slave's I/O thread](#) keeps this [binary log](#) position updated as it downloads events.

- See [CHANGE MASTER TO: Option Persistence](#) for more information.
- 

```
--master-password
```

- **Commandline:**

```
--master-password=name
```

- **Description:** The password the slave thread will authenticate with when connecting to the master. If not set, an empty password is assumed. The value in master.info will take precedence if it can be read.
- 

```
--master-port
```

- **Commandline:**

```
--master-port=#
```

- **Description:** The port the master is listening on. If not set, the compiled setting of MYSQL\_PORT is assumed. If you have not tinkered with configure options, this should be 3306. The value in master.info will take precedence if it can be read.
- 

```
--master-retry-count
```

- **Commandline:**

```
--master-retry-count=#
```

- **Description:** Number of times a slave will attempt to connect to a master before giving up. The retry interval is determined by the MASTER\_CONNECT\_RETRY option for the CHANGE MASTER statement. A value of 0 means the slave will not stop attempting to reconnect. Reconnects are triggered when a slave has timed out. See [slave\\_net\\_timeout](#) .

- **Default Value:**

86400

- **Range - 32 bit:**  
0 to 4294967295

- **Range - 64 bit:**  
0 to 18446744073709551615
- 

--master-ssl

- **Commandline:**  
--master-ssl
  - **Description:** Enable the slave to [connect to the master using TLS](#).
- 

--master-ssl-ca

- **Commandline:**  
--master-ssl-ca[=name]
  - **Description:** Master TLS CA file. Only applies if you have enabled [master-ssl](#).
- 

--master-ssl-capath

- **Commandline:**  
--master-ssl-capath[=name]
  - **Description:** Master TLS CA path. Only applies if you have enabled [master-ssl](#).
- 

--master-ssl-cert

- **Commandline:**  
--master-ssl-cert[=name]
  - **Description:** Master TLS certificate file name. Only applies if you have enabled [master-ssl](#).
- 

--master-ssl-cipher

- **Commandline:**  
--master-ssl-cipher[=name]
  - **Description:** Master TLS cipher. Only applies if you have enabled [master-ssl](#).
- 

--master-ssl-key

- **Commandline:**  
--master-ssl-key[=name]
  - **Description:** Master TLS keyfile name. Only applies if you have enabled [master-ssl](#).
- 

--master-user

- **Commandline:**

```
--master-user=name
```

- **Description:** The username the slave thread will use for authentication when connecting to the master. The user must have FILE privilege. If the master user is not set, user test is assumed. The value in master.info will take precedence if it can be read.
- 

```
--max-binlog-dump-events
```

- **Commandline:**

```
--max-binlog-dump-events=#
```

- **Description:** Option used by mysql-test for debugging and testing of replication.
- 

```
--replicate-rewrite-db
```

- **Commandline:**

```
--replicate-rewrite-db=master_database->slave_database
```

- **Description:** This option allows you to configure a [replication slave](#) to rewrite database names. It uses the format

```
master_database->slave_database
```

. If a slave encounters a [binary log](#) event in which the default database (i.e. the one selected by the

```
USE
```

statement) is

```
master_database
```

, then the slave will apply the event in

```
slave_database
```

instead.

- This option will **not** work with cross-database updates with [statement-based logging](#) . See the [Statement-Based Logging](#) section for more information.
- This option only affects statements that involve tables. This option does not affect statements involving the database itself, such as

```
CREATE DATABASE
```

```
,
```

```
ALTER DATABASE
```

, and

```
DROP DATABASE
```

- This option can **not** be set dynamically.
  - When setting it on the command-line or in a server [option group](#) in an [option file](#) , the option does not accept a comma-separated list. If you would like to specify multiple filters, then you need to specify the option multiple times.
  - See [Replication Filters](#) for more information.
- 

```
--replicate-same-server-id
```

- **Commandline:**

```
--replicate-same-server-id
```

- **Description:** In replication, if set to 1, do not skip events having our server id. Default value is 0 (to break infinite loops in circular replication). Can't be set to 1 if [log-slave-updates](#) is used.
- 

```
--sporadic-binlog-dump-fail
```

- **Commandline:**

```
--sporadic-binlog-dump-fail
```

- **Description:** Option used by mysql-test for debugging and testing of replication.
- 

```
--sysdate-is-now
```

- **Commandline:**

```
--sysdate-is-now
```

- **Description:** Non-default option to alias [SYSDATE\(\)](#) to [NOW\(\)](#) to make it safe for replication . Since 5.0, SYSDATE() has returned a 'dynamic' value different for different invocations, even within the same statement.
- 

## Replication and Binary Logging Options and System Variables

The following options and system variables are related to [replication](#) and the [binary log](#) :

- [auto-increment-increment](#)
- [auto-increment-offset](#)
- [binlog-annotate-row-events](#)
- [binlog-cache-size](#)
- [binlog-checksum](#)
- [binlog-commit-wait-count](#)
- [binlog-commit-wait-usec](#)
- [binlog-direct-non-transactional-updates](#)
- [binlog-expire-logs-seconds](#)
- [binlog-file-cache-size](#)
- [binlog-format](#)
- [binlog-optimize-thread-scheduling](#)
- [binlog-row-image](#)
- [binlog-row-metadata](#)
- [binlog-stmt-cache-size](#)
- [default-master-connection](#)
- [gtid-cleanup-batch-size](#)
- [gtid-domain-id](#)
- [gtid-ignore-duplicates](#)
- [gtid-strict-mode](#)
- [init-slave](#)
- [log-bin](#)
- [log-bin-compress](#)
- [log-bin-compress-min-len](#)
- [log-bin-index](#)
- [log-bin-trust-function-creators](#)
- [log-slave-updates](#)
- [master-verify-checksum](#)
- [max-binlog-cache-size](#)
- [max-binlog-size](#)
- [max-binlog-stmt-cache-size](#)
- [max-relay-log-size](#)
- [read-binlog-speed-limit](#)
- [relay-log](#)
- [relay-log-index](#)
- [relay-log-info-file](#)
- [relay-log-purge](#)
- [relay-log-recovery](#)
- [relay-log-space-limit](#)
- [replicate-annotate-row-events](#)
- [replicate-do-db](#)
- [replicate-do-table](#)
- [replicate-events-marked-for-skip](#)
- [replicate-ignore-db](#)
- [replicate-ignore-table](#)
- [replicate-wild-do-table](#)
- [replicate-wild-ignore-table](#)
- [report-host](#)
- [report-password](#)

- [report-port](#)
- [report-user](#)
- [rpl-recovery-rank](#)
- [server-id](#)
- [slave-compressed-protocol](#)
- [slave-ddl-exec-mode](#)
- [slave-domain-parallel-threads](#)
- [slave-exec-mode](#)
- [slave-load-tmpdir](#)
- [slave-max-allowed-packet](#)
- [slave-net-timeout](#)
- [slave-parallel-max-queued](#)
- [slave-parallel-threads](#)
- [slave-run-triggers-for-rbr](#)
- [slave-skip-errors](#)
- [slave-sql-verify-checksum](#)
- [slave-transaction-retries](#)
- [slave\\_transaction\\_retry\\_errors](#)
- [slave\\_transaction\\_retry\\_interval](#)
- [slave-type-conversions](#)
- [sync-binlog](#)
- [sync-master-info](#)
- [sync-relay-log](#)
- [sync-relay-log-info](#)

## Semisynchronous Replication Options and System Variables

The options and system variables related to [Semisynchronous Replication](#) are described [here](#).

## Optimizer Options

--record-buffer

- **Commandline:**  
--record-buffer=#
- **Description:** Old alias for [read\\_buffer\\_size](#).
- **Removed:** [MariaDB 5.5](#)

---

--table-cache

- **Commandline:**  
--table-open-cache=#
- **Description:** Removed; use [--table-open-cache](#) instead.
- **Removed:** [MariaDB 5.1.3](#)

---

## Optimizer Options and System Variables

- [alter-algorithm](#)
- [analyze-sample-percentage](#)
- [big-tables](#)
- [bulk-insert-buffer-size](#)
- [expensive-subquery-limit](#)
- [join-buffer-size](#)
- [join-buffer-space-limit](#)
- [join-cache-level](#)
- [max-heap-table-size](#)
- [max-join-size](#)
- [max-seeks-for-key](#)
- [max-sort-length](#)
- [mrr-buffer-size](#)
- [optimizer-maxsel-arg-weight](#)
- [optimizer-prune-level](#)

- [optimizer-search-depth](#)
- [optimizer-selectivity-sampling-limit](#)
- [optimizer-switch](#)
- [optimizer-trace](#)
- [optimizer-trace-max-mem-size](#)
- [optimizer-use-condition-selectivity](#)
- [query-alloc-block-size](#)
- [query-prealloc-size](#)
- [range-alloc-block-size](#)
- [read-buffer-size](#)
- [rowid-merge-buff-size](#)
- [table-definition-cache](#)
- [table-open-cache](#)
- [table-open-cache-instances](#)
- [tmp-disk-table-size](#)
- [tmp-memory-table-size](#)
- [tmp-table-size](#)
- [use-stat-tables](#)

## Storage Engine Options

--skip-bdb

- **Commandline:**

----skip-bdb

- **Description:** Deprecated option; Exists only for compatibility with very old my.cnf files.
- **Removed:** [MariaDB 10.5.1](#)

---

## MyISAM Storage Engine Options

The options related to the [MyISAM](#) storage engine are described below.

--external-locking

- **Commandline:**

--external-locking

- **Description:** Use system (external) locking (disabled by default). With this option enabled you can run [myisamchk](#) to test (not repair) tables while the server is running. Disable with [--skip-external-locking](#).

--log-isam

- **Commandline:**

--log-isam[=file\_name]

- **Description:** Enable the [MyISAM log](#), which logs all MyISAM changes to file. If no filename is provided, the default, *myisam.log* is used.

---

## MyISAM Storage Engine Options and System Variables

Some options and system variables related to the [MyISAM](#) storage engine can be found [here](#). Direct links to many of them can be found below.

- [concurrent-insert](#)
- [delayed-insert-limit](#)
- [delayed-insert-timeout](#)
- [delayed-queue-size](#)
- [keep-files-on-create](#)
- [key-buffer-size](#)
- [key-cache-age-threshold](#)
- [key-cache-block-size](#)
- [key-cache-division-limit](#)
- [key-cache-file-hash-size](#)

- [key-cache-segments](#)
- [myisam-block-size](#)
- [myisam-data-pointer-size](#)
- [myisam-max-sort-file-size](#)
- [myisam-mmap-size](#)
- [myisam-recover-options](#)
- [myisam-repair-threads](#)
- [myisam-sort-buffer-size](#)
- [myisam-stats-method](#)
- [myisam-use-mmap](#)

## InnoDB Storage Engine Options

The options related to the [InnoDB](#) storage engine are described below.

--innodb

- **Commandline:**

```
--innodb=value  
,  
--skip-innodb
```

- **Description:** This variable controls whether or not to load the InnoDB storage engine. Possible values are

```
ON  
,  
OFF  
,  
FORCE  
or  
FORCE_PLUS_PERMANENT
```

(from [MariaDB 5.5](#) ). If set to

OFF

(the same as --skip-innodb), since InnoDB is the default storage engine, the server will not start unless another storage engine has been chosen with [--default-storage-engine](#) .

FORCE

means that the storage engine must be successfully loaded, or else the server won't start.

FORCE\_PLUS\_PERMANENT

enables the plugin, but if plugin cannot initialize, the server will not start. In addition, the plugin cannot be uninstalled while the server is running.

---

--innodb-cmp

- **Commandline:**

```
--innodb-cmp
```

- **Description:**

- **Default:**

ON

---

--innodb-cmp-reset

- **Commandline:**

```
--innodb-cmp-reset
```

- **Description:**

- **Default:**

ON

---

--innodb-cmpmem

- **Commandline:**  
--innodb-cmpmem

- **Description:**
- **Default:**  
ON

---

```
--innodb-cmpmem-reset
```

- **Commandline:**  
--innodb-cmpmem-reset

- **Description:**
- **Default:**  
ON

---

```
--innodb-file-io-threads
```

- **Commandline:**  
--innodb-file-io-threads

- **Description:**
- **Default:**  
4

- **Removed:** [MariaDB 10.3.0](#)
- 

```
--innodb-index-stats
```

- **Commandline:**  
--innodb-index-stats

- **Description:**
- **Default:**  
ON

- **Removed:** [MariaDB 10.0.0](#)
- 

```
--innodb-lock-waits
```

- **Commandline:**  
--innodb-lock-waits

- **Description:**
- **Default:**  
ON

---

```
--innodb-locks
```

- **Commandline:**  
--innodb-locks

- **Description:**

- **Default:**

ON

---

--innodb-rseg

- **Commandline:**

--innodb-rseg

- **Description:**

- **Default:**

ON

- **Removed:** [MariaDB 10.0.0](#)
- 

--innodb-status-file

- **Commandline:**

--innodb-status-file

- **Description:**

- **Default:**

FALSE

---

--innodb-sys-indexes

- **Commandline:**

--innodb-sys-indexes

- **Description:**

- **Default:**

ON

---

--innodb-sys-stats

- **Commandline:**

--innodb-sys-stats

- **Description:**

- **Default:**

ON

- **Removed:** [MariaDB 10.0.0](#)
- 

--innodb-sys-tables

- **Commandline:**

--innodb-sys-tables

- **Description:**

- **Default:**

ON

---

```
--innodb-table-stats
```

- **Commandline:**

```
--innodb-table-stats
```

- **Description:**

- **Default:**

```
ON
```

- **Removed:** MariaDB 10.0.0

---

```
--innodb-trx
```

- **Commandline:**

```
--innodb-trx
```

- **Description:**

- **Default:**

```
ON
```

---

## InnoDB Storage Engine Options and System Variables

Some options and system variables related to the [InnoDB](#) storage engine can be found [here](#). Direct links to many of them can be found below.

- [ignore-built-innodb](#)
- [innodb-adaptive-checkpoint](#)
- [innodb-adaptive-flushing](#)
- [innodb-adaptive-flushing-lwm](#)
- [innodb-adaptive-flushing-method](#)
- [innodb-adaptive-hash-index](#)
- [innodb-adaptive-hash-index-partitions](#)
- [innodb-adaptive-hash-index-parts](#)
- [innodb-adaptive-max-sleep-delay](#)
- [innodb-additional-mem-pool-size](#)
- [innodb-api-bk-commit-interval](#)
- [innodb-api-disable-rowlock](#)
- [innodb-api-enable-binlog](#)
- [innodb-api-enable-mdl](#)
- [innodb-api-trx-level](#)
- [innodb-auto-lru-dump](#)
- [innodb-autoextend-increment](#)
- [innodb-autoinc-lock-mode](#)
- [innodb-background-scrub-data-check-interval](#)
- [innodb-background-scrub-data-compressed](#)
- [innodb-background-scrub-data-interval](#)
- [innodb-background-scrub-data-uncompressed](#)
- [innodb-blocking-buffer-pool-restore](#)
- [innodb-buf-dump-status-frequency](#)
- [innodb-buffer-pool-chunk-size](#)
- [innodb-buffer-pool-dump-at-shutdown](#)
- [innodb-buffer-pool-dump-now](#)
- [innodb-buffer-pool-dump-pct](#)
- [innodb-buffer-pool-evict](#)
- [innodb-buffer-pool-filename](#)
- [innodb-buffer-pool-instances](#)
- [innodb-buffer-pool-load-abort](#)
- [innodb-buffer-pool-load-at-startup](#)
- [innodb-buffer-pool-load-now](#)
- [innodb-buffer-pool-load-pages-abort](#)
- [innodb-buffer-pool-populate](#)
- [innodb-buffer-pool-restore-at-startup](#)
- [innodb-buffer-pool-shm-checksum](#)

- innodb-buffer-pool-shm-key
- innodb-buffer-pool-size
- innodb-change-buffer-max-size
- innodb-change-buffering
- innodb-change-buffering-debug
- innodb-checkpoint-age-target
- innodb-checksum-algorithm
- innodb-checksums
- innodb-cleaner-lsn-age-factor
- innodb-cmp-per-index-enabled
- innodb-commit-concurrency
- innodb-compression-algorithm
- innodb-compression-failure-threshold-pct
- innodb-compression-level
- innodb-compression-pad-pct-max
- innodb-concurrency-tickets
- innodb-corrupt-table-action
- innodb-data-file-path
- innodb-data-home-dir
- innodb-deadlock-detect
- innodb-deadlock-report
- innodb-default-encryption-key-id
- innodb-default-page-encryption-key
- innodb-default-row-format
- innodb-defragment
- innodb-defragment-fill-factor
- innodb-defragment-fill-factor-n-recs
- innodb-defragment-frequency
- innodb-defragment-n-pages
- innodb-defragment-stats-accuracy
- innodb-dict-size-limit
- innodb\_disable\_sort\_file\_cache
- innodb-doublewrite
- innodb-doublewrite-file
- innodb-empty-free-list-algorithm
- innodb-enable-unsafe-group-commit
- innodb-encrypt-log
- innodb-encrypt-tables
- innodb-encrypt-temporary-tables
- innodb-encryption-rotate-key-age
- innodb-encryption-rotation\_iops
- innodb-encryption-threads
- innodb-extra-rsegments
- innodb-extra-undoslots
- innodb-fake-changes
- innodb-fast-checksum
- innodb-fast-shutdown
- innodb-fatal-semaphore-wait-threshold
- innodb-file-format
- innodb-file-format-check
- innodb-file-format-max
- innodb-file-per-table
- innodb-fill-factor
- innodb-flush-log-at-trx-commit
- innodb-flush-method
- innodb-flush-neighbor-pages
- innodb-flush-neighbors
- innodb-flush-sync
- innodb-flushing-avg-loops
- innodb-force-load-corrupted
- innodb-force-primary-key
- innodb-force-recovery
- innodb-foreground-preflush
- innodb-ft-aux-table
- innodb-ft-cache-size
- innodb-ft-enable-diag-print
- innodb-ft-enable-stopword
- innodb-ft-max-token-size
- innodb-ft-min-token-size
- innodb-ft-num-word-optimize
- innodb-ft-result-cache-limit
- innodb-ft-server-stopword-table

- innodb-ft-sort-pll-degree
- innodb-ft-total-cache-size
- innodb-ft-user-stopword-table
- innodb-ibuf-accel-rate
- innodb-ibuf-active-contract
- innodb-ibuf-max-size
- innodb-idle-flush-pct
- innodb-immediate-scrub-data-uncompressed
- innodb-import-table-from-xtrabackup
- innodb-instant-alter-column-allowed
- innodb-instrument-semaphores
- innodb-io-capacity
- innodb-io-capacity-max
- innodb-large-prefix
- innodb-lazy-drop-table
- innodb-lock-schedule-algorithm
- innodb-locking-fake-changes
- innodb-locks-unsafe-for-binlog
- innodb-log-arch-dir
- innodb-log-arch-expire-sec
- innodb-log-archive
- innodb-log-block-size
- innodb-log-buffer-size
- innodb-log-checksum-algorithm
- innodb-log-checksums
- innodb-log-compressed-pages
- innodb-log-file-size
- innodb-log-files-in-group
- innodb-log-group-home-dir
- innodb-log-optimize-ddl
- innodb-log-write-ahead-size
- innodb-lru-flush-size
- innodb-lru-scan-depth
- innodb-max-bitmap-file-size
- innodb-max-changed-pages
- innodb-max-dirty-pages-pct
- innodb-max-dirty-pages-pct-lwm
- innodb-max-purge-lag
- innodb-max-purge-lag-delay
- innodb-max-purge-lag-wait
- innodb-max-undo-log-size
- innodb-merge-sort-block-size
- innodb-mirrored-log-groups
- innodb-monitor-disable
- innodb-monitor-enable
- innodb-monitor-reset
- innodb-monitor-reset-all
- innodb-mtflush-threads
- innodb-numa-interleave
- innodb-old-blocks-pct
- innodb-old-blocks-time
- innodb-online-alter-log-max-size
- innodb-open-files
- innodb-optimize-fulltext-only
- innodb-page-cleaners
- innodb-page-size
- innodb-pass-corrupt-table
- innodb-prefix-index-cluster-optimization
- innodb-print-all-deadlocks
- innodb-purge-batch-size
- innodb-purge-rseg-truncate-frequency
- innodb-purge-threads
- innodb-random-read-ahead
- innodb-read-ahead
- innodb-read-ahead-threshold
- innodb-read-io-threads
- innodb-read-only
- innodb-recovery-update-relay-log
- innodb-replication-delay
- innodb-rollback-on-timeout
- innodb-rollback-segments
- innodb-safe-truncate

- innodb-sched-priority-cleaner
- innodb-scrub-log
- innodb-scrub-log-interval
- innodb-scrub-log-speed
- innodb-show-locks-held
- innodb-show-verbose-locks
- innodb-sort-buffer-size
- innodb-spin-wait-delay
- innodb-stats-auto-recalc
- innodb-stats-auto-update
- innodb-stats-include-delete-marked
- innodb-stats-method
- innodb-stats-modified-counter
- innodb-stats-on-metadata
- innodb-stats-persistent
- innodb-stats-persistent-sample-pages
- innodb-stats-sample-pages
- innodb-stats-transient-sample-pages
- innodb-stats-traditional
- innodb-stats-update-need-lock
- innodb-status-output
- innodb-status-output-locks
- innodb-strict-mode
- innodb-support-xa
- innodb-sync-array-size
- innodb-sync-spin-loops
- innodb-table-locks
- innodb-temp-data-file-path
- innodb-thread-concurrency
- innodb-thread-concurrency-timer-based
- innodb-thread-sleep-delay
- innodb-tmpdir
- innodb-track-changed-pages
- innodb-track-redo-log-now
- innodb-undo-directory
- innodb-undo-log-truncate
- innodb-undo-logs
- innodb-undo-tablespaces
- innodb-use-atomic-writes
- innodb-use-fallocate
- innodb-use-global-flush-log-at-trx-commit
- innodb-use-mtflush
- innodb-use-native\_aio
- innodb-use-purge-thread
- innodb-use-stacktrace
- innodb-use-sys-malloc
- innodb-use-sys-stats-table
- innodb-use-trim
- innodb-write-io-threads
- skip-innodb
- skip-innodb-checksums
- skip-innodb-doublewrite

## Aria Storage Engine Options

The options related to the [Aria](#) storage engine are described below.

`--aria-log-dir-path`

- **Commandline:**

`--aria-log-dir-path=value`

- **Description:** Path to the directory where transactional log should be stored

- **Default:**

SAME AS DATADIR

Some options and system variables related to the [Aria](#) storage engine can be found [here](#). Direct links to many of them can be found below.

- [aria-block-size](#)
- [aria-checkpoint-interval](#)
- [aria-checkpoint-log-activity](#)
- [aria-encrypt-tables](#)
- [aria-force-start-after-recovery-failures](#)
- [aria-group-commit](#)
- [aria-group-commit-interval](#)
- [aria-log-file-size](#)
- [aria-log-purge-type](#)
- [aria-max-sort-file-size](#)
- [aria-page-checksum](#)
- [aria-pagecache-age-threshold](#)
- [aria-pagecache-buffer-size](#)
- [aria-pagecache-division-limit](#)
- [aria-pagecache-file-hash-size](#)
- [aria-recover](#)
- [aria-recover-options](#)
- [aria-repair-threads](#)
- [aria-sort-buffer-size](#)
- [aria-stats-method](#)
- [aria-sync-log-dir](#)
- [aria-used-for-temp-tables](#)
- [deadlock-search-depth-long](#)
- [deadlock-search-depth-short](#)
- [deadlock-timeout-long](#)
- [deadlock-timeout-short](#)

## MyRocks Storage Engine Options

The options and system variables related to the [MyRocks](#) storage engine can be found [here](#).

## S3 Storage Engine Options

The options and system variables related to the [S3](#) storage engine can be found [here](#).

## CONNECT Storage Engine Options

The options related to the [CONNECT](#) storage engine are described below.

### CONNECT Storage Engine Options and System Variables

Some options and system variables related to the [CONNECT](#) storage engine can be found [here](#). Direct links to many of them can be found below.

- [connect-class-path](#)
- [connect-cond-push](#)
- [connect-conv-size](#)
- [connect-default-depth](#)
- [connect-default-prec](#)
- [connect-enable-mongo](#)
- [connect-exact-info](#)
- [connect-force\\_bson](#)
- [connect-idx-map](#)
- [connect-java-wrapper](#)
- [connect-json-all-path](#)
- [connect-json-grp-size](#)
- [connect-json-null](#)
- [connect-jvm-path](#)
- [connect-type-conv](#)
- [connect-use-tempfile](#)
- [connect-work-size](#)
- [connect-xtrace](#)

## Spider Storage Engine Options

The options and system variables related to the [Spider](#) storage engine can be found [here](#).

## Mroonga Storage Engine Options

The options and system variables related to the [Mroonga](#) storage engine can be found [here](#).

# TokuDB Storage Engine Options

The options and system variables related to the [TokuDB](#) storage engine can be found [here](#).

## Performance Schema Options

The options related to the [Performance Schema](#) are described below.

--performance-schema-consumer-events-stages-current

- **Commandline:**

--performance-schema-consumer-events-stages-current

- **Description:** Enable the [events-stages-current](#) consumer.

- **Default:**

OFF

---

--performance-schema-consumer-events-stages-history

- **Commandline:**

--performance-schema-consumer-events-stages-history

- **Description:** Enable the [events-stages-history](#) consumer.

- **Default:**

OFF

---

--performance-schema-consumer-events-stages-history-long

- **Commandline:**

--performance-schema-consumer-events-stages-history-long

- **Description:** Enable the [events-stages-history-long](#) consumer.

- **Default:**

OFF

---

--performance-schema-consumer-events-statements-current

- **Commandline:**

--performance-schema-consumer-events-statements-current

- **Description:** Enable the [events-statements-current](#) consumer. Use

--skip-performance-schema-consumer-events-statements-current  
to disable.

- **Default:**

ON

---

--performance-schema-consumer-events-statements-history

- **Commandline:**

--performance-schema-consumer-events-statements-history

- **Description:** Enable the [events-statements-history](#) consumer.

- **Default:**

OFF

---

--performance-schema-consumer-events-statements-history-long

- **Commandline:**

--performance-schema-consumer-events-statements-history-long

- **Description:** Enable the [events-statements-history-long](#) consumer.

- **Default:**

OFF

---

--performance-schema-consumer-events-waits-current

- **Commandline:**

--performance-schema-consumer-events-waits-current

- **Description:** Enable the [events-waits-current](#) consumer.

- **Default:**

OFF

---

--performance-schema-consumer-events-waits-history

- **Commandline:**

--performance-schema-consumer-events-waits-history

- **Description:** Enable the [events-waits-history](#) consumer.

- **Default:**

OFF

---

--performance-schema-consumer-events-waits-history-long

- **Commandline:**

--performance-schema-consumer-events-waits-history-long

- **Description:** Enable the [events-waits-history-long](#) consumer.

- **Default:**

OFF

---

--performance-schema-consumer-global-instrumentation

- **Commandline:**

--performance-schema-consumer-global-instrumentation

- **Description:** Enable the global-instrumentation consumer. Use

--skip-performance-schema-consumer-global-instrumentation  
to disable.

- **Default:**

ON

---

```
--performance-schema-consumer-statements-digest
```

- **Commandline:**

```
--performance-schema-consumer-statements-digest
```

- **Description:** Enable the statements-digest consumer. Use

```
--skip-performance-schema-consumer-statements-digest  
to disable.
```

- **Default:**

```
ON
```

```
--performance-schema-consumer-thread-instrumentation
```

- **Commandline:**

```
--performance-schema-consumer-thread-instrumentation
```

- **Description:** Enable the statements-thread-instrumentation. Use

```
--skip-performance-schema-thread-instrumentation  
to disable.
```

- **Default:**

```
ON
```

## Performance Schema Options and System Variables

Some options and system variables related to the [Performance Schema](#) can be found [here](#). Direct links to many of them can be found below.

- [performance-schema](#)
- [performance-schema-accounts-size](#)
- [performance-schema-digests-size](#)
- [performance-schema-events-stages-history-long-size](#)
- [performance-schema-events-stages-history-size](#)
- [performance-schema-events-statements-history-long-size](#)
- [performance-schema-events-statements-history-size](#)
- [performance-schema-events-waits-history-long-size](#)
- [performance-schema-events-waits-history-size](#)
- [performance-schema-hosts-size](#)
- [performance-schema-max-cond-classes](#)
- [performance-schema-max-cond-instances](#)
- [performance-schema-max-digest-length](#)
- [performance-schema-max-file-classes](#)
- [performance-schema-max-file-handles](#)
- [performance-schema-max-file-instances](#)
- [performance-schema-max-mutex-classes](#)
- [performance-schema-max-mutex-instances](#)
- [performance-schema-max-rwlock-classes](#)
- [performance-schema-max-rwlock-instances](#)
- [performance-schema-max-socket-classes](#)
- [performance-schema-max-socket-instances](#)
- [performance-schema-max-stage-classes](#)
- [performance-schema-max-statement-classes](#)
- [performance-schema-max-table-handles](#)
- [performance-schema-max-table-instances](#)
- [performance-schema-max-thread-classes](#)
- [performance-schema-max-thread-instances](#)
- [performance-schema-session-connect-atrs-size](#)
- [performance-schema-setup-actors-size](#)
- [performance-schema-setup-objects-size](#)
- [performance-schema-users-size](#)

## Galera Cluster Options

The options related to [Galera Cluster](#) are described below.

```
--wsrep-new-cluster
```

- **Commandline:**

```
--wsrep-new-cluster
```

- **Description:** Bootstrap a cluster. It works by overriding the current value of wsrep\_cluster\_address. It is recommended not to add this option to the config file as this will trigger bootstrap on every server start.
- 

## Galera Cluster Options and System Variables

Some options and system variables related to [Galera Cluster](#) can be found [here](#). Direct links to many of them can be found below.

- [wsrep-auto-increment-control](#)
- [wsrep-causal-reads](#)
- [wsrep-certify-nonPK](#)
- [wsrep-cluster-address](#)
- [wsrep-cluster-name](#)
- [wsrep-convert-LOCK-to-trx](#)
- [wsrep-data-home-dir](#)
- [wsrep-debug-option](#)
- [wsrep-debug](#)
- [wsrep-desync](#)
- [wsrep-dirty-reads](#)
- [wsrep-drupal-282555-workaround](#)
- [wsrep-forced-binlog-format](#)
- [wsrep-gtid-domain-id](#)
- [wsrep-gtid-mode](#)
- [wsrep-ignore-apply-errors](#)
- [wsrep-load-data-splitting](#)
- [wsrep-log-conflicts](#)
- [wsrep-max-ws-rows](#)
- [wsrep-max-ws-size](#)
- [wsrep-mode](#)
- [wsrep-mysql-replication-bundle](#)
- [wsrep-node-address](#)
- [wsrep-node-incoming-address](#)
- [wsrep-node-name](#)
- [wsrep-notify-cmd](#)
- [wsrep-on](#)
- [wsrep-OSU-method](#)
- [wsrep-provider](#)
- [wsrep-provider-options](#)
- [wsrep-recover](#)
- [wsrep-reject\\_queries](#)
- [wsrep-retry-autocommit](#)
- [wsrep-slave-FK-checks](#)
- [wsrep-slave-threads](#)
- [wsrep-slave-UK-checks](#)
- [wsrep-sr-store](#)
- [wsrep-sst-auth](#)
- [wsrep-sst-donor](#)
- [wsrep-sst-donor-rejects-queries](#)
- [wsrep-sst-method](#)
- [wsrep-sst-receive-address](#)
- [wsrep-start-position](#)
- [wsrep-strict-ddl](#)
- [wsrep-sync-wait](#)
- [wsrep-trx\\_fragment\\_size](#)
- [wsrep-trx\\_fragment\\_unit](#)

## Options When Debugging mysqld

```
--debug-assert-if-crashed-table
```

- **Description:** Do an assert in handler::print\_error() if we get a crashed table.

```
--debug-binlog-fsync-sleep
```

- **Description:**

```
--debug-binlog-fsync-sleep=#
```

If not set to zero, sets the number of micro-seconds to sleep after running `fsync()` on the [binary log](#) to flush transactions to disk. This can thus be used to artificially increase the perceived cost of such an `fsync()`.

---

```
--debug-crc-break
```

- **Description:**

```
--debug-crc-break=#
```

Call `my_debug_put_break_here()` if crc matches this number (for debug).

---

```
--debug-flush
```

- **Description:** Default debug log with flush after write.

---

```
--debug-no-sync
```

- **Description:**

```
debug-no-sync[=#]
```

Disables system sync calls. Only for running tests or debugging!

---

```
--debug-sync-timeout
```

- **Description:**

```
debug-sync-timeout[=#]
```

Enable the debug sync facility and optionally specify a default wait timeout in seconds. A zero value keeps the facility disabled.

---

```
--gdb
```

- **Description:** Set up signals usable for debugging.

---

```
--silent-startup
```

- **Description:** Don't print Notes to the [error log](#) during startup.

- **Introduced:** [MariaDB 10.1.8](#)

---

```
--sync-sys
```

- **Description:** Enable/disable system sync calls. Syncs should only be turned off (

```
--disable-sync-sys
```

) when running tests or debugging! Replaced by [debug-no-sync](#) from [MariaDB 5.5](#).

- **Removed:** [MariaDB 5.5](#)

---

```
--thread-alarm
```

- **Description:** Enable/disable system thread alarm calls. Should only be turned off (  
    --disable-thread-alarm  
    ) when running tests or debugging!
- 

## Debugging Options and System Variables

- [core-file](#)
- [debug](#)
- [debug-no-thread-alarm](#)

## Other Options

```
--allow-suspicious-udfs
```

- **Commandline:**

```
--allow-suspicious-udfs
```

- **Description:** Allows use of [user-defined functions](#) consisting of only one symbol

```
x()  
without corresponding  
x_init()  
or  
x_deinit()  
. That also means that one can load any function from any library, for example  
exit()  
from  
libc.so  
. Not recommended unless you require old UDF's with one symbol that cannot be recompiled
```

---

```
--bootstrap
```

- **Commandline:**

```
--bootstrap
```

- **Description:** Used by mysql installation scripts, such as [mysql\\_install\\_db](#) to execute SQL scripts before any privilege or system tables exist. Do no use while an existing MariaDB instance is running.
- 

```
--chroot
```

- **Commandline:**

```
--chroot=name
```

- **Description:** Chroot mysqld daemon during startup.
- 

```
--des-key-file
```

- **Commandline:**

```
--des-key-file=name
```

- **Description:** Load keys for [des\\_encrypt\(\)](#) and des\_encrypt from given file.
- 

```
--exit-info
```

- **Commandline:**

```
--exit-info[=#]
```

- **Description:** Used for debugging. Use at your own risk.
- 

```
--getopt-prefix-matching
```

- **Commandline:**

```
--getopt-prefix-matching={0|1}
```

- **Description:** Makes it possible to disable historical "unambiguous prefix" matching in the command-line option parsing.

- **Default:** TRUE

- **Introduced:** MariaDB 10.1.3
- 

```
--help
```

- **Commandline:**

```
--help
```

- **Description:** Displays help with many commandline options described, and exits.
- 

```
--log-ddl-recovery
```

- **Commandline:**

```
--log-ddl-recovery=name
```

- **Description:** Path to file used for recovery of DDL statements after a crash.

- **Default Value:**

```
ddl-recover.log
```

- **Introduced:** MariaDB 10.6.1
- 

```
--log-short-format
```

- **Commandline:**

```
--log-short-format
```

- **Description:** Don't log extra information to update and slow-query logs.
- 

```
--log-slow-file
```

- **Commandline:**

```
--log-slow-file=name
```

- **Description:** Log slow queries to given log file. Defaults logging to hostname-slow.log
- 

```
--log-slow-time
```

- **Commandline:**

```
--log-slow-time=#
```

- **Description:** Log all queries that have taken more than long-query-time seconds to execute to the slow query log, if active. The argument will be treated as a decimal value with microsecond precision.
-

--log-tc

- **Commandline:**

--log-tc=name

- **Description:** Defines the path to the memory-mapped file-based transaction coordinator log, which is only used if the [binary log](#) is disabled. If you have two or more XA-capable storage engines enabled, then a transaction coordinator log must be available. See [Transaction Coordinator Log](#) for more information. Also see the the

[log\\_tc\\_size](#)

system variable and the

[--tc-heuristic-recover](#)

option.

- **Default Value:**

tc.log

---

--master-connect-retry

- **Commandline:**

--master-connect-retry=#

- **Description:** Deprecated in 5.1.17 and removed in 5.5. The number of seconds the slave thread will sleep before retrying to connect to the master, in case the master goes down or the connection is lost.
- 

--memlock

- **Commandline:**

--memlock

- **Description:** Lock mysqld in memory.
- 

--ndb-use-copying-alter-table

- **Commandline:**

--ndb-use-copying-alter-table

- **Description:** Force ndbcluster to always copy tables at alter table (should only be used if on-line alter table fails).
- 

--one-thread

- **Commandline:**

--one-thread

- **Description:** (Deprecated): Only use one thread (for debugging under Linux). Use [thread-handling=no-threads](#) instead.
  - **Removed:** [MariaDB 10.0.4](#)
- 

--plugin-load

- **Commandline:**

--plugin-load=name

- **Description:** This option can be used to configure the server to load specific [plugins](#). This option uses the following format:

- Plugins can be specified in the format

```
name=library
, where
name
is the plugin name and
library
is the plugin library. This format installs a single plugin from the given plugin library.
```

- Plugins can also be specified in the format

```
library
, where
library
is the plugin library. This format installs all plugins from the given plugin library.
```

- Multiple plugins can be specified by separating them with semicolons.

- Special care must be taken when specifying the

[--plugin-load](#)

option multiple times, or when specifying both the

[--plugin-load](#)

option and the

[--plugin-load-add](#)

option together. The

[--plugin-load](#)

option resets the plugin load list, and this can cause unexpected problems if you are not aware. The

[--plugin-load-add](#)

option does **not** reset the plugin load list, so it is much safer to use. See [Plugin Overview: Specifying Multiple Plugin Load Options](#) for more information.

- See [Plugin Overview: Installing a Plugin with Plugin Load Options](#) for more information.

[--plugin-load-add](#)

- **Commandline:**

[--plugin-load-add=name](#)

- **Description:** This option can be used to configure the server to load specific [plugins](#). This option uses the following format:

- Plugins can be specified in the format

```
name=library
, where
name
is the plugin name and
library
is the plugin library. This format installs a single plugin from the given plugin library.
```

- Plugins can also be specified in the format

```
library
, where
library
is the plugin library. This format installs all plugins from the given plugin library.
```

- Multiple plugins can be specified by separating them with semicolons.

- Special care must be taken when specifying both the

[--plugin-load](#)

option and the

[--plugin-load-add](#)

option together. The

[--plugin-load](#)

option resets the plugin load list, and this can cause unexpected problems if you are not aware. The

```
--plugin-load-add
```

option does **not** reset the plugin load list, so it is much safer to use. See [Plugin Overview: Specifying Multiple Plugin Load Options](#) for more information.

- See [Plugin Overview: Installing a Plugin with Plugin Load Options](#) for more information.
  - **Introduced:** MariaDB 10.0.1
- 

```
--port-open-timeout
```

- **Commandline:**

```
--port-open-timeout=#
```

- **Description:** Maximum time in seconds to wait for the port to become free. (Default: No wait).
- 

```
--safe-user-create
```

- **Commandline:**

```
--safe-user-create
```

- **Description:** Don't allow new user creation by the user who has no write privileges to the `mysql.user` table.
- 

```
--safemalloc-mem-limit
```

- **Commandline:**

```
--safemalloc-mem-limit=#
```

- **Description:** Simulate memory shortage when compiled with the

```
--
```

```
with-debug=full  
option.
```

```
--show-slave-auth-info
```

- **Commandline:**

```
--show-slave-auth-info
```

- **Description:** Show user and password in SHOW SLAVE HOSTS on this master.
- 

```
--skip-grant-tables
```

- **Commandline:**

```
--skip-grant-tables
```

- **Description:** Start without grant tables. This gives all users FULL ACCESS to all tables, which is useful in case of a lost root password. Use `mysqladmin flush-privileges` , `mysqladmin reload` or `FLUSH PRIVILEGES` to resume using the grant tables.
- 

```
--skip-host-cache
```

- **Commandline:**

```
--skip-host-cache
```

- **Description:** Don't cache host names.
- 

```
--skip-partition
```

- **Commandline:**

```
--skip-partition  
,  
--disable-partition
```

- **Description:** Disables user-defined [partitioning](#). Previously partitioned tables cannot be accessed or modified. Tables can still be seen with [SHOW TABLES](#) or by viewing the [INFORMATION\\_SCHEMA.TABLES](#) table. Tables can be dropped with [DROP TABLE](#), but this only removes .frm files, not the associated .par files, which will need to be removed manually.
- 

```
--skip-slave-start
```

- **Commandline:**

```
--skip-slave-start
```

- **Description:** If set, slave is not autostarted.
- 

```
--skip-ssl
```

- **Commandline:**

```
--skip-ssl
```

- **Description:** Disable [TLS connections](#).
- 

```
--skip-symlink
```

- **Commandline:**

```
--skip-symlink
```

- **Description:** Don't allow symlinking of tables. Deprecated and removed in [MariaDB 5.5](#). Use [symbolic-links](#) with the skip option prefix instead.

- **Removed:** [MariaDB 5.5](#)
- 

```
--skip-thread-priority
```

- **Commandline:**

```
--skip-thread-priority
```

- **Description:** Don't give threads different priorities. Deprecated and removed in [MariaDB 10.0](#).
  - **Removed:** [MariaDB 10.0](#)
- 

```
--sql-bin-update-same
```

- **Commandline:**

```
--sql-bin-update-same=#
```

- **Description:** The update log was deprecated in version 5.0 and replaced by the [binary log](#), so this option did nothing since then. Deprecated and removed in [MariaDB 5.5](#).
  - **Removed:** [MariaDB 5.5](#)
- 

--ssl

- **Commandline:**

--ssl

- **Description:** Enable [TLS for connection](#) (automatically enabled with other flags). Disable with '

--

skip-ssl

'

---

--stack-trace

- **Commandline:**

--stack-trace

,

--skip-stack-trace

- **Description:** Print a stack trace on failure. Enabled by default, disable with

-skip-stack-trace

---

--symbolic-links

- **Commandline:**

--symbolic-links

- **Description:** Enables symbolic link support. When set, the [have\\_symlink](#) system variable shows as

YES

. Silently ignored in Windows. Use

--skip-symbolic-links

to disable.

---

--tc-heuristic-recover

- **Commandline:**

--tc-heuristic-recover=name

- **Description:** If [manual heuristic recovery](#) is needed, this option defines the decision to use in the heuristic recovery process. Manual heuristic recovery may be needed if the [transaction coordination log](#) is missing or if it doesn't contain all prepared transactions. This option can be set to

OFF

,

COMMIT

, or

ROLLBACK

. The default is

OFF

. See also the

--log-tc

server option and the

[log\\_tc\\_size](#)

system variable.

---

--temp-pool

- **Commandline:**

--temp-pool

- **Description:** Using this option will cause most temporary files created to use a small set of names, rather than a unique name for each new file. Defaults to

1  
until [MariaDB 10.5.6](#) , use  
--skip-temp-pool  
to disable. Deprecated and defaults to  
0  
from [MariaDB 10.5.7](#) , as benchmarking shows it causes a heavy mutex contention.

---

--test-expect-abort

- **Commandline:**

--test-expect-abort

- **Description:** Expect that server aborts with 'abort'; Don't write out server variables on 'abort'. Useful only for test scripts.
- 

--test-ignore-wrong-options

- **Commandline:**

--test-ignore-wrong-options

- **Description:** Ignore wrong enums values in command line arguments. Useful only for test scripts.
- 

--user

- **Commandline:**

--user=name

- **Description:** Run mysqld daemon as user.
- 

--verbose

- **Commandline:**

-v  
,  
--verbose

- **Description:** Used with [help](#) option for detailed help.
- 

## Other Options and System Variables

- [allow-suspicious-udfs](#)
- [automatic-sp-privileges](#)
- [back-log](#)
- [basedir](#)
- [check-constraint-checks](#)

- column-compression-threshold
- column-compression-zlib-level
- column-compression-zlib-strategy
- column-compression-zlib-wrap
- completion-type
- connect-timeout
- datadir
- date-format
- datetime-format
- deadlock-search-depth-long
- deadlock-search-depth-short
- deadlock-timeout-long
- deadlock-timeout-short
- default-password-lifetime
- default-regex-flags
- default-storage-engine
- default-table-type
- delay-key-write
- disconnect-on-expired-password
- div-precision-increment
- enable-named-pipe
- encrypt-binlog
- encrypt-tmp-disk-tables
- encrypt-tmp-files
- encryption-algorithm
- engine-condition-pushdown
- eq-range-index-dive-limit
- event-scheduler
- expire-logs-days
- explicit-defaults-for-timestamp
- extra-max-connections
- extra-port
- flush
- flush-time
- ft-boolean-syntax
- ft-max-word-len
- ft-min-word-len
- ft-query-expansion-limit
- ft-stopword-file
- general-log
- general-log-file
- group-concat-max-len
- histogram-size
- histogram-type
- host-cache-size
- idle-readonly-transaction-timeout
- idle-transaction-timeout
- idle-write-transaction-timeout
- ignore-db dirs
- in-predicate-conversion-threshold
- init-connect
- init-file
- interactive-timeout
- large-pages
- local-infile
- lock-wait-timeout
- log
- log-disabled-statements
- log-error
- log-output
- log-queries-not-using-indexes
- log-slow-admin-statements
- log-slow-disabled-statements
- log-slow-filter
- log-slow-queries
- log-slow-rate-limit
- log-slow-verbosity
- log-tc-size
- log-warnings
- long-query-time
- low-priority-updates
- lower-case-table-names

- max-allowed-packet
- max-connections
- max-connect-errors
- max-delayed-threads
- max-digest-length
- max-error-count
- max-length-for-sort-data
- max-long-data-size
- max-password-errors
- max-prepared-stmt-count
- max-recursive-iterations
- max-rowid-filter-size
- max-session-mem-used
- max-sp-recursion-depth
- max-statement-time
- max-tmp-tables
- max-user-connections
- max-write-lock-count
- metadata-locks-cache-size
- metadata-locks-hash-instances
- min-examined-row-limit
- mrr-buffer-size
- multi-range-count
- --mysql56-temporal-format
- net-buffer-length
- net-read-timeout
- net-retry-count
- net-write-timeout
- open-files-limit
- pid-file
- plugin-dir
- plugin-maturity
- port
- preload-buffer-size
- profiling-history-size
- progress-report-time
- proxy-protocol-networks
- query-cache-limit
- query-cache-min-res-unit
- query-cache-strip-comments
- query-cache-wlock-invalidate
- read-rnd-buffer-size
- read-only
- require-secure-transport
- safe-show-database
- secure-auth
- secure-file-priv
- secure-timestamp
- session-track-schema
- session-track-state-change
- session-track-system-variables
- session-track-transaction-info
- skip-automatic-sp-privileges
- skip-external-locking
- skip-large-pages
- skip-log-error
- skip-name-resolve
- skip-networking
- skip-show-database
- slow-launch-time
- slow-query-log
- slow-query-log-file
- socket
- sort-buffer-size
- sql-if-exists
- sql-mode
- ssl-ca
- ssl-capath
- ssl-cert
- ssl-cipher
- ssl-crl
- ssl-cripath

- `ssl-key`
- `standards_compliant_cte`
- `stored-program-cache`
- `strict_password_validation`
- `sync-frm`
- `system-versioning-alter-history`
- `system-versioning-asof`
- `system-versioning-innodb-algorithm-simple`
- `table-lock-wait-timeout`
- `tcp-keepalive-interval`
- `tcp-keepalive-probes`
- `tcp-keepalive-time`
- `tcp-nodelay`
- `thread-cache-size`
- `thread-concurrency`
- `thread-handling`
- `thread-pool-dedicated-listener`
- `thread-pool-exact-stats`
- `thread-pool-idle-timeout`
- `thread-pool-max-threads`
- `thread-pool-min-threads`
- `thread-pool-oversubscribe`
- `thread-pool-prio-kickup-timer`
- `thread-pool-priority`
- `thread-pool-size`
- `thread-pool-stall-limit`
- `thread-stack`
- `timed-mutexes`
- `time-format`
- `tls-version`
- `tmpdir`
- `transaction-isolation`
- `transaction-alloc-block-size`
- `transaction-prealloc-size`
- `transaction-read-only`
- `updatable-views-with-limit`
- `userstat`
- `version`
- `wait-timeout`

## Authentication Plugins - Options and System Variables

### Authentication Plugin - `ed25519`

The options related to the

`ed25519`

authentication plugin can be found [here](#).

### Authentication Plugin - `gssapi`

The system variables related to the

`gssapi`

authentication plugin can be found [here](#).

The options related to the

`gssapi`

authentication plugin can be found [here](#).

## Authentication Plugin - named\_pipe

The options related to the

[named\\_pipe](#)

authentication plugin can be found [here](#).

## Authentication Plugin - pam

The system variables related to the

[pam](#)

authentication plugin can be found [here](#).

The options related to the

[pam](#)

authentication plugin can be found [here](#).

## Authentication Plugin - unix\_socket

The options related to the

[unix\\_socket](#)

authentication plugin can be found [here](#).

# Encryption Plugins - Options and System Variables

## Encryption Plugin - aws\_key\_management

The system variables related to the

[aws\\_key\\_management](#)

encryption plugin can be found [here](#).

The options related to the

[aws\\_key\\_management](#)

encryption plugin can be found [here](#).

## Encryption Plugin - file\_key\_management

The system variables related to the

[file\\_key\\_management](#)

encryption plugin can be found [here](#).

The options related to the

`file_key_management`

encryption plugin can be found [here](#) .

## Password Validation Plugins - Options and System Variables

### Password Validation Plugin - `simple_password_check`

The system variables related to the

`simple_password_check`

password validation plugin can be found [here](#) .

The options related to the

`simple_password_check`

password validation plugin can be found [here](#) .

### Password Validation Plugin - `cracklib_password_check`

The system variables related to the

`cracklib_password_check`

password validation plugin can be found [here](#) .

The options related to the

`cracklib_password_check`

password validation plugin can be found [here](#) .

## Audit Plugins - Options and System Variables

### Audit Plugin - `server_audit`

Options and system variables related to the

`server_audit`

audit plugin can be found [here](#) .

### Audit Plugin - `SQL_ERROR_LOG`

The system variables related to the

`SQL_ERROR_LOG`

audit plugin can be found [here](#) .

The options related to the

`SQL_ERROR_LOG`

audit plugin can be found [here](#) .

## Audit Plugin - QUERY\_RESPONSE\_TIME\_AUDIT

The options related to the

[QUERY\\_RESPONSE\\_TIME\\_AUDIT](#)

audit plugin can be found [here](#).

## Daemon Plugins - Options and System Variables

### Daemon Plugin - handlersocket

The options for the HandlerSocket plugin are all described on the [HandlerSocket Configuration Option](#) page.

## Information Schema Plugins - Options and System Variables

### Information Schema Plugin - DISKS

The options related to the

[DISKS](#)

information schema plugin can be found [here](#).

### Information Schema Plugin - feedback

The system variables related to the

[feedback](#)

plugin can be found [here](#).

The options related to the

[feedback](#)

plugin can be found [here](#).

### Information Schema Plugin - LOCALES

The options related to the

[LOCALES](#)

information schema plugin can be found [here](#).

### Information Schema Plugin - METADATA\_LOCK\_INFO

The options related to the

[METADATA\\_LOCK\\_INFO](#)

information schema plugin can be found [here](#).

## Information Schema Plugin - QUERY\_CACHE\_INFO

The options related to the

[QUERY\\_CACHE\\_INFO](#)

information schema plugin can be found [here](#).

## Information Schema Plugin - QUERY\_RESPONSE\_TIME

The system variables related to the

[QUERY\\_RESPONSE\\_TIME](#)

information schema plugin can be found [here](#).

The options related to the

[QUERY\\_RESPONSE\\_TIME](#)

information schema plugin can be found [here](#).

## Information Schema Plugin - user\_variables

The options related to the

[user\\_variables](#)

information schema plugin can be found [here](#).

## Information Schema Plugin - WSREP\_MEMBERSHIP

The options related to the

[WSREP\\_MEMBERSHIP](#)

information schema plugin can be found [here](#).

## Information Schema Plugin - WSREP\_STATUS

The options related to the

[WSREP\\_STATUS](#)

information schema plugin can be found [here](#).

# Replication Plugins - Options and System Variables

## Replication Plugin - rpl\_semi\_sync\_master

The system variables related to the

[rpl\\_semi\\_sync\\_master](#)

replication plugin can be found [here](#).

The options related to the

[rpl\\_semi\\_sync\\_master](#)

replication plugin can be found [here](#).

## Replication Plugin - [rpl\\_semi\\_sync\\_slave](#)

The system variables related to the

[rpl\\_semi\\_sync\\_slave](#)

replication plugin can be found [here](#).

The options related to the

[rpl\\_semi\\_sync\\_slave](#)

replication plugin can be found [here](#).

## Default Values

You can verify the default values for an option by doing:

```
mysqld --no-defaults --help --verbose
```

## 2.1.6.5 What to Do if MariaDB Doesn't Start

### Contents

1. [The Error Log and the Data Directory](#)
2. [Option Files](#)
  1. [Invalid Option or Option Value](#)
3. [Can't Open Privilege Tables](#)
4. [Can't Create Test File](#)
5. [InnoDB](#)
  1. [Cannot Allocate Memory for the InnoDB Buffer Pool](#)
  2. [InnoDB Table Corruption](#)
6. [MyISAM](#)
7. [systemd](#)
8. [SELinux](#)

There could be many reasons that MariaDB fails to start. This page will help troubleshoot some of the more common reasons and provide solutions.

If you have tried everything here, and still need help, you can ask for help on IRC or on the forums - see [Where to find other MariaDB users and developers](#) - or ask a question at the [Starting and Stopping MariaDB](#) page.

## The Error Log and the Data Directory

The reason for the failure will almost certainly be written in the [error log](#) and, if you are starting MariaDB manually, to the console. By default, the error log is named *host-name*.err and is written to the data directory.

Common Locations:

- /var/log/
- /var/log/mysql
- C:\Program Files\MariaDB x.y\data (x.y refers to the version number)
- C:\Program Files (x86)\MariaDB x.y\data (32bit version on 64bit Windows)

It's also possible that the error log has been explicitly written to another location. This is often done by changing the

[datadir](#)

or

## log\_error

system variables in an [option file](#). See [Option Files](#) below for more information about that.

A quick way to get the values of these system variables is to execute the following commands:

```
mysqld --help --verbose | grep 'log-error' | tail -1  
mysqld --help --verbose | grep 'datadir' | tail -1
```

## Option Files

Another kind of file to consider when troubleshooting is [option files](#). The default option file is called

`my.cnf`

. Option files contain configuration options, such as the location of the data directory mentioned above. If you're unsure where the option file is located, see [Configuring MariaDB with Option Files: Default Option File Locations](#) for information on the default locations.

You can check which configuration options MariaDB server will use from its option files by executing the following command:

```
mysqld --print-defaults
```

You can also check by executing the following command:

```
my_print_defaults --mysqld
```

See [Configuring MariaDB with Option Files: Checking Program Options](#) for more information on checking configuration options.

## Invalid Option or Option Value

Another potential reason for a startup failure is that an [option file](#) contains an invalid option or an invalid option value. In those cases, the [error log](#) should contain an error similar to this:

```
140514 12:19:37 [ERROR] /usr/local/mysql/bin/mysqld: unknown variable 'option=value'
```

This is more likely to happen when you upgrade to a new version of MariaDB. In most cases the [option file](#) from the old version of MariaDB will work just fine with the new version. However, occasionally, options are removed in new versions of MariaDB, or the valid values for options are changed in new versions of MariaDB. Therefore, it's possible for an [option file](#) to stop working after an upgrade.

Also remember that option names are case sensitive.

Examine the specifics of the error. Possible fixes are usually one of the following:

- If the option is completely invalid, then remove it from the [option file](#).
- If the option's name has changed, then fix the name.
- If the option's valid values have changed, then change the option's value to a valid one.
- If the problem is caused by a simple typo, then fix the typo.

## Can't Open Privilege Tables

It is possible to see errors similar to the following:

```
System error 1067 has occurred.  
Fatal error: Can't open privilege tables: Table 'mysql.host' doesn't exist
```

If errors like this occur, then critical [system tables](#) are either missing or are in the wrong location. The above error is quite common after an upgrade if the [option files](#) set the

`basedir`

or

`datadir`

to a non-standard location, but the new server is using the default location. Therefore, make sure that the

`basedir`

and

`datadir`

variables are correctly set.

If you're unsure where the option file is located, see [Configuring MariaDB with Option Files: Default Option File Locations](#) for information on the default locations.

If the `system tables` really do not exist, then you may need to create them with

```
mysql_install_db
```

. See [Installing System Tables \(mysql\\_install\\_db\)](#) for more information.

## Can't Create Test File

One of the first tests on startup is to check whether MariaDB can write to the data directory. When this fails, it will log an error like this:

```
May 13 10:24:28 mariadb3 mysqld[19221]: 2019-05-13 10:24:28 0 [Warning] Can't create test file /usr/local/data/mariadb/mariadb3.  
May 13 10:24:28 mariadb3 mysqld[19221]: 2019-05-13 10:24:28 0 [ERROR] Aborting
```

This is usually a permission error on the directory in which this file is being written. Ensure that the entire

```
datadir
```

is owned by the user running

```
mysqld
```

, usually

```
mysql
```

. Ensure that directories have the "x" (execute) directory permissions for the owner. Ensure that all the parent directories of the

```
datadir
```

upwards have "x" (execute) permissions for all (

user

,

group

, and

other

).

Once this is checked look at the [systemd](#) and [selinux](#) documentation below, or [apparmor](#).

## InnoDB

[InnoDB](#) is probably the MariaDB component that most frequently causes a crash. In the error log, lines containing InnoDB messages generally start with "InnoDB:".

### Cannot Allocate Memory for the InnoDB Buffer Pool

In a typical installation on a dedicated server, at least 70% of your memory should be assigned to [InnoDB buffer pool](#); sometimes it can even reach 85%. But be very careful: don't assign to the buffer pool more memory than it can allocate. If it cannot allocate memory, InnoDB will use the disk's swap area, which is very bad for performance. If swapping is disabled or the swap area is not big enough, InnoDB will crash. In this case, MariaDB will probably try to restart several times, and each time it will log a message like this:

```
140124 17:29:01 InnoDB: Fatal error: cannot allocate memory for the buffer pool
```

In that case, you will need to add more memory to your server/VM or decrease the value of the `innodb_buffer_pool_size` variables.

Remember that the buffer pool will slightly exceed that limit. Also, remember that MariaDB also needs allocate memory for other storage engines and several per-connection buffers. The operating system also needs memory.

### InnoDB Table Corruption

By default, InnoDB deliberately crashes the server when it detects table corruption. The reason for this behavior is preventing corruption propagation. However, in some situations, server availability is more important than data integrity. For this reason, we can avoid these crashes by changing the value of `innodb_corrupt_table_action` to 'warn'.

If InnoDB crashes the server after detecting data corruption, it writes a detailed message in the error log. The first lines are similar to the following:

```
InnoDB: Database page corruption on disk or a failed  
InnoDB: file read of page 7.  
InnoDB: You may have to recover from a backup.
```

Generally, it is still possible to recover most of the corrupted data. To do so, restart the server in [InnoDB recovery mode](#) and try to extract the data that you want to backup. You can save them in a CSV file or in a non-InnoDB table. Then, restart the server in normal mode and restore the data.

## MyISAM

Most tables in the `mysql` database are MyISAM tables. These tables are necessary for MariaDB to properly work, or even start.

A MariaDB crash could cause system tables corruption. With the default settings, MariaDB will simply not start if the system tables are corrupted. With `myisam_recover_options`, we can force MyISAM to repair damaged tables.

## systemd

If you are using

```
systemd
```

, then there are a few relevant notes about startup failures:

- If MariaDB is configured to access files under

```
/home  
,  
/root  
, or  
/run/user
```

, then the default systemd unit file will prevent access to these directories with a `Permission Denied` error. This happens because the unit file set

```
ProtectHome=true
```

. See [Systemd: Configuring Access to Home Directories](#) for information on how to work around this.

- The default systemd unit file also sets `ProtectSystem=full`, which places restrictions on writing to a few other directories. Overwriting this with `ProtectSystem=off` in the same way as above will restore access to these directories.
- If MariaDB takes longer than 90 seconds to start, then the default systemd unit file will cause it to fail with an error. This happens because the default value for the

```
TimeoutStartSec
```

option is 90 seconds. See [Systemd: Configuring the Systemd Service Timeout](#) for information on how to work around this.

- The systemd journal may also contain useful information about startup failures. See [Systemd: Systemd Journal](#) for more information.

See [systemd](#) documentation for further information on systemd configuration.

## SELinux

[Security-Enhanced Linux \(SELinux\)](#) is a Linux kernel module that provides a framework for configuring [mandatory access control \(MAC\)](#) system for many resources on the system. It is enabled by default on some Linux distributions, including RHEL, CentOS, Fedora, and other similar Linux distribution. SELinux prevents programs from accessing files, directories or ports unless it is configured to access those resources.

You might need to troubleshoot SELinux-related issues in cases, such as:

- MariaDB is using a non-default port.
- MariaDB is reading from or writing to some files (datadir, log files, option files, etc.) located at non-default paths.
- MariaDB is using a plugin that requires access to resources that default installations do not use.

Setting SELinux state to

```
permissive
```

is a common way to investigate what is going wrong while allowing MariaDB to function normally.

```
permissive
```

is supposed to produce a log entry every time it should block a resource access, without actually blocking it. However, [there are situations](#) when SELinux blocks resource accesses even in

```
permissive
```

mode.

See [SELinux](#) for more information.

## 2.1.6.6 Running MariaDB from the Build Directory

You can run mysqld directly from the build directory (without doing

```
make install  
).
```

### Starting mariadb After Build on Windows

On Windows, the data directory is produced during the build.

The simplest way to start database from the command line is:

1. Go to the directory where mariadb.exe is located (subdirectory sql\Debug or sql\Relwithdebinfo of the build directory)
2. From here, execute, if you are using [MariaDB 10.5](#) or newer,

```
mariadb.exe --console
```

```
else
```

```
mysqld.exe --console
```

As usual, you can pass other server parameters on the command line, or store them in a my.ini configuration file and pass

```
--defaults-file=path\to\my.ini
```

The default search path on Windows for the my.ini file is:

- GetSystemWindowsDirectory()
- GetWindowsDirectory()
- C:\
- Directory where the executable is located

### Starting mysqld After Build on Unix

Copy the following to your '

```
~/my.cnf  
' file.
```

There are two lines you have to edit: '

```
datadir=  
' and '  
language=  
'.
```

Be sure to change them to match your environment.

```

# Example Mariadb config file.
# You can copy this to one of:
# /etc/my.cnf to set global options,
# /mysql-data-dir/my.cnf to get server specific options or
# ~/my.cnf for user specific options.
#
# One can use all long options that the program supports.
# Run the program with --help to get a list of available options

# This will be passed to all MariaDB clients
[client]
#password=my_password
#port=3306
#socket=/tmp/mysql.sock

# Here is entries for some specific programs
# The following values assume you have at Least 32M ram

# The mariadb server (both [mysqld] and [mariadb] works here)
[mariadb]
#port=3306
#socket=/tmp/mysql.sock

# The following three entries caused mysqld 10.0.1-MariaDB (and possibly other versions) to abort...
# skip-locking
# set-variable = key_buffer=16M

loose-innodb_data_file_path = ibdata1:1000M
loose-mutex-deadlock-detector
gdb

##### Fix the two following paths

# Where you want to have your database
datadir=/path/to/data/dir

# Where you have your mysql/MariaDB source + sql/share/english
language=/path/to/src/dir/sql/share/english

##### One can also have a different path for different versions, to simplify development.

[mariadb-10.1]
lc-messages-dir=/my/maria-10.1/sql/share

[mariadb-10.2]
lc-messages-dir=/my/maria-10.2/sql/share

[mysqldump]
quick
set-variable = max_allowed_packet=16M

[mysql]
no-auto-rehash

[myisamchk]
set-variable= key_buffer=128M

```

With the above file in place, go to your MariaDB source directory and execute:

```
./scripts/mysql_install_db --srcdir=$PWD --datadir=/path/to/data/dir --user=$LOGNAME
```

Above '\$PWD' is the environment variable that points to your current directory. If you added

```
datadir
to your
my.cnf
, you don't have to give this option above. Also above, --user=$LOGNAME is necessary when using msqyld 10.0.1-MariaDB (and possibly other
versions)
```

Now you can start

```
mariadb
(or
```

```
mysqld
if you are using a version older than MariaDB 10.5 ) in the debugger:
```

```
cd sql  
ddd ./mariadb &
```

Or start mysqld on its own:

```
cd sql  
.mariadb &
```

After starting up

```
mariadb  
using one of the above methods (with the debugger or without), launch the client (as root if you don't have any users setup yet).
```

```
./client/mysql
```

## Using a Storage Engine Plugin

The simplest case is to compile the storage engine into MariaDB:

```
cmake -DWITH_PLUGIN_<plugin_name>=1` .
```

Another option is to point

```
mariadb  
to the storage engine directory:
```

```
./mariadb --plugin-dir={build-dir-path}/storage/connect/.libs
```

## 2.1.6.7 mysql.server

### Contents

- 1. [Using mysql.server](#)
  - 1. [Options](#)
  - 2. [Option Files](#)
    - 1. [Option Groups](#)
    - 3. [Customizing mysql.server](#)
- 2. [Installed Locations](#)
  - 1. [Installed SysVinit Locations](#)
    - 1. [Manually Installing with SysVinit](#)

The

[mysql.server](#)

startup script is in MariaDB distributions on Linux and Unix. It is a wrapper that works as a standard [sysVinit](#) script. However, it can be used independently of [sysVinit](#) as a regular

```
sh  
script. The script starts the
```

[mysqld](#)

server process by first changing its current working directory to the MariaDB install directory and then starting

[mysqld\\_safe](#)

. The script requires the standard [sysVinit](#) arguments, such as

```
start  
,  
stop  
,  
restart  
, and  
status  
. For example:
```

```
mysql.server start  
mysql.server restart  
mysql.server stop  
mysql.server status
```

It can be used on systems such as Linux, Solaris, and Mac OS X.

The

```
mysql.server  
script starts
```

[mysqld](#)

by first changing to the MariaDB install directory and then calling

[mysqld\\_safe](#)

## Using mysql.server

The command to use

```
mysql.server
```

and the general syntax is:

```
mysql.server [ start | stop | restart | status ] <options> <mysqld_options>
```

## Options

If an unknown option is provided to

[mysqld\\_safe](#)

on the command-line, then it is passed to

[mysqld\\_safe](#)

[mysql.server](#)

supports the following options:

| Option                              | Description                                                                                                                                                                                                                                                                                   |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| --basedir=path                      | The path to the MariaDB installation directory.                                                                                                                                                                                                                                               |
| --datadir=path                      | The path to the MariaDB data directory.                                                                                                                                                                                                                                                       |
| --pid-file=file_name                | The path name of the file in which the server should write its process ID. If not provided, the default, <code>host_name.pid</code> is used.                                                                                                                                                  |
| --service-startup-timeout=file_name | How long in seconds to wait for confirmation of server startup. If the server does not start within this time, <code>mysql.server</code> exits with an error. The default value is 900. A value of 0 means not to wait at all for startup. Negative values mean to wait forever (no timeout). |
| --use-mysqld_safe                   | Use <a href="#">mysqld_safe</a> to start the server. This is the default.                                                                                                                                                                                                                     |
| --use-manager                       | Use Instance Manager to start the server.                                                                                                                                                                                                                                                     |

|                      |                                                               |
|----------------------|---------------------------------------------------------------|
| --<br>user=user_name | The login user name to use for running<br><code>mysqld</code> |
|----------------------|---------------------------------------------------------------|

## Option Files

In addition to reading options from the command-line,

`mysql.server`  
can also read options from [option files](#).

The following options relate to how MariaDB command-line tools handles option files. They must be given as the first argument on the command-line:

| Option                               | Description                                      |
|--------------------------------------|--------------------------------------------------|
| <code>--print-defaults</code>        | Print the program argument list and exit.        |
| <code>--no-defaults</code>           | Don't read default options from any option file. |
| <code>--defaults-file=#</code>       | Only read default options from the given file #. |
| <code>--defaults-extra-file=#</code> | Read this file after the global files are read.  |

## Option Groups

`mysql.server`

reads options from the following [option groups](#) from [option files](#):

| Group                       | Description                                                                                         |
|-----------------------------|-----------------------------------------------------------------------------------------------------|
| <code>[mysql.server]</code> | Options read by<br><code>mysql.server</code> , which includes both MariaDB Server and MySQL Server. |

`mysql.server`

also reads options from the following server [option groups](#) from [option files](#):

| Group                      | Description                                                                                                                                                   |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>[mysqld]</code>      | Options read by<br><code>mysqld</code> , which includes both MariaDB Server and MySQL Server.                                                                 |
| <code>[server]</code>      | Options read by MariaDB Server.                                                                                                                               |
| <code>[mysqld-X.Y]</code>  | Options read by a specific version of<br><code>mysqld</code> , which includes both MariaDB Server and MySQL Server. For example,<br><code>[mysqld-5.5]</code> |
| <code>[mariadb]</code>     | Options read by MariaDB Server.                                                                                                                               |
| <code>[mariadb-X.Y]</code> | Options read by a specific version of MariaDB Server.                                                                                                         |

|                 |                                                                                                                                                                                          |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [client-server] | Options read by all MariaDB <a href="#">client programs</a> and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients. |
| [galera]        | Options read by a galera-capable MariaDB Server. Available on systems compiled with Galera support.                                                                                      |

## Customizing mysql.server

If you have installed MariaDB to a non-standard location, then you may need to edit the

`mysql.server`  
script to get it to work right.

If you do not want to edit the

`mysql.server`  
script itself, then  
`mysql.server`  
also sources a few other  
`sh`

scripts. These files can be used to set any variables that might be needed to make the script work in your specific environment. The files are:

- `/etc/default/mysql`
- `/etc/sysconfig/mysql`
- `/etc/conf.d/mysql`

## Installed Locations

`mysql.server`  
can be found in the  
`support-files`  
directory under your MariaDB installation directory or in a MariaDB source distribution.

## Installed SysVinit Locations

On systems that use [sysVinit](#),

`mysql.server`  
may also be installed in other locations and with other names.

If you installed MariaDB on Linux using [RPMs](#), then the

`mysql.server`  
script will be installed into the  
`/etc/init.d`  
directory with the name  
`mysql`  
. You need not install it manually.

## Manually Installing with SysVinit

If you install MariaDB from [source](#) or from a [binary tarball](#) that does not install

`mysql.server`

automatically, and if you are on a system that uses [sysVinit](#), then you can manually install  
`mysql.server`  
with [sysVinit](#). This is usually done by copying it to  
`/etc/init.d/`  
and then creating specially named symlinks in the appropriate  
`/etc/rcX.d/`  
directories (where 'X' is a number between 0 and 6).

In the examples below we will follow the historical convention of renaming the

`mysql.server`  
script to '  
`mysql`'  
'when we copy it to  
`/etc/init.d/`  
'

The first step for most Linux distributions is to copy the

```
mysql.server  
script to  
/etc/init.d/  
and make it executable:
```

```
cd /path/to/your/mariadb-version/support-files/  
cp mysql.server /etc/init.d/mysql  
chmod +x /etc/init.d/mysql
```

Now all that is needed is to create the specially-named symlinks. On both RPM and Debian-based Linux distributions there are tools which do this for you. Consult your distribution's documentation if neither of these work for you and follow their instructions for generating the symlinks or creating them manually.

On RPM-based distributions (like Fedora and CentOS), you use

```
chkconfig  
:  
:
```

```
chkconfig --add mysql  
chkconfig --level 345 mysql on
```

On Debian-based distributions you use

```
update-rc.d  
:  
:
```

```
update-rc.d mysql defaults
```

On FreeBSD, the location for startup scripts is

```
/usr/local/etc/rc.d/  
and when you copy the  
mysql.server  
script there you should rename it so that it matches the  
*.sh  
pattern, like so:
```

```
cd /path/to/your/mariadb/support-files/  
cp mysql.server /usr/local/etc/rc.d/mysql.server.sh
```

As stated above, consult your distribution's documentation for more information on starting services like MariaDB at system startup.

See [mysqld startup options](#) for information on configuration options for

```
mysqld
```

## 2.1.6.8 mysqld\_safe

MariaDB starting with 10.4.6

From [MariaDB 10.4.6](#) ,  
mariadb-safe  
is a symlink to  
mysqld\_safe  
.

MariaDB starting with 10.5.2

From [MariaDB 10.5.2](#) ,  
mariadb-safe  
is the name of the server, with  
mysqld\_safe  
a symlink .

## Contents

1. [Using mysqld\\_safe](#)
  1. [Options](#)
  2. [Option Files](#)
    1. [Option Groups](#)
  3. [Configuring the Open Files Limit](#)
  4. [Configuring the Core File Size](#)
  5. [Configuring MariaDB to Write the Error Log to Syslog](#)
2. [Specifying mysql](#)
3. [Specifying datadir](#)
4. [Logging](#)
5. [Editing mysqld\\_safe](#)
6. [NetWare](#)
7. [See Also](#)

The

`mysqld_safe`

startup script is in MariaDB distributions on Linux and Unix. It is a wrapper that starts

`mysqld`

with some extra safety features. For example, if

`mysqld_safe`

notices that

`mysqld`

has crashed, then

`mysqld_safe`

will automatically restart

`mysqld`

.

`mysqld_safe`

is the recommended way to start

`mysqld`

on Linux and Unix distributions that do not support

`systemd`

. Additionally, the

`mysql.server`

init script used by

`sysVinit`

starts

`mysqld`

with

`mysqld_safe`

by default.

## Using mysqld\_safe

The command to use

`mysqld_safe`

and the general syntax is:

```
mysqld_safe [ --no-defaults | --defaults-file | --defaults-extra-file | --defaults-group-suffix | --print-defaults ] <options> <
```

## Options

Many of the options supported by

`mysqld_safe`

are identical to options supported by

## `mysqld`

. If an unknown option is provided to  
`mysqld_safe`  
on the command-line, then it is passed to  
`mysqld`

`mysqld_safe`  
supports the following options:

| Option                                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>--help</code>                     | Display a help message and exit.                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>--autoclose</code>                | (NetWare only) On NetWare,<br><code>mysqld_safe</code><br>provides a screen presence. When you unload (shut down) the<br><code>mysqld_safe</code><br>NLM, the screen does not by default go away. Instead, it prompts for user input:<br>NLM has terminated; Press any key to close the screen<br>. If you want NetWare to close the screen automatically instead, use the<br><code>--autoclose</code><br>option to <code>mysqld_safe</code> . |
| <code>--basedir=path</code>             | The path to the MariaDB installation directory.                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>--core-file-size=size</code>      | The size of the core file that <code>mysqld</code> should be able to create. The option value is passed to ulimit -c.                                                                                                                                                                                                                                                                                                                          |
| <code>--crash-script=file</code>        | Script to call in the event of <code>mysqld</code> crashing.                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>--datadir=path</code>             | The path to the data directory.                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>--defaults-extra-file=path</code> | The name of an option file to be read in addition to the usual option files. This must be the first option on the command line if it is used. If the file does not exist or is otherwise inaccessible, the server will exit with an error.                                                                                                                                                                                                     |
| <code>--defaults-file=file_name</code>  | The name of an option file to be read instead of the usual option files. This must be the first option on the command line if it is used.                                                                                                                                                                                                                                                                                                      |
| <code>--defaults-group-suffix=#</code>  | In addition to the default option groups, also read option groups with this suffix.                                                                                                                                                                                                                                                                                                                                                            |
| <code>--flush-caches</code>             | Flush and purge buffers/caches before starting the server.                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>--ledir=path</code>               | If <code>mysqld_safe</code> cannot find the server, use this option to indicate the path name to the directory where the server is located.                                                                                                                                                                                                                                                                                                    |

|                          |                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| --log-error=file_name    | Write the error log to the given file.                                                                                                                                                                                                                                                                                                                                     |
| --malloc-lib=lib         | Preload shared library <i>lib</i> if available. See <a href="#">debugging MariaDB</a> for an example.                                                                                                                                                                                                                                                                      |
| --mysqld=prog_name       | The name of the server program (in the ledir directory) that you want to start. This option is needed if you use the MariaDB binary distribution but have the data directory outside of the binary distribution. If mysqld_safe cannot find the server, use the --ledir option to indicate the path name to the directory where the server is located.                     |
| --mysqld-version=suffix  | This option is similar to the --mysqld option, but you specify only the suffix for the server program name. The basename is assumed to be mysqld. For example, if you use<br>--mysqld-version=debug<br>, mysqld_safe starts the mysqld-debug program in the ledir directory. If the argument to --mysqld-version is empty, mysqld_safe uses mysqld in the ledir directory. |
| --nice=priority          | Use the nice program to set the server's scheduling priority to the given value.                                                                                                                                                                                                                                                                                           |
| --no-auto-restart        | Exit after starting mysqld.                                                                                                                                                                                                                                                                                                                                                |
| --no-defaults            | Do not read any option files. This must be the first option on the command line if it is used.                                                                                                                                                                                                                                                                             |
| --no-watch               | Exit after starting mysqld.                                                                                                                                                                                                                                                                                                                                                |
| --numa-interleave        | Run mysqld with its memory interleaved on all NUMA nodes.                                                                                                                                                                                                                                                                                                                  |
| --open-files-limit-count | The number of files that mysqld should be able to open. The option value is passed to ulimit -n. Note that you need to start mysqld_safe as root for this to work properly.                                                                                                                                                                                                |
| --pid-file=file_name     | The path name of the process ID file.                                                                                                                                                                                                                                                                                                                                      |
| --plugin-dir=dir_name,   | Directory for client-side plugins.                                                                                                                                                                                                                                                                                                                                         |
| --port=port_num          | The port number that the server should use when listening for TCP/IP connections. The port number must be 1024 or higher unless the server is started by the root system user.                                                                                                                                                                                             |
| --print-defaults         | Print the program argument list and exit.                                                                                                                                                                                                                                                                                                                                  |

|                               |                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| --skip-kill-mysqld            | Do not try to kill stray mysqld processes at startup. This option works only on Linux.                                                                                                                                                                                                                                                               |
| --socket=path                 | The Unix socket file that the server should use when listening for local connections.                                                                                                                                                                                                                                                                |
| --syslog,<br>--skip-syslog    | --syslog<br>causes error messages to be sent to syslog on systems that support the logger program.<br>--skip-syslog<br>suppresses the use of syslog; messages are written to an error log file.                                                                                                                                                      |
| --syslog-tag=tag              | For logging to syslog, messages from<br><code>mysqld_safe</code><br>and <code>mysqld</code> are written with a tag of <code>mysqld_safe</code> and <code>mysqld</code> , respectively. To specify a suffix for the tag, use<br>--syslog-tag=tag<br>, which modifies the tags to be<br><code>mysqld_safe-tag</code><br>and<br><code>mysqld-tag</code> |
| --timezone=timezone           | Set the TZ time zone <a href="#">environment variable</a> to the given option value. Consult your operating system documentation for legal time zone specification formats. Also see <a href="#">Time Zones</a> .                                                                                                                                    |
| --user={user_name or user_id} | Run the mysqld server as the user having the name <code>user_name</code> or the numeric user ID <code>user_id</code> . ("User" in this context refers to a system login account, not a MariaDB user listed in the grant tables.)                                                                                                                     |

## Option Files

In addition to reading options from the command-line,

```
mysqld_safe
can also read options from option files. If an unknown option is provided to
mysqld_safe
in an option file, then it is ignored.
```

The following options relate to how MariaDB command-line tools handles option files. They must be given as the **first argument** on the command-line:

| Option                    | Description                                                                         |
|---------------------------|-------------------------------------------------------------------------------------|
| --print-defaults          | Print the program argument list and exit.                                           |
| --no-defaults             | Don't read default options from any option file.                                    |
| --defaults-file=#         | Only read default options from the given file #.                                    |
| --defaults-extra-file=#   | Read this file after the global files are read.                                     |
| --defaults-group-suffix=# | In addition to the default option groups, also read option groups with this suffix. |

## Option Groups

`mysqld_safe`  
reads options from the following [option groups](#) from [option files](#):

| Group          | Description                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------|
| [mysqld_safe]  | Options read by<br>mysqld_safe<br>, which includes both MariaDB Server and MySQL Server.                        |
| [safe_mysqld]  | Options read by<br>mysqld_safe<br>, which includes both MariaDB Server and MySQL Server.                        |
| [mariadb_safe] | Options read by<br>mysqld_safe<br>from MariaDB Server.                                                          |
| [mariadb-safe] | Options read by<br>mysqld_safe<br>from MariaDB Server. Available starting with <a href="#">MariaDB 10.4.6</a> . |

The

[safe\_mysqld]  
option group is primarily supported for backward compatibility. You should rename such option groups to [mysqld\_safe] in MariaDB installations to prevent breakage in the future if this compatibility is removed.

mysqld\_safe  
also reads options from the following server [option groups](#) from [option files](#):

| Group           | Description                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [mysqld]        | Options read by<br>mysqld<br>, which includes both MariaDB Server and MySQL Server.                                                                                                      |
| [server]        | Options read by MariaDB Server.                                                                                                                                                          |
| [mysqld-X.Y]    | Options read by a specific version of<br>mysqld<br>, which includes both MariaDB Server and MySQL Server. For example,<br>[mysqld-5.5]                                                   |
| [mariadb]       | Options read by MariaDB Server.                                                                                                                                                          |
| [mariadb-X.Y]   | Options read by a specific version of MariaDB Server.                                                                                                                                    |
| [client-server] | Options read by all MariaDB <a href="#">client programs</a> and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients. |
| [galera]        | Options read by a galera-capable MariaDB Server. Available on systems compiled with Galera support.                                                                                      |

For example, if you specify the

`log_error`

option in a server option group in an option file, like this:

```
[mariadb]
log_error=error.log
```

Then

mysqld\_safe  
will also use this value for its own  
--log-error

option:

## Configuring the Open Files Limit

When using

`mysqld_safe`  
, the system's open files limit can be changed by providing the  
`--open-files-limit`  
option either on the command-line or in an option file. For example:

```
[mysqld_safe]  
open_files_limit=4294967295
```

The option value is passed to

`ulimit -n`  
. Note that you need to start  
`mysqld_safe`  
as root for this to work properly. However, you can't currently set this to  
unlimited  
. See [MDEV-18410](#) about that.

When

`mysqld_safe`  
starts  
`mysqld`  
, it also uses this option to set the value of the

`open_files_limit`

system variable for  
`mysqld`

## Configuring the Core File Size

When using

`mysqld_safe`  
, if you would like to [enable core dumps](#), the system's core file size limit can be changed by providing the  
`--core-file-size`  
option either on the command-line or in an option file. For example:

```
[mysqld_safe]  
core_file_size=unlimited
```

The option value is passed to

`ulimit -c`  
. Note that you need to start  
`mysqld_safe`  
as root for this to work properly.

## Configuring MariaDB to Write the Error Log to Syslog

When using

`mysqld_safe`  
, if you would like to redirect the error log to the [syslog](#), then that can easily be done by using the  
`--syslog`  
option.  
`mysqld_safe`  
redirects two types of log messages to the syslog--its own log messages, and log messages for  
`mysqld`

- - `mysqld_safe`  
configures its own log messages to go to the  
daemon  
syslog facility. The log level for these messages is either  
`notice`  
or  
`error`

, depending on the specific type of log message. The default tag is  
`mysqld_safe`

- `mysqld_safe`  
also configures the log messages for  
`mysqld`  
to go to the  
daemon  
syslog facility. The log level for these messages is  
error  
. The default tag is  
`mysqld`

Sometimes it can be helpful to add a suffix to the syslog tag, such as if you are running multiple instances of MariaDB on the same host. To add a suffix to each syslog tag, use the

`--syslog-tag`  
option.

## Specifying mysqld

By default,

`mysqld_safe`  
tries to start an executable named  
`mysqld`

You can also specify another executable for

`mysqld_safe`  
to start instead of  
`mysqld`  
by providing the  
`--mysqld`  
or  
`--mysqld-version`  
options either on the command-line or in an option file.

By default, it will look for

`mysqld`  
in the following locations in the following order:

- `$BASEDIR/libexec/mysqld`
- `$BASEDIR/sbin/mysqld`
- `$BASEDIR/bin/mysqld`
- `$PWD/bin/mysqld`
- `$PWD/libexec/mysqld`
- `$PWD/sbin/mysqld`
- `@libexecdir@/mysql`

Where

`$BASEDIR`  
is set by the  
`--basedir`  
option,  
`$PWD`  
is the current working directory where  
`mysqld_safe`

was invoked, and  
@libexecdir@  
is set at compile-time by the  
INSTALL\_BINDIR  
option for

`cmake`

You can also specify where the executable is located by providing the  
--ledir  
option either on the command-line or in an option file.

## Specifying datadir

By default,

`mysqld_safe`  
will look for the  
datadir  
in the following locations in the following order:

- \$BASEDIR/data/mysql
- \$BASEDIR/data
- \$BASEDIR/var/mysql
- \$BASEDIR/var
- @localstatedir@

Where

\$BASEDIR  
is set by the  
--basedir  
option, and  
@localstatedir@  
is set at compile-time by the  
INSTALL\_MYSQLDATADIR  
option for

`cmake`

You can also specify where the  
datadir  
is located by providing the  
--datadir  
option either on the command-line or in an option file.

## Logging

When you use

`mysqld_safe`  
to start  
`mysqld`  
,

`mysqld_safe`  
logs to the same destination as  
`mysqld`

`mysqld_safe`

has several log-related options:

- - - syslog
      - : Write error messages to syslog on systems that support the logger program.
    - - skip-syslog
        - : Do not write error messages to syslog. Messages are written to the default error log file (host\_name.err in the data directory), or to a named file if the
        - log-error
          - option is given.
      - - log-error=file\_name
          - : Write error messages to the named error file.

If none of these options is provided, then the default is

--skip-syslog

If

--syslog  
and  
--log-error  
are both provided, then a warning is issued and  
--log-error  
takes precedence.

mysqld\_safe  
also writes notices to  
stdout  
and errors to  
stderr

## Editing mysqld\_safe

mysqld\_safe  
is a  
sh

script, so if you need to change its behavior, then it can easily be edited. However, you should not normally edit the script. A lot of behavior can be changed by providing options either on the command-line or in an option file.

If you do edit

mysqld\_safe

, then you should be aware of the fact that a package upgrade can overwrite your changes. If you would like to preserve your changes, be sure to have a backup.

## NetWare

On NetWare, mysqld\_safe is a NetWare Loadable Module (NLM) that is ported from the original Unix shell script. It starts the server as follows:

1. Runs a number of system and option checks.
2. Runs a check on MyISAM tables.
3. Provides a screen presence for the MariaDB server.
4. Starts mysqld, monitors it, and restarts it if it terminates in error.
5. Sends error messages from mysqld to the host\_name.err file in the data directory.
6. Sends mysqld\_safe screen output to the host\_name.safe file in the data directory.

## See Also

- [How to increase max number of open files on Linux](#) . This can be used to solve issues like this warning from mysqld:  
Changed limits: max\_open\_files: 1024 (requested 5000)"

- [mysqld Options](#)
- [systemd](#)

## 2.1.6.9 mysqladmin

From MariaDB 10.4.6 ,  
mariadb-admin  
is a symlink to  
mysqladmin  
.

MariaDB starting with 10.5.2

From MariaDB 10.5.2 ,  
mariadb-admin  
is the name of the script, with  
mysqladmin  
a symlink .

## Contents

1. [Using mysqladmin](#)
  1. [Options](#)
  2. [Option Files](#)
    1. [Option Groups](#)
  2. [mysqladmin Variables](#)
  3. [The shutdown Command and the --wait-for-all-slaves Option](#)
  4. [Examples](#)
    1. [Other Ways To Stop mysqld \(Unix\)](#)
  5. [See Also](#)

`mysqladmin`  
is an administration program for the mysqld daemon. It can be used to:

- Monitor what the MariaDB clients are doing (processlist)
- Get usage statistics and variables from the MariaDB server
- Create/drop databases
- Flush (reset) logs, statistics and tables
- Kill running queries.
- Stop the server (shutdown)
- Start/stop replicas
- Check if the server is alive (ping)

## Using mysqladmin

The command to use

`mysqladmin`  
and the general syntax is:

```
mysqladmin [options] command [command-arg] [command [command-arg]] ...
```

## Options

`mysqladmin`  
supports the following options:

| Option                    | Description                                                                                  | Added |
|---------------------------|----------------------------------------------------------------------------------------------|-------|
| --character-sets-dir=name | Directory where the <a href="#">character set</a> files are located.                         |       |
| -C<br>'<br>--compress     | Compress all information sent between the client and the server if both support compression. |       |

|                                                     |                                                                                                                                                                                                                                                                                                                                                                                      |                                  |
|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|
| --connect_timeout=val                               | Maximum time in seconds before connection timeout. The default value is 43200 (12 hours).                                                                                                                                                                                                                                                                                            |                                  |
| -c val<br>,--count=val                              | Number of iterations to make. This works with<br>-i<br>(<br>--sleep<br>) only.                                                                                                                                                                                                                                                                                                       |                                  |
| --debug[=debug_options]<br>,--#[br/>[debug_options] | Write a debugging log. A typical debug_options string is<br>d:t:o,file_name<br>. The default is<br>d:t:o,/tmp/mysqladmin.trace                                                                                                                                                                                                                                                       |                                  |
| --debug-check                                       | Check memory and open file usage at exit.                                                                                                                                                                                                                                                                                                                                            |                                  |
| --debug-info                                        | Print debugging information and memory and CPU usage statistics when the program exits.                                                                                                                                                                                                                                                                                              |                                  |
| --default-auth=plugin                               | Default authentication client-side plugin to use.                                                                                                                                                                                                                                                                                                                                    |                                  |
| --default-character-set=name                        | Set the default character set.                                                                                                                                                                                                                                                                                                                                                       |                                  |
| -f<br>,--force                                      | Don't ask for confirmation on drop database; with multiple commands, continue even if an error occurs.                                                                                                                                                                                                                                                                               |                                  |
| -?<br>,--help                                       | Display this help and exit.                                                                                                                                                                                                                                                                                                                                                          |                                  |
| -h name<br>,--host=name                             | Hostname to connect to.                                                                                                                                                                                                                                                                                                                                                              |                                  |
| -l<br>,--local                                      | Suppress the SQL command(s) from being written to the <a href="#">binary log</a> by enabling <code>sql_log_bin=0</code> for the session, or, from <a href="#">MariaDB 10.2.7</a> and <a href="#">MariaDB 10.1.24</a> , for flush commands only, using<br>FLUSH LOCAL<br>rather than<br>SET <code>sql_log_bin=0</code><br>, so the privilege requirement is RELOAD rather than SUPER. | MariaDB 10.2.5 , MariaDB 10.1.22 |
| -b<br>,--no-beep                                    | Turn off beep on error.                                                                                                                                                                                                                                                                                                                                                              |                                  |

|                            |                                                                                                                                                                                                                                                                                                                                                                                         |  |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| -<br>p[password]<br>,      | password to use when connecting to server. If password is not given it's asked from the terminal.                                                                                                                                                                                                                                                                                       |  |
| --pipe<br>,                | On Windows, connect to the server via a named pipe. This option applies only if the server supports named-pipe connections.                                                                                                                                                                                                                                                             |  |
| -P<br>portnum<br>,         | Port number to use for connection, or 0 for default to, in order of preference, my.cnf, \$MYSQL_TCP_PORT, /etc/services, built-in default (3306).                                                                                                                                                                                                                                       |  |
| --<br>protocol=name        | The protocol to use for connection (tcp, socket, pipe, memory).                                                                                                                                                                                                                                                                                                                         |  |
| -r<br>,                    | Show difference between current and previous values when used with<br>-i<br>. Currently only works with extended-status.                                                                                                                                                                                                                                                                |  |
| -O value<br>,              | Change the value of a variable. Please note that this option is deprecated; you can set variables directly with<br>--variable-name=value                                                                                                                                                                                                                                                |  |
| --<br>variable=value       |                                                                                                                                                                                                                                                                                                                                                                                         |  |
| --<br>shutdown_timeout=val | Maximum number of seconds to wait for server shutdown. The default value is 3600 (1 hour).                                                                                                                                                                                                                                                                                              |  |
| -s<br>,                    | Silently exit if one can't connect to server.                                                                                                                                                                                                                                                                                                                                           |  |
| --silent                   |                                                                                                                                                                                                                                                                                                                                                                                         |  |
| -i delay<br>,              | Execute commands repeatedly, sleeping for <i>delay</i> seconds in between. The<br>--count<br>option determines the number of iterations. If<br>--count<br>is not given, <i>mysqladmin</i> executes commands indefinitely until interrupted.                                                                                                                                             |  |
| -S name<br>,               | For connections to localhost, the Unix socket file to use, or, on Windows, the name of the named pipe to use.                                                                                                                                                                                                                                                                           |  |
| --socket=name              |                                                                                                                                                                                                                                                                                                                                                                                         |  |
| --ssl                      | Enables <a href="#">TLS</a> . TLS is also enabled even without setting this option when certain other TLS options are set. Starting with <a href="#">MariaDB 10.2</a> , the<br>--ssl<br>option will not enable <a href="#">verifying the server certificate</a> by default. In order to verify the server certificate, the user must specify the<br>--ssl-verify-server-cert<br>option. |  |

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| ca=name<br>--ssl-            | Defines a path to a PEM file that should contain one or more X509 certificates for trusted Certificate Authorities (CAs) to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. See <a href="#">Secure Connections Overview: Certificate Authorities (CAs)</a> for more information. This option implies the<br><br>--ssl<br>option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                |
| capath=name<br>--ssl-        | Defines a path to a directory that contains one or more PEM files that should each contain one X509 certificate for a trusted Certificate Authority (CA) to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the<br><br><code>openssl rehash</code><br><br>command. See <a href="#">Secure Connections Overview: Certificate Authorities (CAs)</a> for more information. This option is only supported if the client was built with OpenSSL or yaSSL. If the client was built with GnuTLS or Schannel, then this option is not supported. See <a href="#">TLS and Cryptography Libraries Used by MariaDB</a> for more information about which libraries are used on which platforms. This option implies the<br><br>--ssl<br>option. |                |
| cert=name<br>--ssl-          | Defines a path to the X509 certificate file to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. This option implies the<br><br>--ssl<br>option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |
| cipher=name<br>--ssl-        | List of permitted ciphers or cipher suites to use for <a href="#">TLS</a> . This option implies the<br><br>--ssl<br>option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |
| crl=name<br>--ssl-           | Defines a path to a PEM file that should contain one or more revoked X509 certificates to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. See <a href="#">Secure Connections Overview: Certificate Revocation Lists (CRLs)</a> for more information. This option is only supported if the client was built with OpenSSL or Schannel. If the client was built with yaSSL or GnuTLS, then this option is not supported. See <a href="#">TLS and Cryptography Libraries Used by MariaDB</a> for more information about which libraries are used on which platforms.                                                                                                                                                                                                                                   |                |
| crlpath=name<br>--ssl-       | Defines a path to a directory that contains one or more PEM files that should each contain one revoked X509 certificate to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. The directory specified by this option needs to be run through the<br><br><code>openssl rehash</code><br><br>command. See <a href="#">Secure Connections Overview: Certificate Revocation Lists (CRLs)</a> for more information. This option is only supported if the client was built with OpenSSL. If the client was built with yaSSL, GnuTLS, or Schannel, then this option is not supported. See <a href="#">TLS and Cryptography Libraries Used by MariaDB</a> for more information about which libraries are used on which platforms.                                                                             |                |
| key=name<br>--ssl-           | Defines a path to a private key file to use for <a href="#">TLS</a> . This option requires that you use the absolute path, not a relative path. This option implies the<br><br>--ssl<br>option.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |
| verify-server-cert<br>--ssl- | Enables <a href="#">server certificate verification</a> . This option is disabled by default.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |
| version=name<br>--tls-       | This option accepts a comma-separated list of TLS protocol versions. A TLS protocol version will only be enabled if it is present in this list. All other TLS protocol versions will not be permitted. See <a href="#">Secure Connections Overview: TLS Protocol Versions</a> for more information.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | MariaDB 10.4.6 |
| user=name<br>-u<br>,         | User for login if not current user.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |
| -v<br>,                      | Write more information.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |
| --verbose                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |

|  |                                                             |                                                                                                                                                                                    |                   |
|--|-------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
|  | <code>-V</code><br>,<br><code>--version</code>              | Output version information and exit.                                                                                                                                               |                   |
|  | <code>-E</code><br>,<br><code>--vertical</code>             | Print output vertically. Is similar to '<br><code>--relative</code><br>', but prints output vertically.                                                                            |                   |
|  | <code>-w[ count]</code><br>,<br><code>--wait[=count]</code> | If the connection cannot be established, wait and retry instead of aborting. If a <i>count</i> value is given, it indicates the number of times to retry. The default is one time. |                   |
|  | <code>--wait-for-all-slaves</code>                          | Wait for the last binlog event to be sent to all connected replicas before shutting down. This option is off by default.                                                           | MariaDB<br>10.4.4 |

## Option Files

In addition to reading options from the command-line,

```
mysqladmin
can also read options from option files. If an unknown option is provided to
mysqladmin
in an option file, then it is ignored.
```

The following options relate to how MariaDB command-line tools handles option files. They must be given as the first argument on the command-line:

| Option                                 | Description                                                                         |
|----------------------------------------|-------------------------------------------------------------------------------------|
| <code>--print-defaults</code>          | Print the program argument list and exit.                                           |
| <code>--no-defaults</code>             | Don't read default options from any option file.                                    |
| <code>--defaults-file=#</code>         | Only read default options from the given file #.                                    |
| <code>--defaults-extra-file=#</code>   | Read this file after the global files are read.                                     |
| <code>--defaults-group-suffix=#</code> | In addition to the default option groups, also read option groups with this suffix. |

In [MariaDB 10.2](#) and later,

```
mysqladmin
is linked with MariaDB Connector/C. However, MariaDB Connector/C does not yet handle the parsing of option files for this client. That is still
performed by the server option file parsing code. See MDEV-19035 for more information.
```

## Option Groups

```
mysqladmin
reads options from the following option groups from option files:
```

| Group        | Description                                                                             |
|--------------|-----------------------------------------------------------------------------------------|
| [mysqladmin] | Options read by<br>mysqladmin<br>, which includes both MariaDB Server and MySQL Server. |

|                  |                                                                                                                                                                                          |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [mariadb-admin]  | Options read by<br>mysqladmin<br>. Available starting with <a href="#">MariaDB 10.4.6</a> .                                                                                              |
| [client]         | Options read by all MariaDB and MySQL <a href="#">client programs</a> , which includes both MariaDB and MySQL clients. For example,<br>mysqldump                                         |
| [client-server]  | Options read by all MariaDB <a href="#">client programs</a> and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients. |
| [client-mariadb] | Options read by all MariaDB <a href="#">client programs</a> .                                                                                                                            |

## mysqladmin Variables

Variables can be set with

--variable-name=value

| Variables and boolean options | Value                       |
|-------------------------------|-----------------------------|
| count                         | 0                           |
| debug-check                   | FALSE                       |
| debug-info                    | FALSE                       |
| force                         | FALSE                       |
| compress                      | FALSE                       |
| character-sets-dir            | (No default value)          |
| default-character-set         | (No default value)          |
| host                          | (No default value)          |
| no-beep                       | FALSE                       |
| port                          | 3306                        |
| relative                      | FALSE                       |
| socket                        | /var/run/mysqld/mysqld.sock |

|                        |                    |
|------------------------|--------------------|
| sleep                  | 0                  |
| ssl                    | FALSE              |
| ssl-ca                 | (No default value) |
| ssl-capath             | (No default value) |
| ssl-cert               | (No default value) |
| ssl-cipher             | (No default value) |
| ssl-key                | (No default value) |
| ssl-verify-server-cert | FALSE              |
| user                   | (No default value) |
| verbose                | FALSE              |
| vertical               | FALSE              |
| connect_timeout        | 43200              |
| shutdown_timeout       | 3600               |

## mysqladmin Commands

```
mysqladmin [options] command [command-arg] [command [command-arg]] ...
```

Command is one or more of the following. Commands may be shortened to a unique prefix.

| Command                | Description                                        | Added |
|------------------------|----------------------------------------------------|-------|
| create<br>databasename | Create a new database.                             |       |
| debug                  | Instruct server to write debug information to log. |       |
| drop<br>databasename   | Delete a database and all its tables.              |       |

|                         |                                                               |                                  |
|-------------------------|---------------------------------------------------------------|----------------------------------|
| extended-status         | Return all <a href="#">status variables</a> and their values. |                                  |
| flush-all-statistics    | Flush all statistics tables                                   |                                  |
| flush-all-status        | Flush status and statistics.                                  |                                  |
| flush-binary-log        | Flush <a href="#">binary log</a> .                            | MariaDB 10.1.25 , MariaDB 10.2.5 |
| flush-client-statistics | Flush client statistics.                                      |                                  |
| flush-engine-log        | Flush engine log.                                             | MariaDB 10.1.25 , MariaDB 10.2.5 |
| flush-error-log         | Flush <a href="#">error log</a> .                             | MariaDB 10.1.25 , MariaDB 10.2.5 |
| flush-general-log       | Flush <a href="#">general query log</a> .                     | MariaDB 10.1.25 , MariaDB 10.2.5 |
| flush-hosts             | Flush all cached hosts.                                       |                                  |
| flush-index-statistics  | Flush index statistics.                                       |                                  |
| flush-logs              | Flush all logs.                                               |                                  |
| flush-privileges        | Reload grant tables (same as reload).                         |                                  |
| flush-relay-log         | Flush <a href="#">relay log</a> .                             | MariaDB 10.1.25 , MariaDB 10.2.5 |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                     |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|
| flush-slow-log            | Flush slow query log.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                     |
| flush-ssl                 | Flush SSL certificates.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | (unreleased)                                                        |
| flush-status              | Clear <a href="#">status variables</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                                     |
| flush-table-statistics    | Clear table statistics.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                     |
| flush-tables              | Flush all tables.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                     |
| flush-threads             | Flush the thread cache.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                     |
| flush-user-resources      | Flush user resources.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <a href="#">MariaDB 10.1.25</a> ,<br><a href="#">MariaDB 10.2.5</a> |
| flush-user-statistics     | Flush user statistics.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                     |
| kill id,id,...            | Kill mysql threads.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                                                     |
| password new-password     | Change old password to new-password. The new password can be passed on the commandline as the next argument (for example,<br><code>mysqladmin password "new_password"</code><br>, or can be omitted (as long as no other command follows), in which case the user will be prompted for a password. If the password contains special characters, it needs to be enclosed in quotation marks. In Windows, the quotes can only be double quotes, as single quotes are assumed to be part of the password. If the server was started with the <a href="#">--skip-grant-tables</a> option, changing the password in this way will have no effect.) |                                                                     |
| old-password new-password | Change old password to new-password using the old pre-MySQL 4.1 format.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                     |
| ping                      | Check if mysqld is alive. Return status is 0 if the server is running (even in the case of an error such as access denied), 1 if it is not.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                     |
| processlist               | Show list of active threads in server, equivalent to <a href="#">SHOW PROCESSLIST</a> . With<br><code>--verbose</code><br>, equivalent to <a href="#">SHOW FULL PROCESSLIST</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                     |

|                  |                                                                                                                                                                                                                                                                                                                  |  |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| reload           | Reload grant tables.                                                                                                                                                                                                                                                                                             |  |
| refresh          | Flush all tables and close and open log files.                                                                                                                                                                                                                                                                   |  |
| shutdown         | Take server down by executing the <a href="#">SHUTDOWN</a> command on the server. If connected to a local server using a Unix socket file, mysqladmin waits until the server's process ID file has been removed to ensure that the server has stopped properly. See also the<br>--wait-for-all-slaves<br>option. |  |
| status           | Gives a short status message from the server.                                                                                                                                                                                                                                                                    |  |
| start-all-slaves | Start all replicas.                                                                                                                                                                                                                                                                                              |  |
| start-slave      | Start replication on a replica server.                                                                                                                                                                                                                                                                           |  |
| stop-all-slaves  | Stop all replicas.                                                                                                                                                                                                                                                                                               |  |
| stop-slave       | Stop replication on a replica server.                                                                                                                                                                                                                                                                            |  |
| variables        | Prints variables available.                                                                                                                                                                                                                                                                                      |  |
| version          | Returns version as well as status info from the server.                                                                                                                                                                                                                                                          |  |

## The shutdown Command and the --wait-for-all-slaves Option

MariaDB starting with [10.4.4](#)

The

--wait-for-all-slaves  
option was first added in [MariaDB 10.4.4](#).

When a master server is shutdown and it goes through the normal shutdown process, the master kills client threads in random order. By default, the master also considers its binary log dump threads to be regular client threads. As a consequence, the binary log dump threads can be killed while client threads still exist, and this means that data can be written on the master during a normal shutdown that won't be replicated. This is true even if [semi-synchronous replication](#) is being used.

In [MariaDB 10.4](#) and later, this problem can be solved by shutting down the server with the

`mysqladmin`

utility and by providing the  
--wait-for-all-slaves  
option to the utility and by executing the  
shutdown

command with the utility. For example:

```
mysqladmin --wait-for-all-slaves shutdown
```

When the

--wait-for-all-slaves

option is provided, the server only kills its binary log dump threads after all client threads have been killed, and it only completes the shutdown after the last [binary log](#) has been sent to all connected replicas.

See [Replication Threads: Binary Log Dump Threads and the Shutdown Process](#) for more information.

## Examples

Quick check of what the server is doing:

```
shell> mysqladmin status
Uptime: 8023 Threads: 1 Questions: 14 Slow queries: 0 Opens: 15 Flush tables: 1 Open tables: 8 Queries per second avg: 0.1
shell> mysqladmin processlist
+-----+-----+-----+-----+-----+
| Id | User | Host | db | Command | Time | State | Info |
+-----+-----+-----+-----+-----+
.....
+-----+-----+-----+-----+-----+
```

More extensive information of what is happening 'just now' changing (great for troubleshooting a slow server):

```
shell> mysqladmin --relative --sleep=1 extended-status | grep -v " 0 "
```

Check the variables for a running server:

```
shell> mysqladmin variables | grep datadir
| datadir | /my/data/ |
```

Using a shortened prefix for the

version

command:

```
shell> mysqladmin ver
mysqladmin Ver 9.1 Distrib 10.1.6-MariaDB, for debian-linux-gnu on x86_64
Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Server version 10.1.6-MariaDB-1~trusty-log
Protocol version 10
Connection Localhost via UNIX socket
UNIX socket /var/run/mysqld/mysqld.sock
Uptime: 1 hour 33 min 33 sec

Threads: 1 Questions: 281 Slow queries: 0 Opens: 64 Flush tables: 1 Open tables: 76 Queries per second avg: 0.050
```

## Other Ways To Stop mysqld (Unix)

If you get the error:

```
mysqladmin: shutdown failed; error: 'Access denied; you need (at least one of) the SHUTDOWN privilege(s) for this operation'
```

It means that you didn't use

mysqladmin

with a user that has the SUPER or SHUTDOWN privilege.

If you don't know the user password, you can still take the mysqld process down with a system

kill

command:

```
kill -SIGTERM pid-of-mysqld-process
```

The above is identical to

```
mysqladmin shutdown
```

On windows you should use:

You can use the [SHUTDOWN](#) command from any client.

## See Also

- [SHUTDOWN command](#)
- [mytop](#), a 'top' like program for MariaDB/MySQL that allows you to see what the server is doing. A mytop optimized for MariaDB is included in [MariaDB 5.3](#)

## 2.1.6.10 Switching Between Different Installed MariaDB Versions

### Contents

1. [Stopping a pre-installed MySQL/MariaDB from interfering with your tests](#)
2. [How to create a binary distribution \(tar file\)](#)
3. [Creating a directory structure for the different installations](#)
4. [Setting up the data directory](#)
  1. [Setting up a common data directory](#)
  2. [Setting up different data directories](#)
5. [Running a MariaDB server](#)
6. [Setting up a .my.cnf file for running multiple MariaDB main versions](#)

This article is about managing many different installed MariaDB versions and running them one at a time. This is useful when doing benchmarking, testing, or for when developing different MariaDB versions.

This is most easily done using the tar files from [downloads.askmonty.org](#).

## Stopping a pre-installed MySQL/MariaDB from interfering with your tests

If MySQL/MariaDB is already installed and running, you have two options:

1. Use test MariaDB servers with a different port & socket.
  - In this case you are probably best off creating a specific section for MariaDB in your  
`~/.my.cnf`  
 file.
2. Stop mysqld with
 

```
/etc/rc.d/mysql stop
```

 or
 

```
mysqladmin shutdown
```

Note that you don't have to uninstall or otherwise remove MySQL!

## How to create a binary distribution (tar file)

Here is a short description of how to generate a tar file from a source distribution. If you have [downloaded](#) a binary tar file, you can skip this section.

The steps to create a binary tar file are:

- Decide where to put the source. A good place is under  
`/usr/local/src/mariadb-5.#`
- [Get the source](#)
- [Compile the source](#)
- [Create the binary tar ball](#)

You will then be left with a tar file named something like:

```
mariadb-5.3.2-MariaDB-beta-linux-x86_64.tar.gz
```

## Creating a directory structure for the different installations

Install the binary tar files under

```
/usr/local/  
with the following directory names (one for each MariaDB version you want to use):
```

- mariadb-5.1
- mariadb-5.2
- mariadb-5.3
- mariadb-5.5
- mariadb-10.0

The above assumes you are just testing major versions of MariaDB. If you are testing specific versions, use directory names like  
`mariadb-5.3.2`

With the directories in place, create a sym-link named

```
mariadb  
which points at the  
mariadb-XXX  
directory you are currently testing. When you want to switch to testing a different version, just update the sym-link.
```

Example:

```
cd /usr/local  
tar xfz /tmp/mariadb-5.3.2-MariaDB-beta-linux-x86_64.tar.gz  
mv -vi mariadb-5.3.2-MariaDB-beta-linux-x86_64 mariadb-5.3  
ln -vs mariadb-5.3 mariadb
```

## Setting up the data directory

When setting up the data directory, you have the option of either using a shared database directory or creating a unique database directory for each server version. For testing, a common directory is probably easiest. Note that you can only have one

```
mysqld  
server running against one data directory.
```

### Setting up a common data directory

The steps are:

1. Create the

```
mysql  
system user if you don't have it already! (On Linux you do it with the  
useradd  
command).
```

2. Create the directory (we call it

```
mariadb-data  
in the example below) or add a symlink to a directory which is in some other place.
```

3. Create the

```
mysql  
permission tables with  
mysql_install_db
```

```
cd /usr/local/  
mkdir mariadb-data  
cd mariadb  
.bin/mysql_install_db --no-defaults --datadir=/usr/local/mariadb-data  
chown -R mysql mariadb-data mariadb-data/*
```

The reason to use

```
--no-defaults  
is to ensure that we don't inherit incorrect options from some old my.cnf.
```

### Setting up different data directories

To create a different

data  
directories for each installation:

```
cd mariadb  
.scripts/mysql_install_db --no-defaults  
chown -R mysql mariadb-data mariadb-data/*
```

This will create a directory

data  
inside the current directory.

If you want to use another disk you should do:

```
cd mariadb  
ln -s path-to-empty-directory-for-data data  
.scripts/mysql_install_db --no-defaults --datadir=./data  
chown -R mysql mariadb-data mariadb-data/*
```

## Running a MariaDB server

The normal steps are:

```
rm mariadb  
ln -s mariadb-5.# mariadb  
cd mariadb  
.bin/mysqld_safe --no-defaults --datadir=/usr/local/mariadb-data &
```

## Setting up a .my.cnf file for running multiple MariaDB main versions

If you are going to start/stop MariaDB a lot of times, you should create a

~/.my.cnf  
file for the common options you are using.

The following example shows how to use a non-standard TCP-port and socket (to not interfere with a main MySQL/MariaDB server) and how to setup different options for each main server:

```
[client-server]  
socket=/tmp/mysql.sock  
port=3306  
[mysqld]  
datadir=/usr/local/mariadb-data  
  
[mariadb-5.2]  
# Options for MariaDB 5.2  
[mariadb-5.3]  
# Options for MariaDB 5.3
```

If you create an

~/.my.cnf  
file, you should start  
mysqld  
with  
--defaults-file=~/.my.cnf  
instead of  
--no-defaults  
in the examples above.

## 2.1.6.11 Specifying Permissions for Schema (Data) Directories and Tables

### Default File Permissions

By default MariaDB uses the following permissions for files and directories:

| Object Type | Default Mode | Default Permissions |
|-------------|--------------|---------------------|
|-------------|--------------|---------------------|

|             |      |            |
|-------------|------|------------|
| Files       | 0660 | -rw-rw---- |
| Directories | 0700 | drwx-----  |

## Configuring File Permissions with Environment Variables

You can configure MariaDB to use different permissions for files and directories by setting the following [environment variables](#) before you start the server:

| Object Type | Environment Variable |
|-------------|----------------------|
| Files       | UMASK                |
| Directories | UMASK_DIR            |

In other words, if you would run the following in a shell:

```
export UMASK=0640
export UMASK_DIR=0750
```

These environment variables do not set the umask. They set the default file system permissions. See [MDEV-23058](#) for more information.

## Configuring File Permissions with systemd

If your server is started by

`systemd`

, then there is a specific way to configure the umask. See [Systemd: Configuring the umask](#) for more information.

### 2.1.6.12 mysqld\_multi

MariaDB starting with [10.4.6](#)

From [MariaDB 10.4.6](#) ,  
`mariadb-multi`  
 is a symlink to  
`mysqld_multi`  
 .

MariaDB starting with [10.5.2](#)

From [MariaDB 10.5.2](#) ,  
`mariadb-multi`  
 is the name of the server, with  
`mysqld_multi`  
 a symlink .

Before using `mysqld_multi` be sure that you understand the meanings of the options that are passed to the `mysqld` servers and why you would want to have separate `mysqld` processes. Beware of the dangers of using multiple `mysqld` servers with the same data directory. Use separate data directories, unless you know what you are doing. Starting multiple servers with the same data directory does not give you extra performance in a threaded system.

## Contents

1. [Using mysqld\\_multi](#)
  1. [Options](#)
  2. [Option Files](#)
    1. [Option Groups](#)
  3. [Authentication and Privileges](#)
2. [User Account](#)
3. [Example](#)

The

```
mysqld_multi
```

startup script is in MariaDB distributions on Linux and Unix. It is a wrapper that is designed to manage several  
mysqld  
processes running on the same host. In order for multiple  
mysqld  
processes to work on the same host, these processes must:

- Use different Unix socket files for local connections.
- Use different TCP/IP ports for network connections.
- Use different data directories.
- Use different process ID files (specified by the  
--pid-file  
option) if using

```
mysqld_safe
```

to start

```
mysqld
```

.

```
mysqld_multi
```

can start or stop servers, or report their current status.

## Using mysqld\_multi

The command to use

```
mysqld_multi
```

and the general syntax is:

```
mysqld_multi [options] {start|stop|report} [GNR[,GNR] ...]
```

```
start  
,  
stop  
, and  
report  
indicate which operation to perform.
```

You can specify which servers to perform the operation on by providing one or more

```
GNR  
values.
```

GNR  
refers to an option group number, and it is explained more in the [option groups](#) section below. If there is no  
GNR  
list, then  
mysqld\_multi  
performs the operation for all  
GNR  
values found in its option files.

Multiple

```
GNR  
values can be specified as a comma-separated list.  
GNR  
values can also be specified as a range by separating the numbers by a dash. There must not be any whitespace characters in the  
GNR  
list.
```

For example:

This command starts a single server using option group

```
[mysqld17]
```

```
:
```

```
mysqld_multi start 17
```

This command stops several servers, using option groups

```
[mysqld8]
```

```
and
```

```
[mysqld10]
```

```
through
```

```
[mysqld13]
```

```
:
```

```
mysqld_multi stop 8,10-13
```

## Options

`mysqld_multi`

supports the following options:

| Option                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| --example              | Give an example of a config file with extra information.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| --help                 | Display help and exit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| --log=filename         | Specify the path and name of the log file. If the file exists, log output is appended to it.                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| --mysqladmin=prog_name | The <code>mysqladmin</code> binary to be used to stop servers. Can be given within groups<br>[mysqld#]<br>.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| --mysqld=prog_name     | The mysqld binary to be used. Note that you can also specify <code>mysqld_safe</code> as the value for this option. If you use <code>mysqld_safe</code> to start the server, you can include the<br>mysqld<br>or<br>ledir<br>options in the corresponding<br>[mysqldN]<br>option group. These options indicate the name of the server that <code>mysqld_safe</code> should start and the path name of the directory where the server is located. Example:<br>[mysqld38]<br><br>mysqld = mysqld-debug<br><br>ledir = /opt/local/mysql/libexec<br>. |
| --no-log               | Print to stdout instead of the log file. By default the log file is turned on.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| --password=password    | The password of the MariaDB account to use when invoking <code>mysqladmin</code> . Note that the password value is not optional for this option, unlike for other MariaDB programs.                                                                                                                                                                                                                                                                                                                                                               |

|                     |                                                                                                                                                                                                                                                                                                                                      |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| --silent            | Silent mode; disable warnings.                                                                                                                                                                                                                                                                                                       |
| --tcp-ip            | Connect to the MariaDB server(s) via the TCP/IP port instead of the UNIX socket. This affects stopping and reporting. If a socket file is missing, the server may still be running, but can be accessed only via the TCP/IP port. By default connecting is done via the UNIX socket. This option affects stop and report operations. |
| --user=username     | The user name of the MariaDB account to use when invoking <a href="#">mysqladmin</a> .                                                                                                                                                                                                                                               |
| --verbose           | Be more verbose.                                                                                                                                                                                                                                                                                                                     |
| --version           | Display version information and exit.                                                                                                                                                                                                                                                                                                |
| --wsrep-new-cluster | Bootstrap a cluster. Added in <a href="#">MariaDB 10.1.15</a> .                                                                                                                                                                                                                                                                      |

## Option Files

In addition to reading options from the command-line,

```
mysqld_multi
can also read options from option files. If an unknown option is provided to
mysqld_multi
in an option file, then it is ignored.
```

The following options relate to how MariaDB command-line tools handles option files. They must be given as the first argument on the command-line:

| Option                    | Description                                                                         |
|---------------------------|-------------------------------------------------------------------------------------|
| --print-defaults          | Print the program argument list and exit.                                           |
| --no-defaults             | Don't read default options from any option file.                                    |
| --defaults-file=#         | Only read default options from the given file #.                                    |
| --defaults-extra-file=#   | Read this file after the global files are read.                                     |
| --defaults-group-suffix=# | In addition to the default option groups, also read option groups with this suffix. |

## Option Groups

`mysqld_safe`  
reads options from the following [option groups](#) from [option files](#):

| Group          | Description                                                                                         |
|----------------|-----------------------------------------------------------------------------------------------------|
| [mysqld_multi] | Options read by<br><code>mysqld_multi</code> , which includes both MariaDB Server and MySQL Server. |

`mysqld_multi`  
also searches [option files](#) for [option groups](#) with names like

[mysqldN]  
, where  
N  
can be any positive integer. This number is referred to in the following discussion as the option group number, or  
GNR  
:

| Group     | Description                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [mysqldN] | Options read by a<br>mysqld<br>instance managed by<br>mysqld_multi<br>, which includes both MariaDB Server and MySQL Server. The<br>N<br>refers to the instance's<br>GNR<br>. |

GNR  
values distinguish option groups from one another and are used as arguments to  
mysqld\_multi  
to specify which servers you want to start, stop, or obtain a status report for. The  
GNR  
value should be the number at the end of the option group name in the option file. For example, the  
GNR  
for an option group named  
[mysqld17]  
is  
17  
.

Options listed in these option groups are the same that you would use in the regular server option groups used for configuring  
mysqld  
. However, when using multiple servers, it is necessary that each one use its own value for options such as the Unix socket file and TCP/IP port number.

The

[mysqld\_multi]  
option group can be used for options that are needed for  
mysqld\_multi  
itself.  
[mysqldN]  
option groups can be used for options passed to specific  
mysqld  
instances.

The regular server [option groups](#) can also be used for common options that are read by all instances:

| Group        | Description                                                                                                                                 |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| [mysqld]     | Options read by<br>mysqld<br>, which includes both MariaDB Server and MySQL Server.                                                         |
| [server]     | Options read by MariaDB Server.                                                                                                             |
| [mysqld-X.Y] | Options read by a specific version of<br>mysqld<br>, which includes both MariaDB Server and MySQL Server. For example,<br>[mysqld-5.5]<br>. |
| [mariadb]    | Options read by MariaDB Server.                                                                                                             |

|                 |                                                                                                                                                                                          |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [mariadb-X.Y]   | Options read by a specific version of MariaDB Server.                                                                                                                                    |
| [client-server] | Options read by all MariaDB <a href="#">client programs</a> and the MariaDB Server. This is useful for options like socket and port, which is common between the server and the clients. |
| [galera]        | Options read by a galera-capable MariaDB Server. Available on systems compiled with Galera support.                                                                                      |

For an example of how you might set up an option file, use this command:

```
mysqld_multi --example
```

## Authentication and Privileges

Make sure that the MariaDB account used for stopping the

`mysqld`  
processes (with the

`mysqladmin`

utility) has the same user name and password for each server. Also, make sure that the account has the SHUTDOWN privilege. If the servers that you want to manage have different user names or passwords for the administrative accounts, you might want to create an account on each server that has the same user name and password. For example, you might set up a common `multi_admin` account by executing the following commands for each server:

```
shell> mysql -u root -S /tmp/mysql.sock -p
Enter password:
mysql> GRANT SHUTDOWN ON *.* TO 'multi_admin'@'localhost' IDENTIFIED BY 'multipass';
```

Change the connection parameters appropriately when connecting to each one. Note that the host name part of the account name must allow you to connect as

`multi_admin`  
from the host where you want to run  
`mysqld_multi`

## User Account

Make sure that the data directory for each server is fully accessible to the Unix account that the specific

`mysqld`  
process is started as. If you run the  
`mysqld_multi`  
script as the Unix  
root  
account, and if you want the  
`mysqld`  
process to be started with another Unix account, then you can use the  
--user  
option with  
`mysqld`  
. If you specify the  
--user  
option in an option file, and if you did not run the  
`mysqld_multi`  
script as the Unix  
root  
account, then it will just log a warning and the  
`mysqld`  
processes are started under the original Unix account.

Do not run the

```
mysqld  
process as the Unix  
root  
account, unless you know what you are doing.
```

## Example

The following example shows how you might set up an option file for use with mysqld\_multi. The order in which the mysqld programs are started or stopped depends on the order in which they appear in the option file. Group numbers need not form an unbroken sequence. The first and fifth [mysqldN] groups were intentionally omitted from the example to illustrate that you can have "gaps" in the option file. This gives you more flexibility.

```
# This file should probably be in your home dir (~/.my.cnf)  
# or /etc/my.cnf  
# Version 2.1 by Jani Tolonen  
[mysqld_multi]  
mysqld      = /usr/local/bin/mysqld_safe  
mysqladmin  = /usr/local/bin/mysqladmin  
user        = multi_admin  
password    = multipass  
[mysqld2]  
socket      = /tmp/mysql.sock2  
port        = 3307  
pid-file   = /usr/local/mysql/var2/hostname.pid2  
datadir    = /usr/local/mysql/var2  
language   = /usr/local/share/mysql/english  
user        = john  
[mysqld3]  
socket      = /tmp/mysql.sock3  
port        = 3308  
pid-file   = /usr/local/mysql/var3/hostname.pid3  
datadir    = /usr/local/mysql/var3  
language   = /usr/local/share/mysql/swedish  
user        = monty  
[mysqld4]  
socket      = /tmp/mysql.sock4  
port        = 3309  
pid-file   = /usr/local/mysql/var4/hostname.pid4  
datadir    = /usr/local/mysql/var4  
language   = /usr/local/share/mysql/estonia  
user        = tonu  
[mysqld6]  
socket      = /tmp/mysql.sock6  
port        = 3311  
pid-file   = /usr/local/mysql/var6/hostname.pid6  
datadir    = /usr/local/mysql/var6  
language   = /usr/local/share/mysql/japanese  
user        = jani
```

## 2.1.6.13 launchd

In Mac OS, create a file called /Library/LaunchDaemons/com.mariadb.server.plist with the following contents (edit to suit):

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key> <string>com.mariadb.server</string>
    <key>KeepAlive</key><true/>
    <key>RunAtLoad</key><true/>
    <key>LaunchOnlyOnce</key><false/>
    <key>ExitTimeOut</key><integer>600</integer>
        <key>WorkingDirectory</key><string>/usr/local/var</string>
        <key>Program</key><string>/usr/local/bin/mysqld</string>
        <key>ProgramArguments</key>
        <array>
            <string>/usr/local/bin/mysqld</string>
            <string>--user=_mysql</string>
            <string>--basedir=/usr/local/opt/mariadb</string>
            <string>--plugin-dir=/usr/local/opt/mariadb/lib/plugin</string>
            <string>--datadir=/usr/local/var/mysql</string>
            <string>--log-error=/usr/local/var/mysql/Data-Server.local.err</string>
            <string>--pid-file=/usr/local/var/mysql/Data-Server.local.pid</string>
            <string>--sql-mode=ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION</string>
        </array>
    </dict>
</plist>

```

Then from a shell, run

```

launchctl load /Library/LaunchDaemons/com.mariadb.server.plist
and MariaDB will run immediately, and also upon reboot.

```

## See Also

- [Creating Launch Daemons and Agents](#)
- [A launchd Tutorial](#)

## 2.1.6.14 systemd

MariaDB starting with [10.1.8](#)

The MariaDB

```

systemd
service was first released in MariaDB 10.1.8 on supported Linux distributions.

```

systemd  
is a

[sysVinit](#)

replacement that is the default service manager on the following Linux distributions:

- RHEL 7 and above
- CentOS 7 and above
- Fedora 15 and above
- Debian 8 and above
- Ubuntu 15.04 and above
- SLES 12 and above
- OpenSUSE 12.2 and above

MariaDB's

```

systemd
unit file is included in the server packages for RPMs and DEBs. It is also included in certain binary tarballs.

```

The service name is

```

mariadb.service
. Aliases to
mysql.service
and
mysqld.service
are also installed for convenience.

```

When MariaDB is started with the

systemd

unit file, it directly starts the

`mysqld`

process as the

`mysql`

user. Unlike with

`sysVinit`

, the

`mysqld`

process is not started with

`mysqld_safe`

. As a consequence, options will not be read from the

`[mysqld_safe]`

option group from option files .

## Contents

1. Locating the MariaDB Service's Unit File
2. Interacting with the MariaDB Server Process
  1. Starting the MariaDB Server Process on Boot
  2. Starting the MariaDB Server Process
  3. Stopping the MariaDB Server Process
  4. Restarting the MariaDB Server Process
  5. Checking the Status of the MariaDB Server Process
6. Interacting with Multiple MariaDB Server Processes
  1. Default configuration of Multiple Instances in 10.4 and Later
  2. Custom configuration of Multiple Instances in 10.4 and Later
  3. Configuring Multiple Instances in 10.3 and Earlier
3. Systemd and Galera Cluster
  1. Bootstrapping a New Cluster
  2. Recovering a Node's Cluster Position
  3. SSTs and Systemd
4. Configuring the Systemd Service
  1. Useful Systemd Options
  2. Configuring the Systemd Service Timeout
  3. Configuring the Open Files Limit
  4. Configuring the Core File Size
  5. Configuring MariaDB to Write the Error Log to Syslog
  6. Configuring LimitMEMLOCK
  7. Configuring Access to Home Directories
  8. Configuring the umask
  9. Configuring the data directory
5. Systemd Socket Activation
  1. Using Systemd Socket Activation
  2. When to Use Systemd Socket Activation
  3. Downsides to Using Systemd Socket Activation
  4. Configuring Systemd Socket Activation
  5. Extra Port
  6. Multi-instance socket activation
6. Systemd Socket Activation for Hosting Service Providers
  1. End User Benefits
  2. Hosting Service Provider Benefits
  3. Downsides to the Hosting Service Provider
  4. Example on configuration Items for a per user, systemd socket activated multi-instance service
    1. A MariaDB Template File
    2. Custom Configuration for the Multiinstance Service
    3. Custom Configuration for the Multi-instance Socket
  7. Systemd Journal
  8. Converting mysqld\_safe Options to Systemd Options
  9. Known Issues

## Locating the MariaDB Service's Unit File

By default, the unit file for the service will be installed in a directory defined at build time by the

`INSTALL_SYSTEMD_UNITDIR`  
option provided to

`cmake`

For example, on RHEL, CentOS, Fedora, and other similar Linux distributions,

`INSTALL_SYSTEMD_UNITDIR`  
is defined as  
`/usr/lib/systemd/system/`  
, so it will be installed to:

- `/usr/lib/systemd/system/mariadb.service`

And on Debian, Ubuntu, and other similar Linux distributions,

`INSTALL_SYSTEMD_UNITDIR`  
is defined as  
`/lib/systemd/system/`  
, so it will be installed to:

- `/lib/systemd/system/mariadb.service`

## Interacting with the MariaDB Server Process

The service can be interacted with by using the

`systemctl`

command.

### Starting the MariaDB Server Process on Boot

MariaDB's

`systemd`  
service can be configured to start at boot by executing the following:

```
sudo systemctl enable mariadb.service
```

### Starting the MariaDB Server Process

MariaDB's

`systemd`  
service can be started by executing the following:

```
sudo systemctl start mariadb.service
```

MariaDB's

`systemd`  
unit file has a default startup timeout of about 90 seconds on most systems. If certain startup tasks, such as crash recovery, take longer than this default startup timeout, then

`systemd`  
will assume that  
`mysqld`  
has failed to startup, which causes  
`systemd`  
to kill the  
`mysqld`  
process. To work around this, you can reconfigure the MariaDB  
`systemd`  
unit to have an [infinite timeout](#).

Note that [systemd 236 added the](#)

`EXTEND_TIMEOUT_USEC`  
[environment variable](#) that allows services to extend the startup timeout during long-running processes. Starting with [MariaDB 10.1.33](#), [MariaDB 10.2.15](#), and [MariaDB 10.3.6](#), on systems with `systemd` versions that support it, MariaDB uses this feature to extend the startup timeout during certain startup processes that can run long. Therefore, if you are using

`systemd`  
236 or later, then you should not need to manually override  
`TimeoutStartSec`

, even if your startup tasks, such as crash recovery, run for longer than the configured value. See [MDEV-14705](#) for more information.

## Stopping the MariaDB Server Process

MariaDB's

systemd  
service can be stopped by executing the following:

```
sudo systemctl stop mariadb.service
```

## Restarting the MariaDB Server Process

MariaDB's

systemd  
service can be restarted by executing the following:

```
sudo systemctl restart mariadb.service
```

## Checking the Status of the MariaDB Server Process

The status of MariaDB's

systemd  
service can be obtained by executing the following:

```
sudo systemctl status mariadb.service
```

## Interacting with Multiple MariaDB Server Processes

A

systemd  
template unit file with the name  
mariadb@.service  
is installed in  
INSTALL\_SYSTEMD\_UNITDIR  
on some systems. See [Locating the MariaDB Service's Unit File](#) to see what directory that refers to on each distribution.

This template unit file allows you to interact with multiple MariaDB instances on the same system using the same template unit file. When you interact with a MariaDB instance using this template unit file, you have to provide an instance name as a suffix. For example, the following command tries to start a MariaDB instance with the name

```
node1
```

```
:
```

```
sudo systemctl start mariadb@node1.service
```

MariaDB's build system cannot include the

mariadb@.service  
template unit file in [RPM](#) packages on platforms that have

[cmake](#)

versions older than 3.3.0, because these

[cmake](#)

versions have a [bug](#) that causes it to encounter errors when packaging a file in RPMs if the file name contains the  
@  
character. MariaDB's RHEL 7 and CentOS 7 RPM build hosts only got a new enough

[cmake](#)

version starting with [MariaDB 10.1.39](#) , [MariaDB 10.2.23](#) , and [MariaDB 10.3.14](#) . To use this functionality on a MariaDB version that does not have the file, you can copy the file from a package that does have the file.

## Default configuration of Multiple Instances in 10.4 and Later

```
systemd  
will also look for an option file for a specific MariaDB instance based on the instance name.
```

It will use the

```
.%I  
as the custom option group suffix that is appended to any server option group, in any configuration file included by default.
```

In all distributions, the

```
%I  
is the MariaDB instance name. In the above  
node1  
case, it would use the option file at the path  
/etc/mynode1.cnf
```

When using multiple instances, each instance will of course also need their own

```
datadir  
,  
socket  
and,  
port  
(unless  
skip\_networking  
is specified). As
```

```
mysql\_install\_db#option-groups
```

```
reads the same sections as the server, and  
ExecStartPre=  
run
```

```
mysql\_install\_db
```

```
within the service, the instances are autocreated if there is sufficient priviledges.
```

To use a 10.3 configuration in 10.4 or later and the following customisation in the editor after running

```
sudo systemctl edit mariadb@.service  
:
```

```
[Unit]  
ConditionPathExists=  
  
[Service]  
Environment='MYSQLD_MULTI_INSTANCE=--defaults-file=/etc/my%I.cnf'
```

## Custom configuration of Multiple Instances in 10.4 and Later

Because users may want to do many various things with their multiple instances, we've provided a way to let the user define how they wish their multiple instances to run. The systemd environment variable

```
MYSQLD_MULTI_INSTANCE  
can be set to anything that mysqld and mysql\_install\_db will recognise.
```

A hosting environment where each user has their own instance may look like (with

```
sudo systemctl edit mariadb@.service  
):
```

```
[Service]  
ProtectHome=false  
Environment='MYSQLD_MULTI_INSTANCE=--defaults-file=/home/%I/my.cnf \  
--user=%I \  
--socket=/home/%I.sock \  
--datadir=/home/%I/mariadb_data \  
--skip-networking'
```

Here the instance name is the unix user of the service.

## Configuring Multiple Instances in 10.3 and Earlier

systemd  
will also look for an [option file](#) for a specific MariaDB instance based on the instance name. By default, it will look for the option file in a directory defined at build time by the  
`INSTALL_SYSCONF2DIR`  
option provided to

`cmake`

For example, on RHEL, CentOS, Fedora, and other similar Linux distributions,

`INSTALL_SYSCONF2DIR`

is defined as

`/etc/my.cnf.d/`

, so it will look for an option file that matches the format:

- `/etc/my.cnf.d/my%I.cnf`

And on Debian, Ubuntu, and other similar Linux distributions,

`INSTALL_SYSCONF2DIR`

is defined as

`/etc/mysql/conf.d//`

, so it will look for an option file that matches the format:

- `/etc/mysql/conf.d/my%I.cnf`

In all distributions, the

`%I`

is the MariaDB instance name. In the above

`node1`

case, it would use the [option file](#) at the path

`/etc/my.cnf.d/mynode1.cnf`

for RHEL-like distributions and

`/etc/mysql/conf.d/mynode1.cnf`

for Debian-like distributions.

When using multiple instances, each instance will of course also need their own

`datadir`

. See [mysql\\_install\\_db](#) for information on how to initialize the

`datadir`

for additional MariaDB instances.

## Systemd and Galera Cluster

### Bootstrapping a New Cluster

When using [Galera Cluster](#) with systemd, the first node in a cluster has to be started with

`galera_new_cluster`

. See [Getting Started with MariaDB Galera Cluster: Bootstrapping a New Cluster](#) for more information.

### Recovering a Node's Cluster Position

When using [Galera Cluster](#) with systemd, a node's position in the cluster can be recovered with

`galera_recovery`

. See [Getting Started with MariaDB Galera Cluster: Determining the Most Advanced Node](#) for more information.

### SSTs and Systemd

MariaDB's

```
systemd
unit file has a default startup timeout of about 90 seconds on most systems. If an SST takes longer than this default startup timeout on a joiner
node, then
systemd
will assume that
mysqld
has failed to startup, which causes
systemd
to kill the
mysqld
process on the joiner node. To work around this, you can reconfigure the MariaDB
systemd
unit to have an infinite timeout. See Introduction to State Snapshot Transfers \(SSTs\): SSTs and Systemd for more information.
```

Note that [systemd 236 added the](#)

```
EXTEND_TIMEOUT_USEC
environment variable that allows services to extend the startup timeout during long-running processes. Starting with MariaDB 10.1.35, MariaDB 10.2.17, and MariaDB 10.3.8, on systems with systemd versions that support it, MariaDB uses this feature to extend the startup timeout during long
SSTs. Therefore, if you are using
systemd
236 or later, then you should not need to manually override
TimeoutStartSec
, even if your SSTs run for longer than the configured value. See MDEV-15607 for more information.
```

## Configuring the Systemd Service

You can configure MariaDB's

```
systemd
service by creating a "drop-in" configuration file for the
systemd
service. On most systems, the
systemd
service's directory for "drop-in" configuration files is
/etc/systemd/system/mariadb.service.d/
. You can confirm the directory and see what "drop-in" configuration files are currently loaded by executing:
```

```
$ sudo systemctl status mariadb.service
● mariadb.service - MariaDB 10.1.37 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
   Drop-In: /etc/systemd/system/mariadb.service.d
             └─migrated-from-my.cnf-settings.conf, timeoutstartsec.conf
...
...
```

If you want to configure the

```
systemd
service, then you can create a file with the
.conf
extension in that directory. The configuration option(s) that you would like to change would need to be placed in an appropriate section within the
file, usually
[Service]
. If a
systemd
option is a list, then you may need to set the option to empty before you set the replacement values. For example:
```

```
[Service]
ExecStart=
ExecStart=/usr/bin/numactl --interleave=all /usr/sbin/mysqld ${MYSQLD_OPTS} ${_WSREP_NEW_CLUSTER} ${_WSREP_START_POSITION}
```

After any configuration change, you will need to execute the following for the change to go into effect:

```
sudo systemctl daemon-reload
```

## Useful Systemd Options

Useful

```
systemd
options are listed below. If an option is equivalent to a common
```

## `mysqld_safe`

option, then that is also listed.

| <code>mysqld_safe</code><br>option         | systemd option                                     | Comments                                                                                                                        |
|--------------------------------------------|----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| no option                                  | <code>ProtectHome=false</code>                     | If any MariaDB files are in /home/                                                                                              |
| no option                                  | <code>PrivateDevices=false</code>                  | If any MariaDB storage references raw block devices                                                                             |
| no option                                  | <code>ProtectSystem=</code>                        | If any MariaDB write any files to anywhere under /boot, /usr or /etc                                                            |
| no option                                  | <code>TimeoutStartSec={time}</code>                | Service startup timeout. See <a href="#">Configuring the Systemd Service Timeout</a> .                                          |
| no option (see <a href="#">MDEV-9264</a> ) | <code>OOMScoreAdjust={priority}</code>             | e.g. -600 to lower priority of OOM killer for mysqld                                                                            |
| <code>open-files-limit</code>              | <code>LimitNOFILE={limit}</code>                   | Limit on number of open files. See <a href="#">Configuring the Open Files Limit</a> .                                           |
| <code>core-file-size</code>                | <code>LimitCORE={size}</code>                      | Limit on core file size. Useful when <a href="#">enabling core dumps</a> . See <a href="#">Configuring the Core File Size</a> . |
|                                            | <code>LimitMEMLOCK={size}</code><br>or<br>infinity | Limit on how much can be locked in memory.<br>Useful when <a href="#">large-pages</a> or <a href="#">memlock</a> is used        |
| <code>nice</code>                          | <code>Nice={nice value}</code>                     |                                                                                                                                 |
| <code>syslog</code>                        | <code>StandardOutput=syslog</code>                 | See <a href="#">Configuring MariaDB to Write the Error Log to Syslog</a> .                                                      |

|                              |                                                                                                                                                   |                                                                                                                            |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
|                              | <code>StandardError=syslog</code>                                                                                                                 |                                                                                                                            |
|                              | <code>SyslogFacility=daemon</code>                                                                                                                |                                                                                                                            |
|                              | <code>SyslogLevel=err</code>                                                                                                                      |                                                                                                                            |
| <code>syslog-tag</code>      | <code>SyslogIdentifier</code>                                                                                                                     |                                                                                                                            |
| <code>flush-caches</code>    | <code>ExecStartPre=/usr/bin/sync</code>                                                                                                           |                                                                                                                            |
|                              | <code>ExecStartPre=/usr/sbin/sysctl -q -w vm.drop_caches=3</code>                                                                                 |                                                                                                                            |
| <code>malloc-lib</code>      | <code>Environment=LD_PRELOAD=/path/to/library</code>                                                                                              |                                                                                                                            |
| <code>numa-interleave</code> | <code>NUMAPolicy=interleave</code>                                                                                                                | from systemd v243 onwards                                                                                                  |
|                              | or:<br><code>ExecStart=/usr/bin/numactl --interleave=all /usr/sbin/mysqld \${MYSQLD_OPTS} \${_WSREP_NEW_CLUSTER} \${_WSREP_START_POSITION}</code> | prepending<br><code>ExecStart=/usr/bin/numactl --interleave=all</code><br>to existing<br><code>ExecStart</code><br>setting |
| <code>no-auto-restart</code> | <code>Restart={exit-status}</code>                                                                                                                |                                                                                                                            |

Note: the

`systemd`

manual contains the official meanings for these options. The manual also lists considerably more options than the ones listed above.

There are other options and the

`mariadb-service-convert`

script will attempt to convert these as accurately as possible.

## Configuring the Systemd Service Timeout

MariaDB's

## systemd

unit file has a default startup timeout of about 90 seconds on most systems. If a service startup takes longer than this default startup timeout, then

systemd  
will assume that  
mysqld  
has failed to startup, which causes  
systemd  
to kill the  
mysqld  
process. To work around this, it can be changed by configuring the

### TimeoutStartSec

option for the  
systemd  
service.

A similar problem can happen when stopping the MariaDB service. Therefore, it may also be a good idea to set

### TimeoutStopSec

For example, you can reconfigure the MariaDB

systemd  
service to have an infinite timeout by executing one of the following commands:

If you are using  
systemd  
228 or older, then you can execute the following to set an infinite timeout:

```
sudo tee /etc/systemd/system/mariadb.service.d/timeoutsec.conf <<EOF
[Service]
TimeoutStartSec=0
TimeoutStopSec=0
EOF
sudo systemctl daemon-reload
```

Systemd 229 added the infinity option , so if you are using

systemd  
229 or later, then you can execute the following to set an infinite timeout:

```
sudo tee /etc/systemd/system/mariadb.service.d/timeoutsec.conf <<EOF
[Service]
TimeoutStartSec=infinity
TimeoutStopSec=infinity
EOF
sudo systemctl daemon-reload
```

Note that [systemd 236 added the EXTEND\\_TIMEOUT\\_USEC environment variable](#) that allows services to extend the startup timeout during long-running processes. On systems with systemd versions that support it, MariaDB uses this feature to extend the startup timeout during certain startup processes that can run long.

## Configuring the Open Files Limit

When using

systemd  
, rather than setting the open files limit by setting the

### open-files-limit

option for  
mysqld\_safe  
or the

### open\_files\_limit

system variable, the limit can be changed by configuring the

#### `LimitNOFILE`

option for the MariaDB  
systemd  
service. The default is set to  
`LimitNOFILE=16364`  
in  
`mariadb.service`  
.

For example, you can reconfigure the MariaDB

systemd  
service to have a larger limit for open files by executing the following commands:

```
sudo tee /etc/systemd/system/mariadb.service.d/limitnofile.conf <<EOF
[Service]
LimitNOFILE=infinity
EOF
sudo systemctl daemon-reload
```

An important note is that setting

`LimitNOFILE=infinity`  
doesn't actually set the open file limit to *infinity*.

In

systemd  
234 and later, setting  
`LimitNOFILE=infinity`  
actually sets the open file limit to the value of the kernel's  
`fs.nr_open`  
parameter. Therefore, in these  
systemd  
versions, you may have to change this parameter's value.

The value of the

`fs.nr_open`  
parameter can be changed permanently by setting the value in

#### `/etc/sysctl.conf`

and restarting the server.

The value of the

`fs.nr_open`  
parameter can be changed temporarily by executing the

#### `sysctl`

utility. For example:

```
sudo sysctl -w fs.nr_open=1048576
```

In

systemd  
233 and before, setting  
`LimitNOFILE=infinity`  
actually sets the open file limit to  
65536  
. See [systemd issue #6559](#) for more information. Therefore, in these  
systemd  
versions, it is not generally recommended to set  
`LimitNOFILE=infinity`  
. Instead, it is generally better to set  
`LimitNOFILE`  
to a very large integer. For example:

```
sudo tee /etc/systemd/system/mariadb.service.d/limitnofile.conf <<EOF
[Service]

LimitNOFILE=1048576
EOF
sudo systemctl daemon-reload
```

## Configuring the Core File Size

When using

systemd  
, if you would like to [enable core dumps](#) , rather than setting the

[core-file-size](#)

option for  
mysqld\_safe  
, the limit can be changed by configuring the

[LimitCORE](#)

option for the MariaDB  
systemd  
service. For example, you can reconfigure the MariaDB  
systemd  
service to have an infinite size for core files by executing the following commands:

```
sudo tee /etc/systemd/system/mariadb.service.d/limitcore.conf <<EOF
[Service]

LimitCORE=infinity
EOF
sudo systemctl daemon-reload
```

## Configuring MariaDB to Write the Error Log to Syslog

When using

systemd  
, if you would like to redirect the [error log](#) to the [syslog](#) , then that can easily be done by doing the following:

- Ensure that

[log\\_error](#)

system variable is **not** set.

- Set

[StandardOutput=syslog](#)

- Set

[StandardError=syslog](#)

- Set

[SyslogFacility=daemon](#)

- Set

[SysLogLevel=err](#)

For example:

```

sudo tee /etc/systemd/system/mariadb.service.d/syslog.conf <<EOF
[Service]

StandardOutput=syslog
StandardError=syslog
SyslogFacility=daemon
SysLogLevel=err
EOF
sudo systemctl daemon-reload

```

If you have multiple instances of MariaDB, then you may also want to set

#### [SyslogIdentifier](#)

with a different tag for each instance.

## Configuring LimitMEMLOCK

If using [--memlock](#) or the iouring in InnoDB in [MariaDB 10.6](#), you will need to raise the LimitMEMLOCK limit.

```

sudo tee /etc/systemd/system/mariadb.service.d/limitcore.conf <<EOF
[Service]

LimitMEMLOCK=2M
EOF
sudo systemctl daemon-reload

```

Note: Prior to [MariaDB 10.1.10](#), the [--memlock](#) option could not be used with the MariaDB

systemd  
service.

## Configuring Access to Home Directories

MariaDB's [systemd](#) unit file restricts access to

/home  
,  
/root  
, and  
/run/user  
by default. This restriction can be overridden by setting the [ProtectHome](#) option to  
false  
for the MariaDB  
systemd  
service. This is done by creating a "drop-in" directory  
[/etc/systemd/system/mariadb.service.d/](#)  
and in it a file with a  
.conf  
suffix that contains the  
ProtectHome=false  
directive.

You can reconfigure the MariaDB

systemd  
service to allow access to  
/home  
by executing the following commands:

```

sudo mkdir /etc/systemd/system/mariadb.service.d
sudo tee /etc/systemd/system/mariadb.service.d/dontprotecthome.conf <<EOF
[Service]

ProtectHome=false
EOF
sudo systemctl daemon-reload

```

## Configuring the umask

When using

systemd  
, the default file permissions of

`mysqld`  
can be set by setting the  
`UMASK`  
and  
`UMASK_DIR`  
environment variables for the  
`systemd`  
service. For example, you can configure the MariaDB  
`systemd`  
service's umask by executing the following commands:

```
sudo tee /etc/systemd/system/mariadb.service.d/umask.conf <<EOF
[Service]

Environment="UMASK=0750"
Environment="UMASK_DIR=0750"
EOF
sudo systemctl daemon-reload
```

These environment variables do not set the umask. They set the default file system permissions. See [MDEV-23058](#) for more information.

Keep in mind that configuring the umask this way will only affect the permissions of files created by the

```
mysqld
process that is managed by
systemd
. The permissions of files created by components that are not managed by
systemd
, such as

mysql_install_db

, will not be effected.
```

See [Specifying Permissions for Schema \(Data\) Directories and Tables](#) for more information.

## Configuring the data directory

When doing a standard binary tarball install the datadir will be under `/usr/local/data`. The `systemd` service file makes the whole `/usr` directory tree write protected via

```
# Prevent writes to /usr, /boot, and /etc
ProtectSystem=full
```

though. So when just copying the distributed service file a tarball install will not start up, complaining e.g. about

```
[Warning] Can't create test file /usr/local/.../data/ubuntu-focal.lower-test
[ERROR] mariadbd: File '/usr/local/.../data/aria_log_control' not found (Errcode: 30 "Read-only file system")
[ERROR] mariadbd: Got error 'Can't open file' when trying to use aria control file '/usr/local/.../data/aria_log_control'
```

So when using a data directory under `/usr/local` that specific directory needs to be made writable for the service using the `ReadWritePaths` setting:

```
sudo tee /etc/systemd/system/mariadb.service.d/datadir.conf <<EOF
[Service]
ReadWritePaths=/usr/local/mysql/data
EOF
sudo systemctl daemon-reload
```

## Systemd Socket Activation

MariaDB starting with [10.6.0](#)

MariaDB can use `systemd`'s socket activation.

This is an on-demand service for MariaDB that will activate when required.

Systemd socket activation uses a

`mariadb.socket`  
definition file to define a set of UNIX and TCP sockets. Systemd will listen on these sockets, and when they are connected to, systemd will start the

`mariadb.service`  
and hand over the socket file descriptors for MariaDB to process the connection.

MariaDB remains running at this point and will have all sockets available and process connections exactly like it did before 10.6.

When MariaDB is shut down, the systemd

`mariadb.socket`  
remains active, and a new connection will restart the  
`mariadb.service`

## Using Systemd Socket Activation

To use MariaDB systemd socket activation, instead of enabling/starting

`mariadb.service`

,

`mariadb.socket`  
is used instead.

So the following commands work exactly like the

`mariadb.service`  
equivalents.

```
systemctl start mariadb.socket
systemctl enable mariadb.socket
```

These files alone only contain the UNIX and TCP sockets and basic network connection information to which will be listening for connections.

`@mariadb`

is a UNIX abstract socket, which means it doesn't appear on the filesystem. Connectors based on MariaDB Connector/C will be able to connect with these by using the socket name directly, provided the higher level implementation doesn't try to test for the file's existence first. Some connectors like PHP use mysqlnd that is a pure PHP implementation and as such will only be able to connect to on filesystem UNIX sockets.

With systemd activated sockets there is only a file descriptor limit on the number of listening sockets that can be created.

## When to Use Systemd Socket Activation

A common use case for systemd socket activated MariaDB is when there needs to be a quick boot up time. MariaDB needs to be ready to run, but it doesn't need to be running.

The ideal use case for systemd socket activation for MariaDB is for infrastructure providers running many multiple instances of MariaDB, where each instance is dedicated for a user.

## Downsides to Using Systemd Socket Activation

From the time the connection occurs, the client is going to be waiting until MariaDB has fully initialized before MariaDB can process the awaiting connection. If MariaDB was previously hard shutdown and needs to perform an extensive InnoDB rollback, then the activation time may be larger than the desired wait time of the client connection.

## Configuring Systemd Socket Activation

When MariaDB is run under systemd socket activation, the usual `socket`, `port`, and `backlog` system variables are ignored, as these settings are contained within the systemd socket definition file.

There is no configuration required in MariaDB to use MariaDB under socket activation.

The systemd options available are from the

[systemd documentation](#)

, however

[ListenStream](#)

and

[BackLog](#)

would be the most common configuration options.

As MariaDB isn't creating these sockets, the sockets don't need to be created with a

```
mysql
user. The sockets MariaDB may end up listening to under systemd socket activation, it may have not had the privileges to create itself.
```

Changes to the default

```
mariadb.socket
can be made in the same way as services,
systemctl edit mariadb.socket
, or using
/etc/systemd/system/mariadb.socket.d/someconfig.conf
files.
```

## Extra Port

A systemd socket can be configured as an

```
extra_port
, by using the
FileDescriptorName=extra
in the
.socket
file.
```

The

```
mariadb-extra.socket
is already packaged and ready for use.
```

## Multi-instance socket activation

```
mariadb@.socket
is MariaDB's packaged multi-instance definition. It creates multiple UNIX sockets based on the socket file started.
```

Starting

```
mariadb@bob.socket
will use the
mariadb@.socket
definition with
%I
within the definition replaced with "bob".
```

When something connects to a socket defined there, the

```
mariadb@bob.service
will be started.
```

## Systemd Socket Activation for Hosting Service Providers

A systemd socket activation service with multi-instance can provide an on-demand per user access to a hosting service provider's dedicated database. "User", in this case, refers to the customer of the hosting service provider.

## End User Benefits

This provides the following benefits for the user:

- Each user has their own dedicated instance with the following benefits:
  - The instance is free from the database contention of neighbors on MariaDB shared resources (table cache, connections, etc)
  - The user is free to change their own configuration of MariaDB, within the limits and permissions of the service provider.
  - Database service level backups, like mariabackup, are now directly available.
  - A user can install their own plugins.
  - The user can run a different database version to their neighbors.
  - If a user's neighbor triggers a fault in the server, the user's instance isn't affected.
- The database runs as their unix user in the server facilitating:
  - User can directly migrate their MariaDB data directory to a different provider.
  - The user's data is protected from other users on a kernel level.

## Hosting Service Provider Benefits

In addition to providing user benefits as a sales item, the following are additional benefits for the hosting service provider compared to a monolith service:

- Without passwords for the database, while still having security, support may be easier.

- When a user's database isn't active, there is no resource usage, only listening file descriptors by systemd.
- The socket activation transparently, with a minor startup time, starts the service as required.
- When the user's database hasn't had any activity after a time, it will deactivate ( [MDEV-25282](#) ).
- Planned enhancements in InnoDB provide:
  - an on-demand consumption of memory ( [MDEV-25340](#) ).
  - a proactive reduction in memory ( [MDEV-25341](#) ).
  - a memory resource pressure reduction in memory use ( [MDEV-24670](#) ).
- The service provider can still cap the user's database memory usage in a ulimit way that a user cannot override in settings.
- The service provider may choose a CPU/memory/IO based billing to the user on Linux cgroup accounting rather than the available compared to the rather limited options in

#### [CREATE USER](#)

- Because a user's database will shutdown when inactive, a database upgrade on the server will not take effect for the user until it passively shuts down, restarts, and then gets reactivated hence reducing user downtime..

## Downsides to the Hosting Service Provider

The extra memory used by more instances. This is mitigated by the on-demand activation. The deactivation when idle, and improved InnoDB memory management.

With plenty of medium size database servers running, the Linux OOM kill has the opportunity to kill off only a small number of database servers running rather than everyone.

## Example on configuration Items for a per user, systemd socket activitated multi-instance service

From a server perspective the operation would be as follows;

To make the socket ready to connect and systemd will be listening to the socket:

```
# systemctl start mariadb@username.socket
# systemctl start mariadb-extra@username.socket
```

To enable this on reboot (the same way as a systemd service):

```
# systemctl enable mariadb@username.socket
# systemctl enable mariadb-extra@username.socket
```

## A MariaDB Template File

A global template file. Once installed as a user's

\$HOME/.my.cnf  
file, it will become the default for many applications, and the MariaDB server itself.

```
# cat /etc/my.cnf.templ
[client]
socket=/home/USER/mariadb.sock

[client-server]
user=USER

[mariadb]
datadir=/home/USER/mariadb-database
```

## Custom Configuration for the Multiinstance Service

This extends/modifies the MariaDB multi-instance service.

The features of this extension are:

- that it will auto-create configuration file for user applications
- It will install the database on first service start
- auth-root-\*
  - in [mariadb-install-db](#) means that the user is their own privileged user with unix socket authentication active. This means non-that user cannot access another user's service, even with access to the unix socket(s). For more information see # [unix socket authentication security](#)
- If the MariaDB version was upgraded, the upgrade changes are made automatically
-

`LimitData`  
places a hard upper limit so the user doesn't exceed a portion of the server resources

```
# cat /etc/systemd/system/mariadb@.service.d/user.conf
[Service]
User=%I
ProtectHome=false

Environment=MYSQLD_MULTI_INSTANCE="--defaults-file=/home/%I/.my.cnf"

ExecStartPre=
ExecStartPre=/bin/sh -c "[ -f /home/%I/.my.cnf ] || sed -e \"s/USER/%I/g\" /etc/my.cnf.templ > /home/%I/.my.cnf"
ExecStartPre=mkdir -p /home/%I/mariadb-datarir
ExecStartPre=/usr/bin/mariadb-install-db $MYSQLD_MULTI_INSTANCE --rpm \
    --auth-root-authentication-method=socket --auth-root-socket-user=%I
ExecStartPost=/usr/bin/mariadb-upgrade $MYSQLD_MULTI_INSTANCE

# To limit user based tuning
LimitData=768M
# For io_uring use by innodb on < 5.12 kernels
LimitMEMLOCK=1M
```

## Custom Configuration for the Multi-instance Socket

This extends/modifies the MariaDB socket definition to be per user.

Create sockets based on the user of the instance (

`%I`

). Permissions are only necessary in the sense that the user can connect to them. It won't matter to the server. Access control is enforced within the server, however if the user web services are run as the user,

`Mode=777`

can be reduced.

`@mariadb-%I`

is an abstract unix socket not on the filesystem. It may help if a user is in a chroot. Not all applications can connect to abstract sockets.

```
# cat /etc/systemd/system/mariadb@.socket.d/user.conf
[Socket]
SocketUser=%I
SocketMode=777
ListenStream=
ListenStream=@mariadb-%I
ListenStream=/home/%I/mariadb.sock
```

The extra socket provides the user the ability to access the server when all max-connections are used:

```
# cat /etc/systemd/system/mariadb-extra@.socket.d/user.conf
[Socket]
SocketUser=%I
SocketMode=777
ListenStream=
ListenStream=@mariadb-extra-%I
ListenStream=/home/%I/mariadb-extra.sock
```

## Systemd Journal

systemd  
has its own logging system called the  
systemd  
journal. The  
systemd  
journal contains information about the service startup process. It is a good place to look when a failure has occurred.

The MariaDB

systemd  
service's journal can be queried by using the

`journalctl`

command. For example:

```
$ sudo journalctl n 20 -u mariadb.service
-- Logs begin at Fri 2019-01-25 13:49:04 EST, end at Fri 2019-01-25 18:07:02 EST. --
Jan 25 13:49:15 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Starting MariaDB 10.1.37 database server...
Jan 25 13:49:16 ip-172-30-0-249.us-west-2.compute.internal mysqld[2364]: 2019-01-25 13:49:16 140547528317120 [Note] /usr/sbin/my
Jan 25 13:49:17 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Started MariaDB 10.1.37 database server.
Jan 25 18:06:42 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Stopping MariaDB 10.1.37 database server...
Jan 25 18:06:44 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Stopped MariaDB 10.1.37 database server.
Jan 25 18:06:57 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Starting MariaDB 10.1.37 database server...
Jan 25 18:08:32 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: mariadb.service start-pre operation timed out. Terminatin
Jan 25 18:08:32 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Failed to start MariaDB 10.1.37 database server.
Jan 25 18:08:32 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: Unit mariadb.service entered failed state.
Jan 25 18:08:32 ip-172-30-0-249.us-west-2.compute.internal systemd[1]: mariadb.service failed.
```

## Converting mysqld\_safe Options to Systemd Options

`mariadb-service-convert`

is a script included in many MariaDB packages that is used by the package manager to convert

`mysqld_safe`

options to

systemd

options. It reads any explicit settings in the

[mysqld\_safe]

option group from option files , and its output is directed to

/etc/systemd/system/mariadb.service.d/migrated-from-my.cnf-settings.conf

. This helps to keep the configuration the same when upgrading from a version of MariaDB that does not use

systemd

to one that does.

Implicitly high defaults of

`open-files-limit`

may be missed by the conversion script and require explicit configuration. See [Configuring the Open Files Limit](#) .

## Known Issues

### 2.1.6.15 sysVinit

#### Contents

- 1. [Interacting with the MariaDB Server Process](#)
- 1. [Starting the MariaDB Server Process on Boot](#)
- 2. [Starting the MariaDB Server Process](#)
- 3. [Stopping the MariaDB Server Process](#)
- 4. [Restarting the MariaDB Server Process](#)
- 5. [Checking the Status of the MariaDB Server Process](#)
- 2. [Manually Installing mysql.server with SysVinit](#)
- 3. [SysVinit and Galera Cluster](#)
  - 1. [Bootstrapping a New Cluster](#)

`sysVinit` is one of the most common service managers. On systems that use `sysVinit` , the

`mysql.server`

script is normally installed to

/etc/init.d/mysql

## Interacting with the MariaDB Server Process

The service can be interacted with by using the

```
service
```

command.

## Starting the MariaDB Server Process on Boot

On RHEL/CentOS and other similar distributions, the

```
chkconfig
```

command can be used to enable the MariaDB Server process at boot:

```
chkconfig --add mysql  
chkconfig --level 345 mysql on
```

On Debian and Ubuntu and other similar distributions, the

```
update-rc.d
```

command can be used:

```
update-rc.d mysql defaults
```

## Starting the MariaDB Server Process

```
service mysql start
```

## Stopping the MariaDB Server Process

```
service mysql stop
```

## Restarting the MariaDB Server Process

```
service mysql restart
```

## Checking the Status of the MariaDB Server Process

```
service mysql status
```

## Manually Installing mysql.server with SysVinit

If you install MariaDB from [source](#) or from a [binary tarball](#) that does not install

```
mysql.server
```

automatically, and if you are on a system that uses [sysVinit](#), then you can manually install `mysql.server` with [sysVinit](#). See [mysql.server: Manually Installing with SysVinit](#) for more information.

## SysVinit and Galera Cluster

### Bootstrapping a New Cluster

When using [Galera Cluster](#) with sysVinit, the first node in a cluster has to be started with

```
service mysql bootstrap
```

. See [Getting Started with MariaDB Galera Cluster: Bootstrapping a New Cluster](#) for more information.

## 2.1.6.16 mariadb-admin

MariaDB starting with 10.4.6

From [MariaDB 10.4.6](#) ,  
mariadb-admin  
is a symlink to  
mysqladmin  
, the administration program for the mysqld daemon.

MariaDB starting with [10.5.2](#)

From [MariaDB 10.5.2](#) ,  
mariadb-admin  
is the name of the administration program for the mysqld daemon, with  
mysqladmin  
a symlink .

See [mysqladmin](#) for details.

## 2.1.6.17 mariadb

MariaDB starting with [10.4.6](#)

From [MariaDB 10.4.6](#) ,  
mariadb  
is a symlink to [mysql](#) , the MariaDB server.

MariaDB starting with [10.5.2](#)

From [MariaDB 10.5.2](#) ,  
mariadb  
is the name of the server, with  
mysqld  
a symlink .

See [mysqld](#) for details.

## 2.1.6.18 mariadb-multi

MariaDB starting with [10.4.6](#)

From [MariaDB 10.4.6](#) ,  
mariadb-multi  
is a symlink to  
mysqld\_multi  
, the wrapper designed to manage several mysqld processes running on the same host.

MariaDB starting with [10.5.2](#)

From [MariaDB 10.5.2](#) ,  
mariadb-multi  
is the name of the server, with  
mysqld\_multi  
a symlink .

See [mysqld\\_multi](#) for details.

## 2.1.6.19 mariadb-safe

MariaDB starting with [10.4.6](#)

From [MariaDB 10.4.6](#) ,  
mariadb-safe  
is a symlink to  
mysqld\_safe  
, the tool for starting mysqld on Linux and Unix distributions that do not support [systemd](#) .

MariaDB starting with [10.5.2](#)

From [MariaDB 10.5.2](#) ,

`mariadb-safe`  
is the name of the binary, with  
`mysqld_safe`  
a symlink .

See [mysqld\\_safe](#) for details.

## 2.1.7 MariaDB Performance & Advanced Configurations

Articles of how to setup your MariaDB optimally on different systems



### Fusion-io

*This category contains information about Fusion-io support in MariaDB*



#### Atomic Write Support

*Enabling atomic writes to speed up InnoDB on selected SSD cards.*



#### Configuring Linux for MariaDB

*Linux kernel settings IO scheduler For optimal IO performance running a da...*



#### Configuring MariaDB for Optimal Performance

*How to get optimal performance.*



#### Configuring Swappiness

*Setting Linux swappiness.*

There are [5 related questions](#).

### 2.1.7.1 Fusion-io

This category contains information about Fusion-io support in MariaDB



#### Fusion-io Introduction

*Fusion-io PCIe SSD cards to speed up MariaDB.*



#### Atomic Write Support

*Enabling atomic writes to speed up InnoDB on selected SSD cards.*



#### MariaDB 10.0.15 Fusion-io Release Notes

*Status: | Release Date: 12 Dec 2014*



#### MariaDB 10.0.15 Fusion-io Changelog

*Status: | Release Date: 12 Dec 2014*



#### InnoDB Page Flushing

*Configuring when and how InnoDB flushes dirty pages to disk.*

### 2.1.7.1.1 Fusion-io Introduction

#### Contents

1. [Use Cases](#)
2. [Atomic Writes](#)
3. [Future Suggested Development](#)
4. [Settings For Best Performance](#)
5. [Example Configuration](#)
6. [Card Models](#)
7. [Additional Software](#)
8. [See Also](#)

Fusion-io develops PCIe based NAND flash memory cards and related software that can be used to speed up MariaDB databases.

The ioDrive branded products can be used as block devices (super-fast disks) or to extend basic DRAM memory. ioDrive is deployed by installing it on an x86 server and then installing the card driver under the operating system. All main line 64-bit operating systems and hypervisors are supported: RHEL, CentOS, SuSe, Debian, OEL etc. and VMWare, Microsoft Windows/Server etc. Drivers and their features are constantly developed further.

ioDrive cards support software RAID and you can combine two or more physical cards into one logical drive. Through ioMemory SDK and its APIs, one can integrate and enable more thorough interworking between your own software and the cards - and cut latency.

The key differentiator between a Fusion-io and a legacy SSD/HDD is the following: **A Fusion-io card is connected directly on the system bus (PCIe)**, this enables high data transfer throughput (1.5 GB/s, 3.0 GB/s or 6GB/s) and the fast direct memory access (DMA) method can be used to transfer data.

The ATA/SATA protocol stack is omitted and therefore [latency is cut short](#). Fusion-io performance is dependent on server speed: the faster processors and the newer PCIe-bus version you have, the better is the ioDrive performance. [Fusion-io memory is non-volatile](#)., in other words, data remains on the card even when the server is powered off.

## Use Cases

1. You can start by using ioDrive for database files that need heavy random access.
2. Whole database on ioDrive.
3. In some cases, Fusion-io devices allow for atomic writes, which allows the server to safely disable the [doublewrite buffer](#).
4. Use ioDrive as a write-through read cache. This is possible on server level with Fusion-io directCache software or in VMware environments using ioTurbine software or the ioCache bundle product. Reads happen from ioDrive and all writes go directly to your SAN or disk.
5. Highly Available shared storage with ION. Have two different hosts, Fusion-io cards in them and share/replicate data with Fusion-io's ION software.
6. The luxurious Platinum setup: [MariaDB Galera Cluster](#) running on Fusion-io SLC cards on several hosts.

## Atomic Writes

Starting with [MariaDB 5.5.31](#), MariaDB Server supports atomic writes on Fusion-io devices that use the NVMFS (formerly called DirectFS) file system. Unfortunately, NVMFS was never offered under 'General Availability', and SanDisk declared that NVMFS would reach end-of-life in December 2015. Therefore, NVMFS support is no longer offered by SanDisk.

MariaDB Server does not currently support atomic writes on Fusion-io devices with any other file systems.

See [atomic write support](#) for more information about MariaDB Server's atomic write support.

## Future Suggested Development

- Extend InnoDB disk cache to be stored on Fusion-io acting as extended memory.

## Settings For Best Performance

Fusion-io memory can be formatted with different sector size of either 512 or 4096 bytes. Bigger sectors are expected to be faster, but only if I/O is done in blocks of 4KB or multiples of that. Speaking of MariaDB: if only InnoDB data files are stored in Fusion-io memory, all I/O is done in blocks of 16K and thus 4K sector size can be used. If the InnoDB redo log (I/O block size: 512 bytes) goes to the same Fusion-io memory, then short sectors should be used.

Note: XtraDB has the experimental feature of an increased InnoDB log block size of 4K. If this is enabled, then both redo log I/O and page I/O in InnoDB will match a sector size of 4K.

As of file systems: currently XFS is expected to yield the best performance with MariaDB. However depending on the exact kernel version and version of XFS code in use, one might be affected by [a bug that severely limits XFS performance in concurrent environments](#). This has been fixed in kernel versions above 3.5 or RHEL6 kernels kernel-2.6.32-358 or later (because of [bug 807503 being fixed](#)).

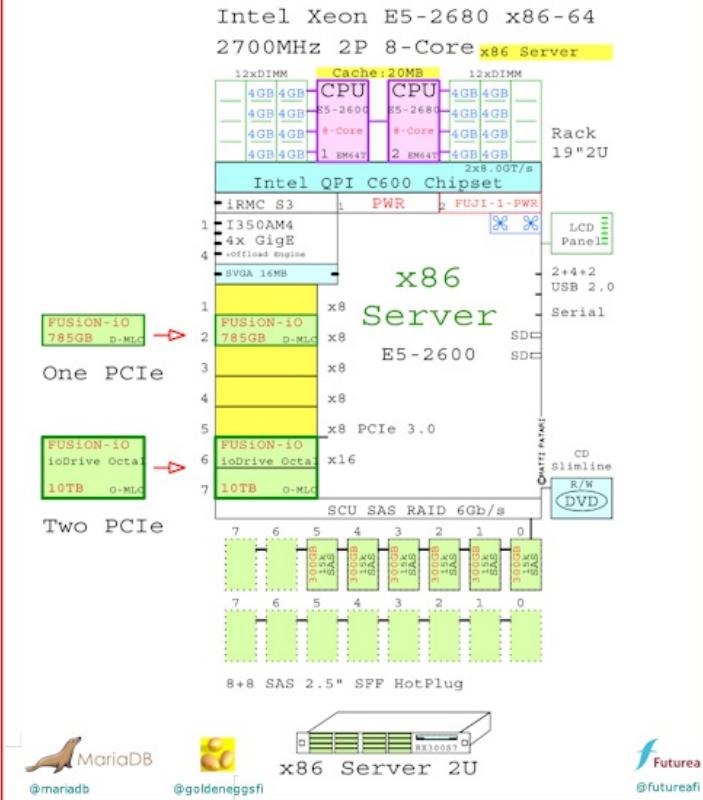
For the pitbull machine where I have run such tests, ext4 was faster than xfs for 32 or more threads:

- up to 8 threads xfs was few percent faster (10% on average).
- at 16 threads it was a draw (2036 tps vs. 2070 tps).
- at 32 threads ext4 was 28% faster (2345 tps vs. 1829 tps).
- at 64 threads ext4 was even 47% faster (2362 tps vs. 1601 tps).
- at higher concurrency ext4 lost its bite, but was still constantly better than xfs.

Those numbers are for spinning disks. I guess for Fusion-io memory the XFS numbers will be even worse.

## Example Configuration

Example configuration of MariaDB with Fusion-io cards, courtesy of [www.goldeneeggs.fi](http://www.goldeneeggs.fi)



## Card Models

There are several card models. ioDrive is older generation, ioDrive2 is newer. SLC sustains more writes. MLC is good enough for normal use.

1. ioDrive2, capacities per card 365GB, 785GB, 1.2TB with MLC. 400GB and 600GB with SLC, performance up to 535000 IOPS & 1.5GB/s bandwidth
2. ioDrive2 Duo, capacities per card 2.4TB MLC and 1.2TB SLC, performance up to 935000 IOPS & 3.0GB/s bandwidth
3. ioDrive, capacities per card 320GB, 640GB MLC and 160GB, 320GB SLC, performance up to 145000 IOPS & 790MB/s bandwidth
4. ioDrive Duo, capacities per card 640GB, 1.28TB MLC and 320GB, 640GB SLC, performance up to 285000 IOPS & 1.5GB/s bandwidth
5. ioDrive Octal, capacities per card 5TB and 10TB MLC, performance up to 1350000 IOPS & 6.7GB/s bandwidth
6. ioFX, a 420GB QDP MLC workstation product, 1.4GB/s bandwidth
7. ioCache, a 600GB MLC card with ioTurbine software bundle that can be used to speed up VMware based virtual hosts.
8. ioScale, 3.2TB card, building block to enable all-flash data center build out in hyperscale web and cloud environments. Product has been developed in co-operation with Facebook.

## Additional Software

- directCache - transforms ioDrive to work as a read cache in your server. Writes go directly to your SAN
- ioTurbine - read cache software for VMware
- ION - transforms ioDrive into a shareable storage
- ioSphere - software to manage and monitor several ioDrives

## See Also

- [FusionIO atomic-series devices](#)

### 2.1.7.1.2 Atomic Write Support

## Contents

1. Partial Write Operations
2. innodb\_doublewrite - an Imperfect Solution
3. Atomic Write - a Faster Alternative to innodb\_doublewrite
4. Enabling Atomic Writes from MariaDB 10.2
5. Enabling Atomic Writes in MariaDB 5.5 to MariaDB 10.1
  1. About innodb\_use\_atomic\_writes (in MariaDB 5.5 to MariaDB 10.1)
6. Devices that Support Atomic Writes with MariaDB

## Partial Write Operations

When Innodb writes to the filesystem, there is generally no guarantee that a given write operation will be complete (not partial) in cases of a poweroff event, or if the operating system crashes at the exact moment a write is being done.

Without detection or prevention of partial writes, the integrity of the database can be compromised after recovery.

## innodb\_doublewrite - an Imperfect Solution

Since its inception, Innodb has had a mechanism to detect and ignore partial writes via the [InnoDB Doublewrite Buffer](#) (also innodb\_checksum can be used to detect a partial write).

Doublewrites, controlled by the [innodb\\_doublewrite](#) system variable, comes with its own set of problems. Especially on SSD, writing each page twice can have detrimental effects (write leveling).

## Atomic Write - a Faster Alternative to innodb\_doublewrite

A better solution is to directly ask the filesystem to provide an atomic (all or nothing) write guarantee. Currently this is only available on [a few SSD cards](#).

## Enabling Atomic Writes from MariaDB 10.2

When starting, [MariaDB 10.2](#) and beyond automatically detects if any of the supported SSD cards are used.

When opening an InnoDB table, there is a check if the tablespace for the table is [on a device that supports atomic writes](#) and if yes, it will automatically enable atomic writes for the table. If atomic writes support is not detected, the doublewrite buffer will be used.

One can disable atomic write support for all cards by setting the variable [innodb-use-atomic-writes](#) to

OFF  
in your my.cnf file. It's  
ON  
by default.

## Enabling Atomic Writes in MariaDB 5.5 to MariaDB 10.1

To use atomic writes instead of the doublewrite buffer, add:

```
innodb_use_atomic_writes = 1
```

to the my.cnf config file.

Note that atomic writes are only supported on [Fusion-io devices that use the NVMe file system](#) in these versions of MariaDB.

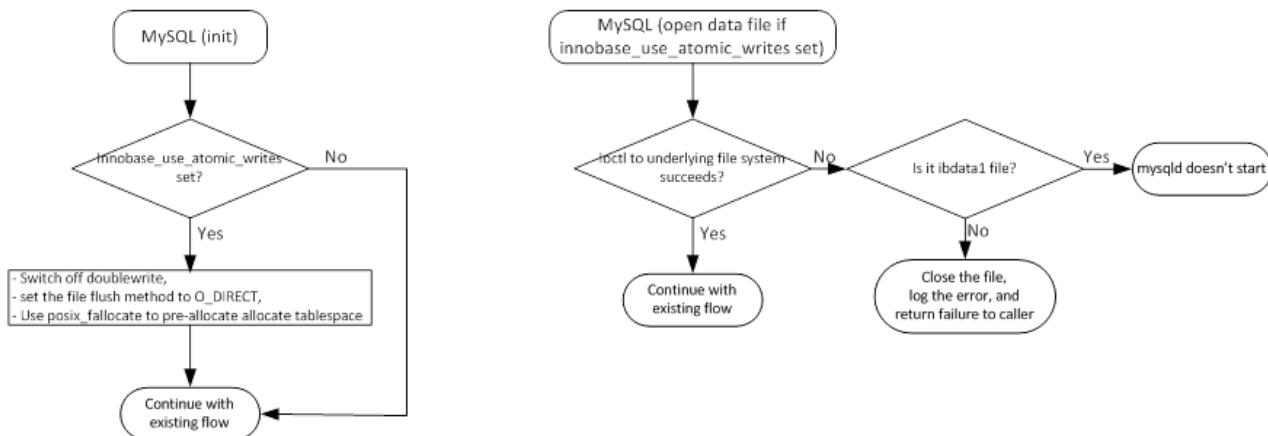
## About innodb\_use\_atomic\_writes (in MariaDB 5.5 to MariaDB 10.1 )

The following happens when atomic writes are enabled

- if [innodb\\_flush\\_method](#) is neither
  - O\_DIRECT
  - ,
  - ALL\_O\_DIRECT
  - , or
  - O\_DIRECT\_NO\_FSYNC
  - , it is switched to
  - O\_DIRECT

- `innodb_use_fallocate` is switched ON  
(files are extended using `posix_fallocate` rather than writing zeros behind the end of file)
- Whenever an InnoDB datafile is opened, a special `ioctl()` is issued to switch on atomic writes. If the call fails, an error is logged and returned to the caller. This means that if the system tablespace is not located on an atomic write capable device or filesystem, InnoDB/XtraDB will refuse to start.
- if `innodb_doublewrite` is set to ON,  
, `innodb_doublewrite` will be switched OFF  
and a message written to the error log.

Here is a flowchart showing how atomic writes work inside InnoDB:



## Devices that Support Atomic Writes with MariaDB

MariaDB currently supports atomic writes on the following devices:

- Fusion-io devices with the NVMS file system . MariaDB 5.5 and above.
- Shannon SSD . MariaDB 10.2 and above.

### 2.1.7.1.3 InnoDB Page Flushing

### 2.1.7.3 Configuring Linux for MariaDB

#### Contents

1. [Linux kernel settings](#)
  1. [IO scheduler](#)
2. [Resource Limits](#)
  - [Configuring the Open Files Limit](#)
  - [Configuring the Core File Size](#)
3. [Swappiness](#)

## Linux kernel settings

### IO scheduler

For optimal IO performance running a database we are using the `noop` scheduler. Recommended schedulers are `noop` and `deadline` . You can check your scheduler setting with:

```
cat /sys/block/${DEVICE}/queue/scheduler
```

For instance, it should look like this output:

```
cat /sys/block/sda/queue/scheduler
[noop] deadline cfq
```

## Resource Limits

### Configuring the Open Files Limit

By default, the system limits how many open file descriptors a process can have open at one time. It has both a soft and hard limit. On many systems, both the soft and hard limit default to 1024. On an active database server, it is very easy to exceed 1024 open file descriptors. Therefore, you may need to increase the soft and hard limits. There are a few ways to do so.

If you are using

```
mysqld_safe  
to start  
mysqld  
, then see the instructions at mysqld\_safe: Configuring the Open Files Limit .
```

If you are using

```
systemd  
to start  
mysqld  
, then see the instructions at systemd: Configuring the Open Files Limit .
```

Otherwise, you can set the soft and hard limits for the

```
mysql  
user account by adding the following lines to
```

```
/etc/security/limits.conf
```

```
:
```

```
mysql soft nofile 65535  
mysql hard nofile 65535
```

After the system is rebooted, the

```
mysql  
user should use the new limits, and the user's  
ulimit  
output should look like the following:
```

```
$ ulimit -Sn  
65535  
$ ulimit -Hn  
65535
```

### Configuring the Core File Size

By default, the system limits the size of core files that could be created. It has both a soft and hard limit. On many systems, the soft limit defaults to 0. If you want to [enable core dumps](#) , then you may need to increase this. Therefore, you may need to increase the soft and hard limits. There are a few ways to do so.

If you are using

```
mysqld_safe  
to start  
mysqld  
, then see the instructions at mysqld\_safe: Configuring the Core File Size .
```

If you are using

```
systemd  
to start  
mysqld  
, then see the instructions at systemd: Configuring the Core File Size .
```

Otherwise, you can set the soft and hard limits for the

```
mysql
user account by adding the following lines to
```

```
/etc/security/limits.conf
```

```
:
```

```
mysql soft core unlimited
mysql hard core unlimited
```

After the system is rebooted, the

```
mysql
user should use the new limits, and the user's
ulimit
output should look like the following:
```

```
$ ulimit -Sc
unlimited
$ ulimit -Hc
unlimited
```

## Swappiness

See [configuring swappiness](#).

### 2.1.7.4 Configuring MariaDB for Optimal Performance

#### Contents

1. [my.cnf Files](#)
2. [InnoDB & XtraDB Storage Engine](#)
3. [Aria Storage Engine](#)
4. [MyISAM](#)
5. [Lots of Connections](#)
  1. [A Lot of Fast Connections + Small Set of Queries + Disconnects](#)
  2. [Connecting From a Lot of Different Machines](#)
6. [See Also](#)
7. [External Links](#)

This article is to help you configure MariaDB for optimal performance.

Note that by default MariaDB is configured to work on a desktop system and should because of this not take a lot of resources. To get things to work for a dedicated server, you have to do a few minutes of work.

For this article we assume that you are going to run MariaDB on a dedicated server.

Feel free to update this article if you have more ideas!

## my.cnf Files

MariaDB is normally configured by editing the [my.cnf](#) file.

The following my.cnf example files were included with MariaDB until [MariaDB 10.3.0](#). If present, you can examine them to see more complete examples of some of the many ways to configure MariaDB and use the one that fits you best as a base. Note that these files are now quite outdated, so what was huge a few years ago may no longer be seen as such.

- my-small.cnf
- my-medium.cnf
- my-large.cnf
- my-huge.cnf

## InnoDB & XtraDB Storage Engine

InnoDB or XtraDB is normally the default storage engine with MariaDB.

- You should set [innodb\\_buffer\\_pool\\_size](#) to about 80% of your memory. The goal is to ensure that 80 % of your working set is in memory!

The other most important InnoDB variables are:

- [innodb\\_log\\_file\\_size](#)
- [innodb\\_flush\\_method](#)

- [innodb\\_thread\\_sleep\\_delay](#)

Some other important InnoDB variables:

- [innodb\\_adaptive\\_max\\_sleep\\_delay](#)
- [innodb\\_buffer\\_pool\\_instances](#)
- [innodb\\_buffer\\_pool\\_size](#)
- [innodb\\_max\\_dirty\\_pages\\_pct\\_lwm](#)
- [innodb\\_read\\_ahead\\_threshold](#)
- [innodb\\_thread\\_concurrency](#)

## Aria Storage Engine

- MariaDB uses by default the Aria storage engine for internal temporary files. If you have many temporary files, you should set [aria\\_pagecache\\_buffer\\_size](#) to a reasonably large value so that temporary overflow data is not flushed to disk. The default is 128M.

## MyISAM

- If you don't use MyISAM tables explicitly (true for most MariaDB 10.4+ users), you can set [key\\_buffer\\_size](#) to a very low value, like 64K.

## Lots of Connections

### A Lot of Fast Connections + Small Set of Queries + Disconnects

- If you are doing a lot of fast connections / disconnects, you should increase [back\\_log](#) and if you are running MariaDB 10.1 or below [thread\\_cache\\_size](#).
- If you have a lot (> 128) of simultaneous running fast queries, you should consider setting [thread\\_handling](#) to [pool\\_of\\_threads](#)

## Connecting From a Lot of Different Machines

- If you are connecting from a lot of different machines you should increase [host\\_cache\\_size](#) to the max number of machines (default 128) to cache the resolving of hostnames. If you don't connect from a lot of machines, you can set this to a very low value!

## See Also

- [MariaDB Memory Allocation](#)
- [Full List of MariaDB Options, System and Status Variables](#)
- [Server system variables](#)
- [mysqld options](#)
- [Performance schema](#) helps you understand what is taking time and resources.
- [Slow query log](#) is used to find queries that are running slow.
- [OPTIMIZE TABLE](#) helps you defragment tables.

## External Links

- <http://www.tocker.ca/2013/09/17/what-to-tune-in-mysql-56-after-installation.html>
- <http://www.percona.com/resources/technical-presentations/optimizing-mysql-configuration-percona-mysql-university-montevideo>

## 2.1.7.5 Configuring Swappiness

### Contents

1. [Why to Avoid Swapping](#)
2. [Setting Swappiness on Linux](#)
3. [Disabling Swap Altogether](#)

## Why to Avoid Swapping

Obviously, accessing swap memory from disk is far slower than accessing RAM directly. This is particularly bad on a database server because:

- MariaDB's internal algorithms assume that memory is not swap, and are highly inefficient if it is. Some algorithms are intended to avoid or delay disk IO, and use memory where possible - performing this with swap can be worse than just doing it on disk in the first place.
- Swap increases IO over just using disk in the first place as pages are actively swapped in and out of swap. Even something like removing a dirty page that is no longer going to be stored in memory, while designed to improve efficiency, will under a swap situation cost more IO.
- Database locks are particularly inefficient in swap. They are designed to be obtained and released often and quickly, and pausing to perform disk IO will have a serious impact on their usability.

The main way to avoid swapping is to make sure you have enough RAM for all processes that need to run on the machine. Setting the [system variables](#) too high can mean that under load the server runs short of memory, and needs to use swap. So understanding what settings to use and how these impact your server's memory usage is critical.

## Setting Swappiness on Linux

Linux has a swappiness setting which determines the balance between swapping out pages (chunks of memory) from RAM to a preconfigured swap space on the hard drive.

The setting is from 0 to 100, with lower values meaning a lower likelihood of swapping. The default is usually 60 - you can check this by running:

```
sysctl vm.swappiness
```

The default setting encourages the server to use swap. Since there probably won't be much else on the database server besides MariaDB processes to put into swap, you'll probably want to reduce this to zero to avoid swapping as much as possible. You can change the default by adding a line to the

```
sysctl.conf  
file (usually found in  
/etc/sysctl.conf  
).
```

To set the swappiness to zero, add the line:

```
vm.swappiness = 0
```

This normally takes effect after a reboot, but you can change the value without rebooting as follows:

```
sysctl -w vm.swappiness=0
```

Since RHEL 6.4, setting swappiness=0 more aggressively avoids swapping out, which increases the risk of OOM killing under strong memory and I/O pressure.

A low swappiness setting is recommended for database workloads. For MariaDB databases, it is recommended to set swappiness to a value of 1.

```
vm.swappiness = 1
```

## Disabling Swap Altogether

While some disable swap altogether, and you certainly want to avoid any database processes from using it, it can be prudent to leave some swap space to at least allow the kernel to fall over gracefully should a spike occur. Having emergency swap available at least allows you some scope to kill any runaway processes.

### 2.1.8 Troubleshooting Installation Issues

Articles relating to installation issues users might run into.



#### Troubleshooting Connection Issues

[Common problems when trying to connect to MariaDB.](#)



#### Installation issues on Windows

[Issues people have encountered when installing MariaDB on Windows](#)



#### Troubleshooting MariaDB Installs on Red Hat/CentOS

[Issues people have encountered when installing MariaDB on Red Hat / CentOS](#)



#### Installation issues on Debian and Ubuntu

[Solutions to different installation issues on Debian and Ubuntu](#)



#### What to Do if MariaDB Doesn't Start

[Troubleshooting MariaDB when it fails to start.](#)



#### Installing on an Old Linux Version

[Typical errors from using an incompatible MariaDB binary on a linux system](#)



#### Error: symbol mysql\_get\_server\_name, version libmysqlclient\_16 not defined

[Error from using MariaDB's mysql command-line client with MySQL's libmysqlclient.so](#)



## Installation Issues with PHP5

*PHP5 may give an error if used with the old connect method*

There are 12 related questions .

### 2.1.8.1 Troubleshooting Connection Issues

#### Contents

1. [Server Not Running in Specified Location](#)
2. [Unable to Connect from a Remote Location](#)
3. [Authentication Problems](#)
  1. [Problems Exporting Query Results](#)
  2. [Access to the Server, but not to a Database](#)
  3. [Option Files and Environment Variables](#)
4. [Unable to Connect to a Running Server / Lost root Password](#)
  5. [localhost and %](#)
4. [See Also](#)

If you are completely new to MariaDB and relational databases, you may want to start with the [MariaDB Primer](#). Also, make sure you understand the connection parameters discussed in the [Connecting to MariaDB](#) article.

There are a number of common problems that can occur when connecting to MariaDB.

#### Server Not Running in Specified Location

If the error you get is something like:

```
mysql -uname -p
ERROR 2002 (HY000): Can't connect to local MySQL server through
socket '/var/run/mysqld/mysqld.sock' (2 "No such file or directory")
```

or

```
mysql -uname -p --port=3307 --protocol=tcp
ERROR 2003 (HY000): Can't connect to MySQL server on 'localhost'
(111 "Connection refused")
```

the server is either not running, or not running on the specified port, socket or pipe. Make sure you are using the correct [host](#), [port](#), [pipe](#), [socket](#) and [protocol](#) options, or alternatively, see [Getting, Installing and Upgrading MariaDB](#), [Starting and Stopping MariaDB](#) or [Troubleshooting Installation Issues](#).

The socket file can be in a non-standard path. In this case, the

socket

option is probably written in the my.cnf file. Check that its value is identical in the [mysqld] and [client] sections; if not, the client will look for a socket in a wrong place.

If unsure where the Unix socket file is running, it's possible to find this out, for example:

```
netstat -ln | grep mysqld
unix 2 [ ACC ] STREAM LISTENING 33209505 /var/run/mysqld/mysqld.sock
```

#### Unable to Connect from a Remote Location

Usually, the MariaDB server does not by default accept connections from a remote client or connecting with tcp and a hostname and has to be configured to permit these.

```
(/my/maria-10.4) ./client/mysql --host=myhost --protocol=tcp --port=3306 test
ERROR 2002 (HY000): Can't connect to MySQL server on 'myhost' (115)
(/my/maria-10.4) telnet myhost 3306
Trying 192.168.0.11...
telnet: connect to address 192.168.0.11: Connection refused
(/my/maria-10.4) perror 115
OS error code 115: Operation now in progress
```

To solve this, see [Configuring MariaDB for Remote Client Access](#)

## Authentication Problems

Note that from [MariaDB 10.4.3](#), the [unix\\_socket authentication plugin](#) is enabled by default on Unix-like systems. This uses operating system credentials when connecting to MariaDB via the local Unix socket file. See [unix\\_socket authentication plugin](#) for instructions on connecting and on switching to password-based authentication as well as [Authentication from MariaDB 10.4](#) for an overview of the [MariaDB 10.4](#) changes..

Authentication is granted to a particular username/host combination.

```
user1'@'localhost'
, for example, is not the same as
user1'@'166.78.144.191'
```

. See the [GRANT](#) article for details on granting permissions.

Passwords are hashed with [PASSWORD](#) function. If you have set a password with the [SET PASSWORD](#) statement, or used [INSERT](#) or [UPDATE](#) to update the [permissions table](#) directly, the [PASSWORD](#) function must be used at the same time. For example,

```
SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass')
rather than just
SET PASSWORD FOR 'bob'@'%.loc.gov' = 'newpass'
;
```

If grant tables have been changed directly, the new passwords or authentication data will not immediately be active. A [FLUSH PRIVILEGES](#) statement, or the [flush-privileges mysqladmin](#) option must be run in order for the changes to take effect.

## Problems Exporting Query Results

If you can run regular queries, but get an authentication error when running the [SELECT ... INTO OUTFILE](#), [SELECT ... INTO DUMPFILE](#) or [LOAD DATA INFILE](#) statements, you do not have permission to write files to the server. This requires the FILE privilege. See the [GRANT](#) article.

## Access to the Server, but not to a Database

If you can connect to the server, but not to a database, for example:

```
USE test;
ERROR 1044 (42000): Access denied for user 'ian'@'localhost' to database 'test'
```

or can connect to a particular database, but not another, for example

```
mysql -u name db1
works but not
mysql -u name db2
, you have not been granted permission for the particular database. See the GRANT article.
```

## Option Files and Environment Variables

It's possible that option files or environment variables may be providing incorrect connection parameters. Check the values provided in any option files read by the client you are using (see [mysqld Configuration Files and Groups](#) and the documentation for the particular client you're using - see [Clients and Utilities](#) ).

Option files can usually be suppressed with

```
no-defaults
option, for example:
```

```
mysqlimport --no-defaults ...
```

## Unable to Connect to a Running Server / Lost root Password

If you are unable to connect to a server, for example because you have lost the root password, you can start the server without using the privilege tables by running the

```
--skip-grant-tables
```

option, which gives users full access to all tables. You can then run [FLUSH PRIVILEGES](#) to resume using the grant tables, followed by [SET PASSWORD](#) to change the password for an account.

localhost and %

You may have created a user with something like:

```
CREATE USER melisa identified by 'password';
```

This creates a user with the '%' wildcard host.

```
select user,host from mysql.user where user='melisa';
+-----+-----+
| user | host |
+-----+-----+
| melisa | %   |
+-----+-----+
```

However, you may still be failing to login from localhost. Some setups create anonymous users, including localhost. So the following records exist in the user table:

```
select user,host from mysql.user where user='melisa' or user='';
+-----+-----+
| user | host |
+-----+-----+
| melisa | %   |
|          | localhost |
+-----+-----+
```

Since you are connecting from localhost, the anonymous credentials, rather than those for the 'melisa' user, are used. The solution is either to add a new user specific to localhost, or to remove the anonymous localhost user.

## See Also

- [CREATE USER](#)
- [GRANT](#)
- [Authentication from MariaDB 10.4](#)
- [Authentication from MariaDB 10.4 video tutorial](#)

## 2.1.8.2 Installation issues on Windows

### Contents

1. [MariaDB 10.4.13](#)
2. [Unsupported Versions of Windows](#)
3. [MariaDB 5.2.5 and earlier](#)
  1. [On Windows Vista/7 , changes to database or my.ini are not persistent, when mysqld.exe is run from the command line.](#)
  4. [Systems with User Account Control](#)

## MariaDB 10.4.13

[MariaDB 10.4.13](#) may not start on Windows. See [MDEV-22555](#).

To resolve this, download, click and install [https://aka.ms/vs/16/release/vc\\_redist.x64.exe](https://aka.ms/vs/16/release/vc_redist.x64.exe) and then install 10.4.13.

## Unsupported Versions of Windows

Recent versions of MariaDB may not install on unsupported Windows versions. See [Deprecated Package Platforms](#) to find the final supported versions.

## MariaDB 5.2.5 and earlier

### On Windows Vista/7 , changes to database or my.ini are not persistent, when mysqld.exe is run from the command line.

The reason for this behavior is Vista/Win7 file system redirection. Writes to protected locations (in this case a subdirectory of Program Files) are redirected to the user's so-called "Virtual Store".

Workarounds:

- Run mysqld.exe as service. See answer [here](#) on how to create a MariaDB service.

- Run mysqld.exe from the [elevated command prompt](#).
- [Change the ACL](#) of the data directory and add full control for the current user.

The Windows installer for [MariaDB 5.2.6](#) and higher will set the data directory ACL to include full access rights for the user who runs the setup to prevent this issue from happening.

## Systems with User Account Control

Running

`mysql_install_db.exe`  
from a standard command prompt might cause the error:

```
FATAL ERROR: OpenSCManager failed
```

To get rid of it, use the elevated command prompt, for example on Windows 7 start it via 'Run as administrator' option.

## 2.1.8.3 Troubleshooting MariaDB Installs on Red Hat/CentOS

The following article is about different issues people have encountered when installing MariaDB on Red Hat / CentOS.

It is highly recommended to [install with yum](#) where possible.

In Red Hat / CentOS it is also possible to install a [RPM](#) or a [tar ball](#). The RPM is the preferred version, except if you want to install many versions of MariaDB or install MariaDB in a non standard location.

## Replacing MySQL

If you removed an MySQL RPM to install MariaDB, note that the MySQL RPM on uninstall renames `/etc/my.cnf` to `/etc/my.cnf.rpmsave`.

After installing MariaDB you should do the following to restore your configuration options:

```
mv /etc/my.cnf.rpmsave /etc/my.cnf
```

## Unsupported configuration options

If you are using any of the following options in your `/etc/my.cnf` or other `my.cnf` file you should remove them. This is also true for MySQL 5.1 or newer:

```
skip-bdb
```

## See also

- [Installing with yum \(recommended\)](#)
- [Installing MariaDB RPM Files](#)
- [Checking MariaDB RPM Package Signatures](#)

## 2.1.8.4 Installation issues on Debian and Ubuntu

Solutions to different installation issues on Debian and Ubuntu



### Differences in MariaDB in Debian (and Ubuntu)

[MariaDB when installed from the Debian repos has a number of differences with standard MariaDB.](#)



### Moving from MySQL to MariaDB in Debian 9

[MariaDB 10.1 is the default mysql server in Debian 9 "Stretch"](#)



### Creating a Debian Repository

[Instructions for creating your own Debian repository](#)



### MariaDB 5.5.33 Debian and Ubuntu Installation Issues

[Workarounds for some repository issues with the 5.5.33 release.](#)



### MariaDB Debian Live Images

[Debian live iso images with pre-installed MariaDB \(obsolete\)](#)



### apt-upgrade Fails, But the Database is Running

[timeout causing apt to fail while the database is running](#)

There are 9 related questions .

## 2.1.8.4.1 Differences in MariaDB in Debian (and Ubuntu)

### Contents

1. [Option File Locations](#)
2. [System Variables](#)
3. [Options](#)
4. [TLS](#)
5. [Authentication](#)
6. [See Also](#)
7. [More Information](#)

The

.deb

packages provided by MariaDB Foundation's and MariaDB Corporation's repositories are not identical to the official

.deb

packages provided by Debian's and Ubuntu's default repositories.

The packages provided by MariaDB Foundation's and MariaDB Corporation's repositories are generated using the Debian packaging in MariaDB's official [source code](#). The Debian packaging scripts are specifically in the

debian/  
directory.

The packages provided by Debian's and Ubuntu's default repositories are generated using the Debian packaging in Debian's mirror of MariaDB's source code, which contains some custom changes. The source tree can be found here:

- <https://anonscm.debian.org/cgit/pkg-mysql/mariadb-10.1.git/tree/debian>

As a consequence, MariaDB behaves a bit differently if it is installed from Debian's and Ubuntu's default repositories.

## Option File Locations

- The [option file](#) located at

/etc/mysql/my.cnf  
is handled by the

[update-alternatives](#)

mechanism when the

mysql-common

package is installed. It is a symbolic link that references either

mysql.cnf

or

mariadb.cnf

depending on whether MySQL or MariaDB is installed. Most of the MariaDB [option files](#) are therefore actually located in  
/etc/mysql/mariadb.d/

## System Variables

| Variable             | MariaDB in Debian  | Standard MariaDB  | Notes                                                            |
|----------------------|--------------------|-------------------|------------------------------------------------------------------|
| character_set_server | utf8mb4            | latin1            | Debian sets a default character set that can support emojis etc. |
| collation_server     | utf8mb4_general_ci | latin1_swedish_ci |                                                                  |

## Options

| Option | MariaDB in Debian | Standard<br>MariaDB | Notes |
|--------|-------------------|---------------------|-------|
|        |                   |                     |       |

|                                              |                                                                                                                                                                                                |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>add plugin-load- auth_socket.so -</pre> | <p>Before <a href="#">MariaDB 10.4.3</a>, MariaDB did not enable the <a href="#">unix_socket</a> authentication plugin by default. This is default in Debian, allowing passwordless login.</p> |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## TLS

- MariaDB binaries from

[.deb](#)

packages provided by Debian's and Ubuntu's default repositories are linked with a different TLS library than MariaDB binaries from

[.deb](#)

packages provided by MariaDB Foundation's and MariaDB Corporation's repositories.

- MariaDB Server binaries:

- In [MariaDB 10.4.6](#) and later, MariaDB Server is statically linked with the bundled [wolfSSL](#) libraries in

[.deb](#)

packages provided by Debian's and Ubuntu's default repositories.

- In [MariaDB 10.4.5](#) and before, MariaDB Server is statically linked with the bundled [yaSSL](#) libraries in

[.deb](#)

packages provided by Debian's and Ubuntu's default repositories.

- In contrast, MariaDB Server is dynamically linked with the system's [OpenSSL](#) libraries in

[.deb](#)

packages provided by MariaDB Foundation and MariaDB Corporation.

- MariaDB [client and utility](#) binaries:

- In [MariaDB 10.4.6](#) and later, MariaDB's [clients and utilities](#) and [MariaDB Connector/C](#) are dynamically linked with the system's [GnuTLS](#) libraries in

[.deb](#)

packages provided by Debian's and Ubuntu's default repositories. [libmysqlclient](#) is still statically linked with the bundled [wolfSSL](#) libraries.

- In [MariaDB 10.2](#) and later, MariaDB's [clients and utilities](#) and [MariaDB Connector/C](#) are dynamically linked with the system's [GnuTLS](#) libraries in

[.deb](#)

packages provided by Debian's and Ubuntu's default repositories. [libmysqlclient](#) is still statically linked with the bundled [yaSSL](#) libraries.

- In [MariaDB 10.1](#) and earlier, MariaDB's [clients and utilities](#) and [libmysqlclient](#) are statically linked with the bundled [yaSSL](#) libraries in

[.deb](#)

packages provided by Debian's and Ubuntu's default repositories.

- In contrast, MariaDB's [clients and utilities](#), [libmysqlclient](#), and [MariaDB Connector/C](#) are dynamically linked with the system's [OpenSSL](#) libraries in

[.deb](#)

packages provided by MariaDB Foundation's and MariaDB Corporation's repositories.

- See [TLS and Cryptography Libraries Used by MariaDB](#) for more information about which libraries are used on which platforms.

## Authentication

- The

[unix\\_socket](#)

authentication plugin is installed by default in **new installations** that use the

.deb

packages provided by Debian's default repositories in Debian 9 and later and Ubuntu's default repositories in Ubuntu 15.10 and later.

- The

root@localhost  
created by

`mysql_install_db`

will also be created to authenticate via the

`unix_socket`

authentication plugin in these builds.

## See Also

- [Moving from MySQL to MariaDB in Debian 9](#)

## More Information

For details, check out the Debian and Ubuntu official repositories:

- <https://packages.debian.org/search?keywords=mariadb-server&searchon=names&suite=all&section=all>
- <http://packages.ubuntu.com/search?keywords=mariadb-server&searchon=names&suite=all&section=all>

## 2.1.8.4.2 Moving from MySQL to MariaDB in Debian 9

MariaDB 10.1 is now the default mysql server in Debian 9 "Stretch". This page provides information on this change and instructions to help with upgrading your Debian 8 "Jessie" version of MySQL or MariaDB to [MariaDB 10.1](#) in Debian 9 "Stretch".

### Contents

1. [Background information](#)
2. [Before you upgrade](#)
  1. [Backup before you begin](#)
  2. [Changed, renamed, and removed options](#)
    1. [Options with changed default values](#)
    2. [Options that have been removed or renamed](#)
  3. [Suggested upgrade procedure for replication](#)
  4. [Other resources to consult before beginning your upgrade](#)
3. [Upgrading to MariaDB 10.1 from MySQL 5.5](#)
4. [Upgrading to MariaDB 10.1 from an older version of MariaDB](#)
5. [MariaDB Galera Cluster](#)
6. [Configuration options for advanced database users](#)
7. [Secure passwordless root accounts only on new installs](#)
8. [See also](#)
9. [Comments and suggestions](#)
10. [Notes](#)

## Background information

The version of MySQL in Debian 8 "Jessie" is 5.5. When installing, most users will install the

`mysql-server`  
package, which depends on the  
`mysql-server-5.5` package  
. In Debian 9 "Stretch" the  
`mysql-server`  
package depends on a new package called  
`default-mysql-server`  
. This package in turn depends on

```
mariadb-server-10.1
. There is no
default-mysql-server
package in Jessie.
```

In both Jessie and Stretch there is also a

```
mariadb-server
package which is a MariaDB-specific analog to the
mysql-server
package. In Jessie this package depends on
mariadb-server-10.0
and in Stretch this package depends on
mariadb-server-10.1
(the same as the
default-mysql-server
package).
```

So, the main repository difference in Debian 9 "Stretch" is that when you install the

```
mysql-server
package on Stretch you will get MariaDB 10.1 instead of MySQL, like you would with previous versions of Debian. Note that
mysql-server
is just an empty transitional meta-package and users are encouraged to install MariaDB using the actual package
mariadb-server
```

All apps and tools, such as the popular LAMP stack, in the repositories that depend on the

```
mysql-server
package will continue to work using MariaDB as the database. For new installs there is nothing different that needs to be done when installing
the mysql-server or mariadb-server packages.
```

## Before you upgrade

If you are currently running MySQL 5.5 on Debian 8 "Jessie" and are planning an upgrade to [MariaDB 10.1](#) on Debian 9 "Stretch", there are some things to keep in mind:

### Backup before you begin

This is a major upgrade, and so complete database backups are strongly suggested before you begin. [MariaDB 10.1](#) is compatible on disk and wire with MySQL 5.5, and the MariaDB developer team has done extensive development and testing to make upgrades as painless and trouble-free as possible. Even so, it's always a good idea to do regular backups, especially before an upgrade. As the database has to shutdown anyway for the upgrade, this is a good opportunity to do a backup!

### Changed, renamed, and removed options

Some default values have been changed, some have been renamed, and others have been removed between MySQL 5.5 and [MariaDB 10.1](#). The following sections detail them.

#### Options with changed default values

Most of the following options have increased a bit in value to give better performance. They should not use much additional memory, but some of them do use a bit more disk space.

| Option                                     | Old default value | New default value |
|--------------------------------------------|-------------------|-------------------|
| <a href="#">aria-sort-buffer-size</a>      | 128M              | 256M              |
| <a href="#">back_log</a>                   | 50                | 150               |
| <a href="#">innodb-concurrency-tickets</a> | 500               | 5000              |

|                                          |      |                                                         |
|------------------------------------------|------|---------------------------------------------------------|
| <code>innodb-log-file-size</code>        | 5M   | 48M                                                     |
| <code>innodb_log_compressed_pages</code> | ON   | OFF                                                     |
| <code>innodb-old-blocks-time</code>      | 0    | 1000                                                    |
| <code>innodb-open-files</code>           | 300  | 400<br>[2]                                              |
| <code>innodb-purge-batch-size</code>     | 20   | 300                                                     |
| <code>innodb-undo-logs</code>            | ON   | 20                                                      |
| <code>join_buffer_size</code>            | 128K | 256K                                                    |
| <code>max_allowed_packet</code>          | 1M   | 4M                                                      |
| <code>max-connect-errors</code>          | 10   | 100                                                     |
| <code>max-relay-log-size</code>          | 0    | 1024M                                                   |
| <code>myisam-sort-buffer-size</code>     | 8M   | 128M                                                    |
| <code>optimizer-switch</code>            | ...  | Added<br><code>extended_keys=on, exists_to_in=on</code> |

|                                      |       |                                                              |
|--------------------------------------|-------|--------------------------------------------------------------|
| <code>query_alloc_block_size</code>  | 8192  | 16384                                                        |
| <code>query_cache_size</code>        | 0     | 1M                                                           |
| <code>query_cache_type</code>        | ON    | OFF                                                          |
| <code>query_prealloc_size</code>     | 8192  | 24576                                                        |
| <code>secure_auth</code>             | OFF   | ON                                                           |
| <code>sql_log_bin</code>             |       | No longer affects replication of events in a Galera cluster. |
| <code>sql_mode</code>                | empty | <code>NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION</code>     |
| <code>sync_master_info</code>        | 0     | 10000                                                        |
| <code>sync_relay_log</code>          | 0     | 10000                                                        |
| <code>sync_relay_log_info</code>     | 0     | 10000                                                        |
| <code>table_open_cache</code>        | 400   | 2000                                                         |
| <code>thread_pool_max_threads</code> | 500   | 1000                                                         |

#### Options that have been removed or renamed

The following options should be removed or renamed if you use them in your config files:

| Option                                | Reason                                                                            |
|---------------------------------------|-----------------------------------------------------------------------------------|
| engine-condition-pushdown             | Replaced with<br><code>set optimizer_switch='engine_condition_pushdown=on'</code> |
| innodb-adaptive-flushing-method       | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-autoextend-increment           | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-blocking-buffer-pool-restore   | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-buffer-pool-pages              | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-buffer-pool-pages-blob         | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-buffer-pool-pages-index        | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-buffer-pool-restore-at-startup | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-buffer-pool-shm-checksum       | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-buffer-pool-shm-key            | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-checkpoint-age-target          | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-dict-size-limit                | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-doublewrite-file               | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-fast-checksum                  | Renamed to <a href="#">innodb-checksum-algorithm</a>                              |
| innodb-flush-neighbor-pages           | Renamed to <a href="#">innodb-flush-neighbors</a>                                 |
| innodb-ibuf-accel-rate                | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-ibuf-active-contract           | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-ibuf-max-size                  | Removed by <a href="#">XtraDB</a>                                                 |
| innodb-import-table-from-xtrabackup   | Removed by <a href="#">XtraDB</a>                                                 |

|                                       |                                        |
|---------------------------------------|----------------------------------------|
| innodb-index-stats                    | Removed by <a href="#">XtraDB</a>      |
| innodb-lazy-drop-table                | Removed by <a href="#">XtraDB</a>      |
| innodb-merge-sort-block-size          | Removed by <a href="#">XtraDB</a>      |
| innodb-persistent-stats-root-page     | Removed by <a href="#">XtraDB</a>      |
| innodb-read-ahead                     | Removed by <a href="#">XtraDB</a>      |
| innodb-recovery-stats                 | Removed by <a href="#">XtraDB</a>      |
| innodb-recovery-update-relay-log      | Removed by <a href="#">XtraDB</a>      |
| innodb-stats-auto-update              | Renamed to<br>innodb-stats-auto-recalc |
| innodb-stats-update-need-lock         | Removed by <a href="#">XtraDB</a>      |
| innodb-sys-stats                      | Removed by <a href="#">XtraDB</a>      |
| innodb-table-stats                    | Removed by <a href="#">XtraDB</a>      |
| innodb-thread-concurrency-timer-based | Removed by <a href="#">XtraDB</a>      |
| innodb-use-sys-stats-table            | Removed by <a href="#">XtraDB</a>      |
| rpl_recovery_rank                     | Unused in 10.0+                        |
| xtradb-admin-command                  | Removed by <a href="#">XtraDB</a>      |

## Suggested upgrade procedure for replication

If you have a [master-slave setup](#), the normal procedure is to first upgrade your slaves to MariaDB, then move one of your slaves to be the master and then upgrade your original master. In this scenario you can upgrade from MySQL to MariaDB or upgrade later to a new version of MariaDB without any downtime.

## Other resources to consult before beginning your upgrade

It may also be useful to check out the [Upgrading MariaDB](#) section. It contains several articles on upgrading from MySQL to MariaDB and from one version of MariaDB to another. For upgrade purposes, MySQL 5.5 and [MariaDB 5.5](#) are very similar. In particular, see the [Upgrading from MariaDB 5.5 to MariaDB 10.0](#) and [Upgrading from MariaDB 10.0 to MariaDB 10.1](#) articles.

If you need help with upgrading or setting up replication, you can always [contact the MariaDB corporation](#) to find experts to help you with this.

# Upgrading to MariaDB 10.1 from MySQL 5.5

The suggested upgrade procedure is:

1. Set `innodb_fast_shutdown` to  
0
  - . This is to ensure that if you make a backup as part of the upgrade, all data is written to the InnoDB data files, which simplifies any restore in the future.
2. Shutdown MySQL 5.5
3. Take a `backup`
  - when the server is shut down is the perfect time to take a backup of your databases
  - store a copy of the backup on external media or a different machine for safety
4. Perform the upgrade from Debian 8 to Debian 9
5. During the upgrade, the `mysql_upgrade` script will be run automatically; this script does two things:
  1. Upgrades the permission tables in the  
`mysql`  
database with some new fields
  2. Does a very quick check of all tables and marks them as compatible with **MariaDB 10.1**
    - In most cases this should be a fast operation (depending of course on the number of tables)
6. Add new options to `my.cnf` to enable features
  - If you change  
`my.cnf`  
then you need to restart  
`mysqld`  
with e.g.  
`sudo service mysql restart`  
or  
`sudo service mariadb restart`

## Upgrading to MariaDB 10.1 from an older version of MariaDB

If you have installed **MariaDB 5.5** or **MariaDB 10.0** on your Debian 8 "Jessie" machine from the MariaDB repositories you will need to upgrade to **MariaDB 10.1** when upgrading to Debian 9 "Stretch". You can choose to continue using the MariaDB repositories or move to using the Debian repositories.

If you want to continue using the MariaDB repositories edit the MariaDB entry in your `sources.list` and change every instance of 5.5 or 10.0 to 10.1. Then upgrade as suggested [above](#).

If you want to move to using **MariaDB 10.1** from the Debian repositories, delete or comment out the MariaDB entries in your `sources.list` file. Then upgrade as suggested [above](#).

If you are already using **MariaDB 10.1** on your Debian 8 "Jessie" machine, you can choose to continue to use the MariaDB repositories or move to using the Debian repositories as with **MariaDB 5.5** and 10.0. In either case, the upgrade will at most be just a minor upgrade from one version of **MariaDB 10.1** to a newer version. In the case that you are already on the current version of MariaDB that exists in the Debian repositories or a newer one) MariaDB will not be upgraded during the system upgrade but will be upgraded when future versions of MariaDB are released.

You should always perform a compete backup of your data prior to performing any major system upgrade, even if MariaDB itself is not being upgraded!

## MariaDB Galera Cluster

If you have been using MariaDB Galera Cluster 5.5 or 10.0 on Debian 8 "Jessie" it is worth mentioning that **Galera Cluster** is included by default in **MariaDB 10.1**, there is no longer a need to install a separate

```
mariadb-galera-server  
package.
```

## Configuration options for advanced database users

To get better performance from MariaDB used in production environments, here are some suggested additions to `your configuration file` which in Debian is at

```
/etc/mysql/mariadb.d/my.cnf  
:  
:
```

```
[[mysqld]]  
# Cache for disk based temporary files  
aria_pagecache_buffer_size=128M  
# If you are not using MyISAM tables directly (most people are using InnoDB)  
key_buffer_size=64K
```

The reason for the above change is that MariaDB is using the newer **Aria** storage engine for disk based temporary files instead of MyISAM. The main

benefit of Aria is that it can cache both indexes and rows and thus gives better performance than MyISAM for large queries.

## Secure passwordless root accounts only on new installs

Unlike the old MySQL packages in Debian, [MariaDB 10.0](#) onwards in Debian uses unix socket authentication on new installs to avoid root password management issues and thus be more secure and easier to use with provision systems of the cloud age.

This only affects new installs. Upgrades from old versions will continue to use whatever authentication and user accounts already existed. This is however good to know, because it can affect upgrades of dependant systems, typically e.g. require users to rewrite their Ansible scripts and similar tasks. The new feature is much easier than the old, so adjusting for it requires little work.

## See also

- [Differences in MariaDB in Debian \(and Ubuntu\)](#)
- [Configuring MariaDB for optimal performance](#)
- [New features in MariaDB you should considering using](#)
- [What is MariaDB 10.1](#)
- [General instructions for upgrading from MySQL to MariaDB](#)

## Comments and suggestions

If you have comments or suggestions on things we can add or change to improve this page. Please add them as comments below.

## Notes

1. ↑ The

```
innodb-open-files  
variable defaults to the value of  
table-open-cache  
(  
400  
is the default) if it is set to any value less than  
10  
so long as  
innodb-file-per-table  
is set to  
1  
or  
TRUE  
(the default). If  
innodb_file_per_table  
is set to  
0  
or  
FALSE  
and  
innodb-open-files  
is set to a value less than  
10  
, the default is  
300
```

### 2.1.8.4.3 Creating a Debian Repository

Below are instructions for creating your own Debian repository. The instructions are based on <http://www.debian.org/doc/manuals/repository-howto/repository-howto.en.html>

```
REPO_DIR={pick some location}  
mkdir $REPO_DIR  
mkdir $REPO_DIR/binary  
mkdir $REPO_DIR/source  
cp *.deb *.ddeb $REPO_DIR/binary  
cd $REPO_DIR  
dpkg-scanpackages binary /dev/null | gzip -9c > binary/Packages.gz  
dpkg-scansources source /dev/null | gzip -9c > source/Sources.gz
```

## Using the Debian repository you just created

One needs to add a new file to the  
/etc/apt/sources.list.d/  
directory. For instance a new file called  
mariadb.list

```
# sergey's MariaDB repository
#
deb file:///home/psergey/testrepo binary/
deb-src file:///home/psergey/testrepo source/
```

after which one can run

```
apt-get update # Let apt learn about the new repository
apt-get install mariadb-server
```

and collect bugs :-).

"apt-get install" will spray output of scripts and servers all over /var/log. It is also possible to set DEBIAN\_SCRIPT\_DEBUG=1 to get some (not all) of it to stdout.

## Cleaning up after failed installation

Run

```
dpkg --get-selections | grep mariadb
dpkg --get-selections | grep mysql
```

to see what is installed, and then

```
dpkg --purge <packages>
```

until the former produces empty output. Note: after some failures, /etc/mysql and /var/lib/mysql are not cleaned and still need to be removed manually.

## 2.1.8.4.4 apt-upgrade Fails, But the Database is Running

After running

```
apt-upgrade mariadb
, it's possible that apt shows a fail in trying to start the server, but in fact the database is up and running, which then provokes apt to remain in a non finished state.
```

For example:

```
# apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
2 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n]
Setting up mariadb-server-10.1 (10.1.10+maria-1~trusty) ...
 * Stopping MariaDB database server mysqld
   ...done.
 * Starting MariaDB database server mysqld
   ...fail!
invoke-rc.d: initscript mysql, action "start" failed.
dpkg: error processing package mariadb-server-10.1 (--configure):
 subprocess installed post-installation script returned error exit status 1
dpkg: dependency problems prevent configuration of mariadb-server:
 mariadb-server depends on mariadb-server-10.1 (= 10.1.10+maria-1~trusty); however:
  Package mariadb-server-10.1 is not configured yet.

dpkg: error processing package mariadb-server (--configure):
 dependency problems - leaving unconfigured
No apport report written because the error message indicates its a followup error from a previous failure.
Errors were encountered while processing:
 mariadb-server-10.1
 mariadb-server
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

This situation could occur if the timeout for the init script was too short. For example, see [MDEV-9382](#), a situation where the timeout was 30 seconds, but the server was taking 48 seconds to start.

To overcome this, the timeout needs to be increased. This can be achieved as follows:

- **On systems where systemd is not enabled/supported:** The timeout can be increased by setting MYSQLD\_STARTUP\_TIMEOUT either directly in the script or via the command line. In [MariaDB 10.1.13](#) and later versions, the init script also sources /etc/default/mariadb, so it can also be used to set MYSQLD\_STARTUP\_TIMEOUT to persistently change the startup timeout. The default timeout has been increased from 30s to 60s in [MariaDB 10.1.13](#).
- **On systems that support systemd:** The startup timeout can be increased by setting `TimeoutStartSec` `systemd` option.

## 2.1.8.5 What to Do if MariaDB Doesn't Start

### Contents

1. [The Error Log and the Data Directory](#)
2. [Option Files](#)
  1. [Invalid Option or Option Value](#)
  3. [Can't Open Privilege Tables](#)
  4. [Can't Create Test File](#)
  5. [InnoDB](#)
    1. [Cannot Allocate Memory for the InnoDB Buffer Pool](#)
    2. [InnoDB Table Corruption](#)
  6. [MyISAM](#)
  7. [systemd](#)
  8. [SELinux](#)

There could be many reasons that MariaDB fails to start. This page will help troubleshoot some of the more common reasons and provide solutions.

If you have tried everything here, and still need help, you can ask for help on IRC or on the forums - see [Where to find other MariaDB users and developers](#) - or ask a question at the [Starting and Stopping MariaDB](#) page.

## The Error Log and the Data Directory

The reason for the failure will almost certainly be written in the [error log](#) and, if you are starting MariaDB manually, to the console. By default, the error log is named *host-name*.err and is written to the data directory.

Common Locations:

- /var/log/
- /var/log/mysql
- C:\Program Files\MariaDB x.y\data (x.y refers to the version number)
- C:\Program Files (x86)\MariaDB x.y\data (32bit version on 64bit Windows)

It's also possible that the error log has been explicitly written to another location. This is often done by changing the

`datadir`

or

`log_error`

system variables in an [option file](#). See [Option Files](#) below for more information about that.

A quick way to get the values of these system variables is to execute the following commands:

```
mysqld --help --verbose | grep 'log-error' | tail -1  
mysqld --help --verbose | grep 'datadir' | tail -1
```

## Option Files

Another kind of file to consider when troubleshooting is [option files](#). The default option file is called

`my.cnf`

Option files contain configuration options, such as the location of the data directory mentioned above. If you're unsure where the option file is located, see [Configuring MariaDB with Option Files: Default Option File Locations](#) for information on the default locations.

You can check which configuration options MariaDB server will use from its option files by executing the following command:

```
mysqld --print-defaults
```

You can also check by executing the following command:

```
my_print_defaults --mysqld
```

See [Configuring MariaDB with Option Files: Checking Program Options](#) for more information on checking configuration options.

## Invalid Option or Option Value

Another potential reason for a startup failure is that an [option file](#) contains an invalid option or an invalid option value. In those cases, the [error log](#) should contain an error similar to this:

```
140514 12:19:37 [ERROR] /usr/local/mysql/bin/mysqld: unknown variable 'option=value'
```

This is more likely to happen when you upgrade to a new version of MariaDB. In most cases the [option file](#) from the old version of MariaDB will work just fine with the new version. However, occasionally, options are removed in new versions of MariaDB, or the valid values for options are changed in new versions of MariaDB. Therefore, it's possible for an [option file](#) to stop working after an upgrade.

Also remember that option names are case sensitive.

Examine the specifics of the error. Possible fixes are usually one of the following:

- If the option is completely invalid, then remove it from the [option file](#).
- If the option's name has changed, then fix the name.
- If the option's valid values have changed, then change the option's value to a valid one.
- If the problem is caused by a simple typo, then fix the typo.

## Can't Open Privilege Tables

It is possible to see errors similar to the following:

```
System error 1067 has occurred.  
Fatal error: Can't open privilege tables: Table 'mysql.host' doesn't exist
```

If errors like this occur, then critical [system tables](#) are either missing or are in the wrong location. The above error is quite common after an upgrade if the [option files](#) set the

[basedir](#)

or

[datadir](#)

to a non-standard location, but the new server is using the default location. Therefore, make sure that the

[basedir](#)

and

[datadir](#)

variables are correctly set.

If you're unsure where the option file is located, see [Configuring MariaDB with Option Files: Default Option File Locations](#) for information on the default locations.

If the [system tables](#) really do not exist, then you may need to create them with

[mysql\\_install\\_db](#)

. See [Installing System Tables \(mysql\\_install\\_db\)](#) for more information.

## Can't Create Test File

One of the first tests on startup is to check whether MariaDB can write to the data directory. When this fails, it will log an error like this:

```
May 13 10:24:28 mariadb3 mysqld[19221]: 2019-05-13 10:24:28 0 [Warning] Can't create test file /usr/local/data/mariadb/mariadb3.  
May 13 10:24:28 mariadb3 mysqld[19221]: 2019-05-13 10:24:28 0 [ERROR] Aborting
```

This is usually a permission error on the directory in which this file is being written. Ensure that the entire

[datadir](#)

```

is owned by the user running
mysqld
, usually
mysql
. Ensure that directories have the "x" (execute) directory permissions for the owner. Ensure that all the parent directories of the

datadir

upwards have "x" (execute) permissions for all (
user
,
group
, and
other
).

```

Once this is checked look at the [systemd](#) and [selinux](#) documentation below, or [apparmor](#).

## InnoDB

[InnoDB](#) is probably the MariaDB component that most frequently causes a crash. In the error log, lines containing InnoDB messages generally start with "InnoDB:".

### Cannot Allocate Memory for the InnoDB Buffer Pool

In a typical installation on a dedicated server, at least 70% of your memory should be assigned to [InnoDB buffer pool](#); sometimes it can even reach 85%. But be very careful: don't assign to the buffer pool more memory than it can allocate. If it cannot allocate memory, InnoDB will use the disk's swap area, which is very bad for performance. If swapping is disabled or the swap area is not big enough, InnoDB will crash. In this case, MariaDB will probably try to restart several times, and each time it will log a message like this:

```
140124 17:29:01 InnoDB: Fatal error: cannot allocate memory for the buffer pool
```

In that case, you will need to add more memory to your server/VM or decrease the value of the [innodb\\_buffer\\_pool\\_size](#) variables.

Remember that the buffer pool will slightly exceed that limit. Also, remember that MariaDB also needs allocate memory for other storage engines and several per-connection buffers. The operating system also needs memory.

### InnoDB Table Corruption

By default, InnoDB deliberately crashes the server when it detects table corruption. The reason for this behavior is preventing corruption propagation. However, in some situations, server availability is more important than data integrity. For this reason, we can avoid these crashes by changing the value of [innodb\\_corrupt\\_table\\_action](#) to 'warn'.

If InnoDB crashes the server after detecting data corruption, it writes a detailed message in the error log. The first lines are similar to the following:

```
InnoDB: Database page corruption on disk or a failed
InnoDB: file read of page 7.
InnoDB: You may have to recover from a backup.
```

Generally, it is still possible to recover most of the corrupted data. To do so, restart the server in [InnoDB recovery mode](#) and try to extract the data that you want to backup. You can save them in a CSV file or in a non-InnoDB table. Then, restart the server in normal mode and restore the data.

## MyISAM

Most tables in the [mysql](#) database are MyISAM tables. These tables are necessary for MariaDB to properly work, or even start.

A MariaDB crash could cause system tables corruption. With the default settings, MariaDB will simply not start if the system tables are corrupted. With [myisam\\_recover\\_options](#), we can force MyISAM to repair damaged tables.

## systemd

If you are using

```
systemd
```

, then there are a few relevant notes about startup failures:

- If MariaDB is configured to access files under  
/home

```
,  
/root  
, or  
/run/user  
, then the default systemd unit file will prevent access to these directories with a  
Permission Denied  
error. This happens because the unit file set
```

```
ProtectHome=true
```

. See [Systemd: Configuring Access to Home Directories](#) for information on how to work around this.

- The default systemd unit file also sets `ProtectSystem=full` , which places restrictions on writing to a few other directories. Overwriting this with `ProtectSystem=off` in the same way as above will restore access to these directories.
- If MariaDB takes longer than 90 seconds to start, then the default systemd unit file will cause it to fail with an error. This happens because the default value for the

```
TimeoutStartSec
```

option is 90 seconds. See [Systemd: Configuring the Systemd Service Timeout](#) for information on how to work around this.

- The systemd journal may also contain useful information about startup failures. See [Systemd: Systemd Journal](#) for more information.

See [systemd](#) documentation for further information on systemd configuration.

## SELinux

**Security-Enhanced Linux (SELinux)** is a Linux kernel module that provides a framework for configuring **mandatory access control (MAC)** system for many resources on the system. It is enabled by default on some Linux distributions, including RHEL, CentOS, Fedora, and other similar Linux distribution. SELinux prevents programs from accessing files, directories or ports unless it is configured to access those resources.

You might need to troubleshoot SELinux-related issues in cases, such as:

- MariaDB is using a non-default port.
- MariaDB is reading from or writing to some files (datadir, log files, option files, etc.) located at non-default paths.
- MariaDB is using a plugin that requires access to resources that default installations do not use.

Setting SELinux state to

```
permissive
```

is a common way to investigate what is going wrong while allowing MariaDB to function normally.

```
permissive
```

is supposed to produce a log entry every time it should block a resource access, without actually blocking it. However, [there are situations](#) when SELinux blocks resource accesses even in

```
permissive
```

mode.

See [SELinux](#) for more information.

## 2.1.8.6 Installing on an Old Linux Version

This article lists some typical errors that may happen when you try to use an incompatible MariaDB binary on a linux system:

The following example errors are from trying to install MariaDB built for SuSE 11.x on a SuSE 9.0 server:

```
> scripts/mysql_install_db  
.bin/my_print_defaults: /lib/i686/libc.so.6:  
version `GLIBC_2.4' not found (required by ./bin/my_print_defaults)
```

and

```
> ./bin/mysqld --skip-grant &  
.bin/mysqld: error while loading shared libraries: libwrap.so.0:  
cannot open shared object file: No such file or directory
```

If you see either of the above errors, the binary MariaDB package you installed is not compatible with your system.

The options you have are:

- Find another MariaDB package or tar from the [download page](#) that matches your system.  
or  
• [Download the source](#) and [build it](#) .

## 2.1.8.7 Error: symbol mysql\_get\_server\_name, version libmysqlclient\_16 not defined

If you see the error message:

```
symbol mysql_get_server_name, version libmysqlclient_16 not defined in file libmysqlclient.so.16 with link time reference
```

...then you are probably trying to use the mysql command-line client from MariaDB with libmysqlclient.so from MySQL.

The symbol

```
mysql_get_server_name()  
is something present in the MariaDB source tree and not in the MySQL tree.
```

If you have both the MariaDB client package and the MySQL client packages installed this error will happen if your system finds the MySQL version of libmysqlclient.so first.

To figure out which library is being linked in dynamically (ie, the wrong one) use the 'ldd' tool.

```
ldd $(which mysql) | grep mysql
```

or

```
ldd /path/to/the/binary | grep mysql
```

For example:

```
me@mybox:~$ ldd $(which mysql)|grep mysql  
libmysqlclient.so.16 => /usr/lib/libmysqlclient.so.16 (0xb74df000)
```

You can then use your package manager's tools to find out which package the library belongs to.

On CentOS the command to find out which package installed a specific file is:

```
rpm -qf /path/to/file
```

On Debian-based systems, the command is:

```
dpkg -S /path/to/file
```

Here's an example of locating the library and finding out which package it belongs to on an Ubuntu system:

```
me@mybox:~$ ldd $(which mysql)|grep mysql  
libmysqlclient.so.16 => /usr/lib/libmysqlclient.so.16 (0xb75f8000)  
me@mybox:~$ dpkg -S /usr/lib/libmysqlclient.so.16  
libmariadbclient16: /usr/lib/libmysqlclient.so.16
```

The above shows that the mysql command-line client is using the library

```
/usr/lib/libmysqlclient.so.16  
and that library is part of the  
libmariadbclient16  
Ubuntu package. Unsurprisingly, the mysql command-line client works perfectly on this system.
```

If the answer that came back had been something other than a MariaDB package, then it is likely there would have been issues with running the MariaDB mysql client application.

If the library that the system tries to use is not from a MariaDB package, the remedy is to remove the offending package (and possibly install or re-install the correct package) so that the correct library can be used.

## 2.1.9 Installing System Tables (mysql\_install\_db)

### 2.1.10 mysql\_install\_db.exe

#### Contents

1. [Functionality](#)
2. [Example](#)
3. [Removing Database Instances](#)

The

`mysql_install_db.exe`  
utility is the Windows equivalent of [mysql\\_install\\_db](#).

## Functionality

The functionality of

`mysql_install_db.exe`

is comparable with the shell script

`mysql_install_db`

used on Unix, however it has been extended with both Windows specific functionality (creating a Windows service) and to generally useful functionality. For example, it can set the 'root' user password during database creation. It also creates the

`my.ini`

configuration file in the data directory and adds most important parameters to it (e.g port).

`mysql_install_db.exe`

is used by the MariaDB installer for Windows if the "Database instance" feature is selected. It obsoletes similar utilities and scripts that were used in the past such as

`mysqld.exe`

--

install

,

`mysql_install_db.pl`

, and

`mysql_secure_installation.pl`

| Parameter                                                             | Description                        |
|-----------------------------------------------------------------------|------------------------------------|
| <code>-?</code><br>,<br><code>--</code><br><code>help</code>          | Display help message and exit      |
| <code>-d</code><br>,<br><code>--</code><br><code>datadir=name</code>  | Data directory of the new database |
| <code>-S</code><br>,<br><code>--</code><br><code>service=name</code>  | Name of the Windows service        |
| <code>-p</code><br>,<br><code>--</code><br><code>password=name</code> | Password of the root user          |

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| -P<br>,                    | mysqld port                                            |
| -W<br>,                    | named pipe name                                        |
| --socket=name              |                                                        |
| -D<br>,                    | Create default user                                    |
| --default-user             |                                                        |
| -R<br>,                    | Allow remote access from network for user root         |
| --allow-remote-root-access |                                                        |
| -N<br>,                    | Do not use TCP connections, use pipe instead           |
| --skip-networking          |                                                        |
| -i<br>,                    | Innodb page size, since <a href="#">MariaDB 10.2.5</a> |
| --innodb-page-size         |                                                        |
| -s<br>,                    | Print less information                                 |
| --silent                   |                                                        |
| -o<br>,                    | Include mysqld bootstrap output                        |
| --verbose-bootstrap        |                                                        |

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| -l<br>,               | Use large pages, since <a href="#">MariaDB 10.6.1</a> |
| --<br><br>large-pages |                                                       |

|                  |                                                                   |
|------------------|-------------------------------------------------------------------|
| -c<br>,          | my.ini config template file, since <a href="#">MariaDB 10.6.1</a> |
| --<br><br>config |                                                                   |

**Note :** to create a Windows service,

```
mysql_install_db.exe
```

should be run by a user with full administrator privileges (which means elevated command prompt on systems with UAC). For example, if you are running it on Windows 7, make sure that your command prompt was launched via 'Run as Administrator' option.

## Example

```
mysql_install_db.exe --datadir=C:\db --service=MyDB --password=secret
```

will create the database in the directory C:\db, register the auto-start Windows service "MyDB", and set the root password to 'secret'.

To start the service from the command line, execute

```
sc start MyDB
```

## Removing Database Instances

If you run your database instance as service, to remove it completely from the command line, use

```
sc stop <servicename>
sc delete <servicename>
rmdir /s /q <path-to-datadir>
```

## 2.1.11 Configuring MariaDB with Option Files

### 2.1.12 MariaDB Environment Variables

MariaDB makes use of numerous environment variables that may be set on your system. Environment variables have the lowest precedence, so any options set on the command line or in an option file will take precedence.

It's usually better not to rely on environment variables, and to rather set the options you need directly, as this makes the system a little more robust and easy to administer.

Here is a list of environment variables used by MariaDB.

| Environment Variable | Description                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------|
| CXX                  | Name of the C++ compiler, used for running CMake.                                              |
| CC                   | Name of the C compiler, used for running CMake.                                                |
| DBI_USER             | Perl DBI default username.                                                                     |
| DBI_TRACE            | Perl DBI trace options.                                                                        |
| HOME                 | Default directory for the <a href="#">mysql_history file</a> .                                 |
| MYSQL_DEBUG          | Debug trace options used when debugging.                                                       |
| MYSQL_GROUP_SUFFIX   | In addition to the given option groups, also read groups with this suffix.                     |
| MYSQL_HISTFILE       | Path to the <a href="#">mysql_history file</a> , overriding the \$HOME/.mysql_history setting. |
| MYSQL_HOME           | Path to the directory containing the <a href="#">my.cnf file</a> used by the server.           |

|                 |                                                                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MYSQL_HOST      | Default host name used by the <a href="#">mysql command line client</a> .                                                                                               |
| MYSQL_PS1       | Command prompt for use by the <a href="#">mysql command line client</a> .                                                                                               |
| MYSQL_PWD       | Default password when connecting to mysqld. It is strongly recommended to use a more secure method of sending the password to the server.                               |
| MYSQL_TCP_PORT  | Default TCP/IP port number.                                                                                                                                             |
| MYSQL_UNIX_PORT | On Unix, default socket file used for localhost connections.                                                                                                            |
| PATH            | Path to directories that hold executable programs (such as the <a href="#">mysql client</a> , <a href="#">mysqladmin</a> ), so that these can be run from any location. |
| TMPDIR          | Directory where temporary files are created.                                                                                                                            |
| TZ              | Local <a href="#">time zone</a> .                                                                                                                                       |
| UMASK           | Creation mode when creating files. See <a href="#">Specifying Permissions for Schema (Data) Directories and Tables</a> .                                                |
| UMASK_DIR       | Creation mode when creating directories. See <a href="#">Specifying Permissions for Schema (Data) Directories and Tables</a> .                                          |
| USER            | On Windows, up to <a href="#">MariaDB 5.5</a> , the default user name when connecting to the mysqld server. API GetUserName() is used in later versions.                |

## 2.1.13 MariaDB on Amazon AWS

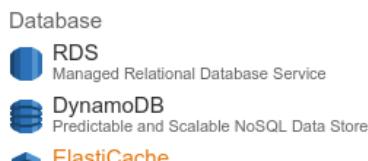
MariaDB is available on Amazon AWS through MariaDB SkySQL, as one of the database options when using Amazon's RDS service, or using a MariaDB AMI on Amazon EC2 from the AWS Marketplace.

### MariaDB SkySQL

Cloud database service is available through MariaDB SkySQL on Amazon AWS. MariaDB SkySQL delivers MariaDB with enterprise features for mission-critical workloads. Support is provided directly by MariaDB. Refer to [SkySQL Documentation](#) for complete details. [Get started](#) to launch a MariaDB database on AWS in minutes.

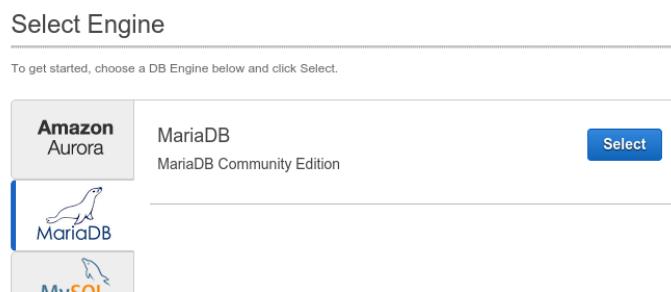
### Amazon RDS

To get started with MariaDB on Amazon's RDS service, click on the RDS link in the Database section of the [AWS console](#) .



Next, click on the **Get Started Now** button. Alternatively, you can click on the **Launch DB Instance** button from the [Instances](#) section of the [RDS Dashboard](#) .

In either case, you will be brought to the page where you can select the database engine you want to use. Click on the **MariaDB** logo and then click on the **Select** button.



You will then move to step 2 where you choose whether or not you want to use your MariaDB instance for production or non-production usage. Amazon has links on this page to documentation on the various options.

After selecting the choice you want you will move to step 3 where you specify the details for your database, including setting up an admin user in the database.

## Specify DB Details

### Instance Specifications

|                     |                                 |
|---------------------|---------------------------------|
| DB Engine           | mariadb                         |
| License Model       | general-public-license          |
| DB Engine Version   | 10.0.17                         |
| DB Instance Class   | db.t2.micro — 1 vCPU, 1 GiB RAM |
| Multi-AZ Deployment | No                              |
| Storage Type        | General Purpose (SSD)           |
| Allocated Storage*  | 5 GB                            |

**!** Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance.  
[Click here](#) for more details.

### Settings

|                         |         |
|-------------------------|---------|
| DB Instance Identifier* | test-db |
| Master Username*        | admin   |
| Master Password*        | .....   |
| Confirm Password*       | .....   |

Retype the value you specified for Master Password.

\* Required

Cancel

Previous

Next Step

You will then move to step 4 where you can configure advanced settings, including security settings, various options, backup settings, maintenance defaults, and so on.

Refer to [Amazon's RDS documentation](#) for complete documentation on all the various settings and for information on connecting to and using your RDS MariaDB instances.

## AMI on EC2

MariaDB AMIs (Amazon Machine Images) are available in the AWS Marketplace. These AMIs, kept up-to-date with the most recently released versions of MariaDB, are a great way to try out the newest MariaDB versions before they make it to RDS and/or to use MariaDB in a more traditional server environment.

### 2.1.14 Migrating to MariaDB

#### 2.1.14.1 Migrating to MariaDB from MySQL

##### 2.1.14.1.1 MySQL vs MariaDB: Performance

## Title: MariaDB versus MySQL - Features

See also [MariaDB vs MySQL - Compatibility](#)

## Differences Per Releases

- For differences between MySQL 8.0 and [MariaDB 10.7](#) specifically, see [Incompatibilities and Feature Differences Between MariaDB 10.7 and MySQL 8.0](#)
- For differences between MySQL 8.0 and [MariaDB 10.6](#) specifically, see [Incompatibilities and Feature Differences Between MariaDB 10.6 and MySQL 8.0](#)
- For differences between MySQL 8.0 and [MariaDB 10.5](#) specifically, see [Incompatibilities and Feature Differences Between MariaDB 10.5 and MySQL 8.0](#)
- For differences between MySQL 8.0 and [MariaDB 10.4](#) specifically, see [Incompatibilities and Feature Differences Between MariaDB 10.4 and MySQL 8.0](#)
- For differences between MySQL 5.7 and [MariaDB 10.3](#) specifically, see [Incompatibilities and Feature Differences Between MariaDB 10.3 and MySQL 5.7](#)
- For differences between MySQL 5.7 and [MariaDB 10.2](#) specifically, see [Incompatibilities and Feature Differences Between MariaDB 10.2 and MySQL 5.7](#)
- For a detailed breakdown of system variable differences, see:
  - [System variable differences between MariaDB 10.7 and MySQL 8.0](#)
  - [System variable differences between MariaDB 10.6 and MySQL 8.0](#)
  - [System variable differences between MariaDB 10.5 and MySQL 8.0](#)
  - [System variable differences between MariaDB 10.4 and MySQL 8.0](#)

- System variable differences between MariaDB 10.3 and MySQL 8.0
- System variable differences between MariaDB 10.3 and MySQL 5.7
- System variable differences between MariaDB 10.2 and MySQL 5.7
- System variable differences between MariaDB 10.1 and MySQL 5.7
- System variable differences between MariaDB 10.1 and MySQL 5.6
- System variable differences between MariaDB 10.0 and MySQL 5.6
- System variable differences between MariaDB 5.5 and MySQL 5.5
- For a detailed breakdown of function differences, see:
  - Function Differences Between MariaDB 10.7 and MySQL 8.0
  - Function Differences Between MariaDB 10.6 and MySQL 8.0
  - Function Differences Between MariaDB 10.5 and MySQL 8.0
  - Function Differences Between MariaDB 10.4 and MySQL 8.0
  - Function Differences Between MariaDB 10.3 and MySQL 8.0
  - Function Differences Between MariaDB 10.3 and MySQL 5.7
  - Function Differences Between MariaDB 10.2 and MySQL 5.7

## More Storage Engines

In addition to the standard `MyISAM`, `BLACKHOLE`, `CSV`, `MEMORY`, `ARCHIVE`, and `MERGE` storage engines, the following are also included with MariaDB Source and Binary packages:

- `ColumnStore`, a column oriented storage engine optimized for Data warehousing.
- `MyRocks`, a storage engine with great compression, in 10.2
- `Aria`, MyISAM replacement with better caching.
- `FederatedX` (drop-in replacement for Federated)
- `OQGRAPH` (In MariaDB 5.2 and later. Disabled in MariaDB 5.5 only.)
- `SphinxSE` (In MariaDB 5.2 and later)
- `CONNECT` in MariaDB 10.0 and later.
- `SEQUENCE` in MariaDB 10.0 and later.
- `Spider` in MariaDB 10.0 and later.
- `TokuDB` (In MariaDB 5.5 and later, removed in 10.6)
- `Cassandra` (In MariaDB 10.0, removed in 10.6)

## Speed Improvements

- MariaDB now provides much faster privilege checks for setups with many user accounts or many database
- The new `FLUSH SSL` command allows SSL certificates to be reloaded without restarting the server
- Many optimizer enhancements in MariaDB 5.3. Subqueries are now finally usable. The complete list and a comparison with MySQL is [here](#). A benchmark can be found [here](#).
- Faster and safer replication: `Group commit for the binary log`. This makes many setups that use replication and lots of updates [more than 2x times faster](#).
- `Parallel replication` — new in 10.0
- `Improvements` for InnoDB asynchronous IO subsystem on Windows.
- Indexes for the `MEMORY(HEAP)` engine are faster. According to a simple test, 24% faster on INSERT for integer index and 60% faster for index on a CHAR(20) column. Fixed in MariaDB 5.5 and MySQL 5.7.
- `Segmented Key Cache` for MyISAM. Can speed up MyISAM tables with up to 4x — new in 5.2
- `Adjustable hash size` for MyISAM and Aria. This can greatly improve shutdown time (from hours to minutes) if using a lot of MyISAM/Aria tables with delayed keys — new in 10.0.13
- `CHECKSUM TABLE` is faster.
- We improved the performance of character set conversions (and removed conversions when they were not really needed). Overall speed improvement is 1-5 % (according to sql-bench) but can be higher for big result sets with all characters between 0x00-0x7f.
- `Pool of Threads` in MariaDB 5.1 and even better in MariaDB 5.5. This allows MariaDB to run with 200,000+ connections and with a notable speed improvement when using many connections.
- Several speed improvements when a client connects to MariaDB. Many of the improvements were done in MariaDB 10.1 and MariaDB 10.2.
- There are some improvements to the DBUG code to make its execution faster when debug is compiled in but not used.
- Our use of the Aria storage engine enables faster complex queries (queries which normally use disk-based temporary tables). The `Aria` storage engine is used for internal temporary tables, which should give a speedup when doing complex selects. Aria is usually faster for temporary tables when compared to MyISAM because Aria caches row data in memory and normally doesn't have to write the temporary rows to disk.
- The test suite has been extended and now runs much faster than before, even though it tests more things.

## Extensions & New Features

We've added a lot of [new features to MariaDB](#). If a patch or feature is useful, safe, and stable — we make every effort to include it in MariaDB. The most notable features are:

- Support introduced for `System-versioned tables`. Allows queries to access both current and historic data, aiding in managing retention, analysis and point-in-time recovery. — new in 10.3
- `ALTER TABLE... DROP COLUMN` can now run as Instant operations. Can also now change the ordering of columns. — new in 10.4
- Support introduced for password expiration, using the `user password expiry` — new in 10.4
- In order to support the use of multiple authentication plugins for a single user, the

system table has been retired in favor of the

```
mysql.glob_priv
```

system table. — new in 10.4

- The [unix\\_socket authentication plugin](#) is now the default on Unix-like systems. This represents a major change to authentication in MariaDB — new in 10.4
- Support introduced for [Optimizer Trace](#), which provides detailed information on how the Optimizer processes queries. To enable Optimizer Trace, set the

```
optimizer_trace
```

system variable — new in 10.4

- The MariaDB SQL/PL stored procedure dialect (enabled with [sql\\_mode=ORACLE](#)) now supports Oracle style packages. Support for the following statements are available: [CREATE PACKAGE](#), [CREATE PACKAGE BODY](#), [DROP PACKAGE](#), [DROP PACKAGE BODY](#), [SHOW CREATE PACKAGE](#), [SHOW CREATE PACKAGE BODY](#) — new in 10.4
- Automatic collection of [Engine Independent Table Statistics](#) — new in 10.4
- Support for the use of parentheses (brackets) for specifying precedence in the ordering of execution for

```
SELECT
```

statements and [Table Value Operations](#), (including the use of [UNION](#), [EXCEPT](#), [INTERSECT](#) operations) — new in 10.4

- Support for [anchored data types](#) added to local stored procedure variables. — new in 10.3
- Support added for [Stored Aggregate](#) functions — new in 10.3
- Oracle compatible [SUBSTR\(\)](#) function is available — new in 10.3
- Oracle compatible [SEQUENCE](#) support is provided — new in 10.3
- Support for [anchored data types](#) added to [stored routine](#) variables — new in 10.3
- Support for [anchored data types](#) added to stored routine parameters — new in 10.3
- [Cursors](#) with parameters are now supported — new in 10.3
- [INVISIBLE](#) columns are now supported — new in 10.3
- Instant [ADD COLUMN](#) is now available for InnoDB — new in 10.3
- [Window functions](#) are supported — new in 10.2
- Number of supported decimals in [DECIMAL](#) has increased from

```
30  
to  
38
```

— new in 10.2

- [Recursive Common Table Expressions](#) — new in 10.2

- New [WITH](#) statement.

```
WITH
```

is a common table expression that allows one to refer to a subquery expression many times in a query — new in 10.2

- [CHECK CONSTRAINT](#) — new in 10.2

- [DEFAULT expression](#), including

```
DEFAULT  
for  
BLOB  
and  
TEXT
```

— new in 10.2

- Added catchall for [list partitions](#) — new in 10.2
- Oracle-style [EXECUTE IMMEDIATE](#) statement — new in 10.2
- Several new [JSON functions](#) — new in 10.2
- Microsecond Precision in Processlist
- [Table Elimination](#)
- [Virtual Columns](#) — new in 5.2
- Microseconds in MariaDB — new in 5.3
- Extended User Statistics — new in 5.2
- [KILL](#) all queries for a user — new in 5.3,
- [KILL QUERY ID](#) - terminates the query by [query\\_id](#), leaving the connection intact — new in 10.0.5,
- Pluggable Authentication — new in 5.2
- Storage-engine-specific [CREATE TABLE](#) — new in 5.2
- Enhancements to [INFORMATION SCHEMA.PLUGINS](#) table — new in 5.2
- Group commit for the binary log . This makes replication notably faster! — new in 5.3
- Added

--

```
rewrite-db
```

[mysqldbinlog](#) option to change the used database — new in 5.2

- Progress reporting for

[ALTER TABLE](#)

and

[LOAD DATA INFILE](#)

— new in 5.3

- Faster joins and subqueries — new in 5.3
- HandlerSocket and faster HANDLER calls — new in 5.3
- Dynamic Columns support — new in 5.3
- GIS Functionality — new in 5.3
- Multi-source replication — new in 10.0
- Global Transaction ID — new in 10.0
- SHOW EXPLAIN gives the EXPLAIN plan for a query running in another thread. — new in 10.0
- Roles — new in 10.0
- PCRE Regular Expressions (including

[REGEXP\\_REPLACE\(\)](#)

) — new in 10.0

- CREATE OR REPLACE
- DELETE ... RETURNING — new in 10.0
- MariaDB supports more collations than MySQL.

For a full list, please see [features for each release](#)

## Better Testing

- More tests in the test suite.
- Bugs in tests fixed.
- Test builds with different configure options to get better feature testing.
- Remove invalid tests. (e.g. don't test feature "X" if that feature is not in the tested build)

## Fewer Warnings and Fewer Bugs

- Bugs are bad. Fix as many bugs as possible and try to not introduce new ones.
- Compiler warnings are also bad. Eliminate as many compiler warnings as possible.

## Truly Open Source

- All code in MariaDB is released under GPL, LGPL or BSD.
- MariaDB does not have closed source modules like the ones that can be found in MySQL Enterprise Edition. In fact, all the closed source features in MySQL 5.5 Enterprise Edition are found in the MariaDB open source version.
- MariaDB client libraries (for C, for Java (JDBC), for Windows (ODBC)) are released under LGPL to allow linking with closed source software. MySQL client libraries are released under GPL that does not allow linking with closed source software.
- MariaDB includes test cases for all fixed bugs. Oracle doesn't provide test cases for new bugs fixed in MySQL 5.5.
- All [bugs](#) and [development plans](#) are public.
- MariaDB is [developed by the community](#) in true open source spirit.

## Related Links

- [Compatibility between MariaDB and MySQL](#)
- [Moving from MySQL](#)
- [Troubleshooting Installation Issues](#)

## 2.1.14.1.2 MariaDB versus MySQL: Compatibility

See also [MariaDB vs MySQL - Features](#)

## Contents

- 1. Replacement for MySQL
  - 1. Drop-in Compatibility of Specific MariaDB Versions
- 2. Replication Compatibility
  - 1. MySQL 5.7
  - 2. MySQL 8.0
- 3. Incompatibilities between Currently Supported MariaDB Versions and MySQL
  - 1. Incompatibilities between MariaDB 10.7 and MySQL 8.0
  - 2. Incompatibilities between MariaDB 10.6 and MySQL 8.0
  - 3. Incompatibilities between MariaDB 10.5 and MySQL 8.0
  - 4. Incompatibilities between MariaDB 10.4 and MySQL 8.0
  - 5. Incompatibilities between MariaDB 10.3 and MySQL 5.7
  - 6. Incompatibilities between MariaDB 10.2 and MySQL 5.7
- 4. Incompatibilities between Currently Unsupported MariaDB Versions and MySQL
  - 1. Incompatibilities between MariaDB 10.1 and MySQL 5.7
  - 2. Incompatibilities between MariaDB 10.0 and MySQL 5.6
  - 3. Incompatibilities between MariaDB 5.5 and MySQL 5.5
  - 4. Incompatibilities between MariaDB 5.3 and MySQL 5.1
  - 5. Incompatibilities between MariaDB 5.2 and MySQL 5.1
  - 6. Incompatibilities between MariaDB 5.1 and MySQL 5.1
- 5. Old, Unsupported Configuration Options
- 6. Replacing a MySQL RPM
- 7. Incompatibilities between MariaDB and MySQL-Proxy
- 8. Related Links

## Replacement for MySQL

Until [MariaDB 5.5](#), MariaDB versions functioned as a "drop-in replacement" for the equivalent MySQL version, with some limitations. From [MariaDB 10.0](#), it is usually still very easy to upgrade from MySQL.

- MariaDB's data files are generally binary compatible with those from the equivalent MySQL version.
  - All filenames and paths are generally the same.
  - Data and table definition files (.frm) files are binary compatible.
  - See note below for an incompatibility with views!
- MariaDB's client protocol is binary compatible with MySQL's client protocol.
  - All client APIs and structs are identical.
  - All ports and sockets are generally the same.
  - All MySQL connectors (PHP, Perl, Python, Java, .NET, MyODBC, Ruby, MySQL C connector etc) work unchanged with MariaDB.
  - There are some [installation issues with PHP5](#) that you should be aware of (a bug in how the old PHP5 client checks library compatibility).

This means that for many cases, you can just uninstall MySQL and [install MariaDB](#) and you are good to go. There is not generally any need to convert any data files.

However, you must still run [mysql\\_upgrade](#) to finish the upgrade. This is needed to ensure that your mysql privilege and event tables are updated with the new fields MariaDB uses.

That said, MariaDB has a lot of [new options, extension, storage engines and bug fixes](#) that are not in MySQL. You can find the feature set for the different MariaDB versions on the [What is in the different MariaDB Releases](#) page.

## Drop-in Compatibility of Specific MariaDB Versions

[MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#) function as limited drop-in replacements for MySQL 5.7, as far as [InnoDB](#) is concerned. However, the implementation differences continue to grow in each new MariaDB version.

[MariaDB 10.0](#) and [MariaDB 10.1](#) function as limited drop-in replacements for MySQL 5.6, as far as [InnoDB](#) is concerned. However, there are some implementation differences in some features.

[MariaDB 5.5](#) functions as a drop-in replacement for MySQL 5.5.

[MariaDB 5.1](#), [MariaDB 5.2](#), and [MariaDB 5.3](#) function as drop-in replacements for MySQL 5.1.

## Replication Compatibility

Replication compatibility depends on:

- The MariaDB Server version
- The MySQL Server version
- The role of each server

Replication compatibility details are described below for each MySQL version that is still currently supported.

For replication compatibility details between MariaDB versions, see [Cross-Version Replication Compatibility](#).

### MySQL 5.7

MariaDB Server 10.2 and later can replicate from a MySQL 5.7 primary server.

MariaDB Server does not support the MySQL implementation of Global Transaction IDs (GTIDs), so the MariaDB replica server must use the binary log file and position for replication. If GTID mode is enabled on the MySQL primary server, the MariaDB replica server will remove the MySQL GTID events and replace them with MariaDB GTID events.

Although MariaDB Server and MySQL 5.7 are compatible at the replication level, they may have some incompatibilities at the SQL (detailed below). Those differences can cause replication failures in some cases. To decrease the risk of compatibility issues, it is recommended to set `binlog_format` to `ROW`. When you want to replicate from MySQL 5.7 to MariaDB Server, it is recommended to test your application, so that any compatibility issues can be found and fixed.

MariaDB can't make any claims about whether a MySQL 5.7 replica server can replicate from a MariaDB primary server.

### MySQL 8.0

MariaDB Server cannot replicate from a MySQL 8.0 primary server, because MySQL 8.0 has a binary log format that is incompatible.

## Incompatibilities between Currently Supported MariaDB Versions and MySQL

### Incompatibilities between MariaDB 10.7 and MySQL 8.0

See [Incompatibilities and Feature Differences Between MariaDB 10.7 and MySQL 8.0](#) for details.

### Incompatibilities between MariaDB 10.6 and MySQL 8.0

See [Incompatibilities and Feature Differences Between MariaDB 10.6 and MySQL 8.0](#) for details.

### Incompatibilities between MariaDB 10.5 and MySQL 8.0

See [Incompatibilities and Feature Differences Between MariaDB 10.5 and MySQL 8.0](#) for details.

### Incompatibilities between MariaDB 10.4 and MySQL 8.0

See [Incompatibilities and Feature Differences Between MariaDB 10.4 and MySQL 8.0](#) for details.

### Incompatibilities between MariaDB 10.3 and MySQL 5.7

See [Incompatibilities and Feature Differences Between MariaDB 10.3 and MySQL 5.7](#) for details.

### Incompatibilities between MariaDB 10.2 and MySQL 5.7

See [Incompatibilities and Feature Differences Between MariaDB 10.2 and MySQL 5.7](#) for details.

## Incompatibilities between Currently Unsupported MariaDB Versions and MySQL

### Incompatibilities between MariaDB 10.1 and MySQL 5.7

- MariaDB 10.1 and above does not support MySQL 5.7's packed JSON objects. MariaDB follows the SQL standard and stores the JSON as a normal TEXT/BLOB. If you want to replicate JSON columns from MySQL to MariaDB, you should store JSON objects in MySQL in a TEXT column or use statement based replication. If you are using JSON columns and want to upgrade to MariaDB, you can either convert the JSON columns to TEXT or use `mysqldump` to copy these tables to MariaDB. In MySQL, JSON is compared according to json values. In MariaDB JSON strings are normal strings and compared as strings.
- MariaDB 10.1's InnoDB encryption is implemented differently than MySQL 5.7's InnoDB encryption.
- MariaDB 10.1 does not support the ngram and MeCab full-text parser plugins - MDEV-10267 , MDEV-10268 .
- MariaDB 10.1 does not support multiple triggers for a table - MDEV-6112 . This is fixed in MariaDB 10.2
- MariaDB 10.1 does not support `CREATE TABLESPACE` for InnoDB.
- MariaDB 10.1 does not support MySQL 5.7's "native" InnoDB partitioning handler.
- MariaDB does not support MySQL 5.7's X protocol.
- MariaDB 10.1 does not support the use of multiple triggers of the same type for a table. This feature was introduced in MariaDB 10.2.2 .
- MariaDB 10.1 does not support MySQL 5.7's transportable tablespaces for partitioned InnoDB tables. `ALTER TABLE ... {DISCARD|IMPORT}` PARTITION is not supported. For a workaround see the following blog post .
- MariaDB 10.1 does not support MySQL 5.7's online undo tablespace truncation. However, this feature was added to MariaDB 10.2 .
- MySQL 5.7 features a new implementation of the
 

```
performance_schema  
and a  
sys  
schema wrapper. These are not yet supported in MariaDB.
```
- MySQL 5.7 adds multi-source replication and replication channels. Multi-source replication was added to MariaDB previously, in MariaDB 10.0 , and uses a different syntax.
- MySQL 5.7 adds group replication. This feature is incompatible with MariaDB's galera-cluster replication.
- MariaDB 10.1 does not support MySQL 5.7's,
 

```
ACCOUNT LOCK/UNLOCK  
syntax for  
CREATE USER  
and  
ALTER USER  
statements.
```
- MariaDB 10.1 does not support MySQL 5.7's
 

```
ALTER TABLE...RENAME INDEX  
statements.
```
- MariaDB 10.1 does not support MySQL 5.7's
 

```
STACKED  
operation for
```
- MariaDB 10.1 does not support MySQL 5.7's
 

```
GET DIAGNOSTICS
```
- MariaDB 10.1 does not support MySQL 5.7's
 

```
{WITH|WITHOUT} VALIDATION  
syntax for  
ALTER TABLE.. EXCHANGE PARTITION  
statements.
```
- MariaDB does not support the optional `init_vector` argument for `AES_ENCRYPT` and `AES_DECRYPT` or the `block_encryption_mode` variable - MDEV-9069
- MariaDB does not support the
 

```
--initialize  
option. Use
```
- MariaDB does not support the
 

```
mysql_install_db
```
- Instead of - MDEV-19010
  - Also see Incompatibilities between MariaDB 10.0 and MySQL 5.6.
  - Also see a detailed breakdown of System variable differences between MariaDB 10.1 and MySQL 5.7 .

## Incompatibilities between MariaDB 10.0 and MySQL 5.6

- MySQL does not support MariaDB's Spider Storage Engine .
- All MySQL binaries (
 

```
mysqld  
, myisamchk etc.) give a warning if one uses a prefix of an option (such as  
--big-table  
instead of  
--big-tables  
)
```

MariaDB binaries work in the same way as most other Unix commands and don't give warnings when using unique prefixes.
- MariaDB GTID is not compatible with MySQL 5.6. This means that one can't have MySQL 5.6 as a slave for MariaDB 10.0 . However MariaDB 10.0 can be a slave of MySQL 5.6 or any earlier MySQL/MariaDB version. Note that MariaDB and MySQL also have different GTID system variables , so these need to be adjusted when migrating.
- MariaDB 10.0 multi-source replication is not supported in MySQL 5.6.
- To make `CREATE TABLE ... SELECT` work the same way in statement based and row based replication it's by default executed as `CREATE OR`

[REPLACE TABLE](#) on the slave. One benefit of this is that if the slave dies in the middle of CREATE ... SELECT it will be able to continue.

- One can use the [slave-dll-exec-mode](#) variable to specify how

```
CREATE TABLE  
and  
DROP TABLE  
is replicated.
```

- See also a detailed breakdown of [System variable differences between MariaDB 10.0 and MySQL 5.6](#) .
- MySQL 5.6 has [performance schema](#) enabled by default. For performance reasons [MariaDB 10.0](#) has it disabled by default. You can enable it by starting

```
mysqld  
with the option  
--performance-schema
```

- [MariaDB 10.0](#) does not support the MySQL Memcached plugin. However, data stored using memcached can be retrieved because the data is stored as InnoDB tables. MariaDB is able to start successfully with an error message of not being able to find libmemcached.so library.
- Users created with MySQL's SHA256 password algorithm cannot be used in [MariaDB 10.0](#) as MariaDB does not include MySQL's sha256\_password plugin.
- [MariaDB 10.0](#) does not support delayed replication - [MDEV-7145](#) .
- Also see a detailed breakdown of [System variable differences between MariaDB 10.0 and MySQL 5.6](#) .
- The low-level temporal format used by TIME, DATETIME and TIMESTAMP is different in MySQL 5.6 and [MariaDB 10.0](#) . (In [MariaDB 10.1](#) , the MySQL implementation is used by default - see [mysql56\\_temporal\\_format](#) .)
- MariaDB implements some changes in the SQL query optimizer over what's available in MySQL. This can result in

```
EXPLAIN
```

statements showing different plans.

- MySQL delayed replication, (through  
[MASTER\\_DELAY](#)  
( ), is not supported in [MariaDB 10.0](#) , it was implemented in [MariaDB 10.2.5](#)
- MariaDB does not support the optional *init\_vector* argument for [AES\\_ENCRYPT](#) and [AES\\_DECRYPT](#) or the [block\\_encryption\\_mode](#) variable - [MDEV-9069](#)

## Incompatibilities between [MariaDB 5.5](#) and MySQL 5.5

- Views with definition ALGORITHM=MERGE or ALGORITHM=TEMPTABLE got accidentally swapped between MariaDB and MySQL! You have to re-create views created with either of these definitions!
- [INSERT IGNORE](#) also gives warnings for duplicate key errors. You can turn this off by setting  
[OLD\\_MODE=NO\\_DUP\\_KEY\\_WARNINGS\\_WITH\\_IGNORE](#)  
(see [OLD\\_MODE](#) ).
- Before [MariaDB 5.5.31](#) ,  
`X'HHHH'`, the standard SQL syntax for binary string literals, erroneously worked in the same way as  
`0xHHHH`, which could work as a number or string depending on the context. In 5.5.31 this was fixed to behave as a string in all contexts (and never as a number), introducing an incompatibility with previous versions of MariaDB, and all versions of MySQL. See [CAST](#) and [Hexadecimal Literals](#) for more details and examples.
- MariaDB [dynamic columns](#) are not supported by MySQL.
- MariaDB [virtual columns](#) are not supported by MySQL.
- MariaDB's [HandlerSocket plugin](#) is not supported by MySQL.
- MariaDB's [Cassandra Storage Engine](#) is not supported by MySQL.
- As of [MariaDB 5.5.35](#) , [EXTRACT \(HOUR FROM ...\)](#) adheres to the SQL standard and returns a result from 0 to 23. In MySQL, and earlier versions of MariaDB, the result can be greater than 23.
- See also a detailed breakdown of [System variable differences between MariaDB 5.5 and MySQL 5.5](#) .

## Incompatibilities between [MariaDB 5.3](#) and MySQL 5.1

- Views with definition ALGORITHM=MERGE or ALGORITHM=TEMPTABLE got accidentally swapped between [MariaDB 5.2](#) and [MariaDB 5.3](#) ! You have to re-create views created with either of these definitions!
- A few error messages related to wrong conversions are different as MariaDB provides more information in the message about what went wrong.
- Error numbers for MariaDB-specific errors have been moved to start from 1900 so as not to conflict with MySQL errors.
- Microseconds now work in all contexts; MySQL, in some contexts, lost the microsecond part from datetime and time.
- [UNIX\\_TIMESTAMP](#) (constant-date-string) returns a timestamp with 6 decimals in MariaDB while MySQL returns it without a decimal. This can cause a problem if you are using [UNIX\\_TIMESTAMP\(\)](#) as a partitioning function. You can fix this by using [FLOOR](#) ([UNIX\\_TIMESTAMP\(..\)](#)) or changing the date string to a date number, like 20080101000000.
- MariaDB performs stricter checking of date, datetime and timestamp values. For example [UNIX\\_TIMESTAMP](#) ('x') now returns NULL instead of 0.
- The old  
`--maria-`  
startup options are removed. You should use the  
`--aria-`  
prefix instead. ( [MariaDB 5.2](#) supports both

- ```
--maria-
and
--aria-
)
```
- SHOW PROCESSLIST**

has an extra  
Progress  
column which shows progress for some commands. You can disable it by starting  
mysqld  
with either  
--old-mode=NO\_PROGRESS\_INFO  
or with the  
--old  
flag (see [OLD\\_MODE](#) ).
- INFORMATION\_SCHEMA.PROCESSLIST**  
has three new columns for progress reporting:  
STAGE  
,  
MAX\_STAGE  
, and  
PROGRESS
- **Long comments** which start with  
/\*M!  
or  
/\*M!#####  
are executed.
  - If you use [max\\_user\\_connections=0](#) (which means any number of connections) when starting mysqld, you can't change the global variable anymore while mysqld remains running. This is because when mysqld is started with  
max\_user\_connections=0  
it does not allocate counting structures (which also involve a mutex for each connection). This would lead to wrong counters if you later changed the variable. If you want to be able to change this variable at runtime, set it to a high value at startup.
  - You can set  
max\_user\_connections  
(both the global variable and the  
GRANT  
option) to  
-1  
to stop users from connecting to the server. The global  
max\_user\_connections  
variable does not affect users with the  
SUPER  
privilege.
  - The [IGNORE](#) directive does not ignore all errors (like fatal errors), only things that are safe to ignore.

## Incompatibilities between MariaDB 5.2 and MySQL 5.1

The list is the same as between [MariaDB 5.1](#) and MySQL 5.1, with one addition:

- A new [SQL\\_MODE](#) value was added:  
IGNORE\_BAD\_TABLE\_OPTIONS  
. If it is not set, using a table, field, or index attribute (option) that is not supported by the chosen storage engine will cause an error. This change might cause warnings in the error log about incorrectly defined tables from the  
mysql  
database, fix that with [mysql\\_upgrade](#) .

For all practical purposes, [MariaDB 5.2](#) is a drop in replacement for [MariaDB 5.1](#) and MySQL 5.1.

## Incompatibilities between MariaDB 5.1 and MySQL 5.1

In some few cases MariaDB has to be incompatible to allow MariaDB to provide more and better information than MySQL.

Here is the list of all known user level incompatibilities you may see when using [MariaDB 5.1](#) instead of MySQL 5.1.

- The installation package names start with MariaDB instead of MySQL.
- Timings may be different as MariaDB is in many cases faster than MySQL.
- mysqld in MariaDB also reads the  
[mariadb]  
sections of your my.cnf files.

- You can't use a binary only storage engine library with MariaDB if it's not compiled for exactly the same MariaDB version. (This is because the server internal structure THD is different between MySQL and MariaDB. This is common also between different MySQL versions). This should not be a problem as most people don't load new storage engines and MariaDB comes with [more storage engines](#) than MySQL.
- ```
CHECKSUM TABLE
may give different result as MariaDB doesn't ignore NULL's in the columns as MySQL 5.1 does (Future MySQL versions should calculate
checksums the same way as MariaDB). You can get the 'old style' checksum in MariaDB by starting mysqld with the
--old
option. Note however that that the MyISAM and Aria storage engines in MariaDB are using the new checksum internally, so if you are
using
--old
,the
CHECKSUM
command will be slower as it needs to calculate the checksum row by row.
```
- The slow query log has [more information](#) about the query, which may be a problem if you have a script which parses the slow query log.
- MariaDB by default takes a bit more memory than MySQL because we have by default enabled the [Aria storage engine](#) for handling internal temporary tables. If you need MariaDB to take very little memory (at the expense of performance), you can set the value of
 

```
aria_pagecache_buffer_size
to
1M
(the default is
128M
).
```
- If you are using [new command options](#), [new features of MariaDB](#) or [new storage engines](#), you can't move easily back and forth between MySQL and MariaDB anymore.

## Old, Unsupported Configuration Options

If you are using any of the following options in your

```
/etc/my.cnf
or other
my.cnf
file you should remove them. This is also true for MySQL 5.1 or newer:
```

- ```
skip-bdb
```

## Replacing a MySQL RPM

If you uninstalled a MySQL RPM to install MariaDB, note that the MySQL RPM on uninstall renames

```
/etc/my.cnf
to
/etc/my.cnf.rpmsave
```

After installing MariaDB you should do the following to restore your old configuration options:

```
mv -vi /etc/my.cnf.rpmsave /etc/my.cnf
```

## Incompatibilities between MariaDB and MySQL-Proxy

A MySQL client API is able to connect to MariaDB using MySQL-Proxy but a MariaDB client API will receive progress reporting informations that MySQL-Proxy does not implement, to get full compatibility in all case just disable progress reporting on the client or server side.

Another option is to use the [MariaDB MaxScale proxy](#), that works with both MySQL and MariaDB.

## Related Links

- [MariaDB vs MySQL - Features](#)
- [Moving from MySQL to MariaDB](#)
- [Troubleshooting Installation Issues](#)
- [Projects and applications that works with MariaDB](#)

### 2.1.14.1.3 Upgrading from MySQL to MariaDB

#### 2.1.14.1.4

# 3 High Availability & Performance Tuning

## 4 Columns, Storage Engines, and Plugins

### 4.1 Data Types

#### 4.1.1 Numeric Data Types

##### 4.1.1.1 Numeric Data Type Overview

###### Contents

- 1. [SIGNED, UNSIGNED and ZEROFILL](#)
  - 1. [Examples](#)
- 2. [Range](#)
  - 1. [Examples](#)
- 3. [Auto\\_increment](#)

There are a number of numeric data types:

- [TINYINT](#)
- [BOOLEAN](#) - Synonym for TINYINT(1)
- [INT1](#) - Synonym for TINYINT
- [SMALLINT](#)
- [INT2](#) - Synonym for SMALLINT
- [MEDIUMINT](#)
- [INT3](#) - Synonym for MEDIUMINT
- [INT](#), [INTEGER](#)
- [INT4](#) - Synonym for INT
- [BIGINT](#)
- [INT8](#) - Synonym for BIGINT
- [DECIMAL](#), [DEC](#), [NUMERIC](#), [FIXED](#)
- [FLOAT](#)
- [DOUBLE](#), [DOUBLE PRECISION](#), [REAL](#)
- [BIT](#)

See the specific articles for detailed information on each.

### SIGNED, UNSIGNED and ZEROFILL

Most numeric types can be defined as

SIGNED  
,  
UNSIGNED  
or  
ZEROFILL  
, for example:

`TINYINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]`

If

SIGNED  
, or no attribute, is specified, a portion of the numeric type will be reserved for the sign (plus or minus). For example, a TINYINT SIGNED can range from -128 to 127.

If

UNSIGNED  
is specified, no portion of the numeric type is reserved for the sign, so for integer types range can be larger. For example, a TINYINT UNSIGNED can range from 0 to 255. Floating point and fixed-point types also can be  
UNSIGNED  
, but this only prevents negative values from being stored and doesn't alter the range.

If

ZEROFILL  
is specified, the column will be set to UNSIGNED and the spaces used by default to pad the field are replaced with zeros.  
ZEROFILL  
is ignored in expressions or as part of a UNION .  
ZEROFILL  
is a non-standard MySQL and MariaDB enhancement.

Note that although the preferred syntax indicates that the attributes are exclusive, more than one attribute can be specified.

Until MariaDB 10.2.7 ( MDEV-8659 ), any combination of the attributes could be used in any order, with duplicates. In this case:

- the presence of  
    ZEROFILL  
    makes the column  
    UNSIGNED ZEROFILL
- the presence of  
    UNSIGNED  
    makes the column  
    UNSIGNED

From MariaDB 10.2.8 , only the following combinations are supported:

- SIGNED
- UNSIGNED
- ZEROFILL
- UNSIGNED ZEROFILL
- ZEROFILL UNSIGNED

The latter two should be replaced with simply

ZEROFILL

, but are still accepted by the parser.

## Examples

```
CREATE TABLE zf (
    i1 TINYINT SIGNED,
    i2 TINYINT UNSIGNED,
    i3 TINYINT ZEROFILL
);

INSERT INTO zf VALUES (2,2,2);

SELECT * FROM zf;
+---+---+---+
| i1 | i2 | i3 |
+---+---+---+
|   2 |   2 |  002 |
+---+---+---+
```

## Range

When attempting to add a value that is out of the valid range for the numeric type, MariaDB will react depending on the strict SQL\_MODE setting.

If strict\_mode has been set (the default from MariaDB 10.2.4 ), MariaDB will return an error.

If strict\_mode has not been set (the default until MariaDB 10.2.3 ), MariaDB will adjust the number to fit in the field, returning a warning.

## Examples

With strict\_mode set:

```

SHOW VARIABLES LIKE 'sql_mode';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| sql_mode      | STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----+-----+

CREATE TABLE ranges (i1 TINYINT, i2 SMALLINT, i3 TINYINT UNSIGNED);

INSERT INTO ranges VALUES (257,257,257);
ERROR 1264 (22003): Out of range value for column 'i1' at row 1

SELECT * FROM ranges;
Empty set (0.10 sec)

```

With strict\_mode unset:

```

SHOW VARIABLES LIKE 'sql_mode%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| sql_mode      |       |
+-----+-----+

CREATE TABLE ranges (i1 TINYINT, i2 SMALLINT, i3 TINYINT UNSIGNED);

INSERT INTO ranges VALUES (257,257,257);
Query OK, 1 row affected, 2 warnings (0.00 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1264 | Out of range value for column 'i1' at row 1 |
| Warning | 1264 | Out of range value for column 'i3' at row 1 |
+-----+-----+
2 rows in set (0.00 sec)

SELECT * FROM ranges;
+-----+-----+
| i1 | i2 | i3 |
+-----+-----+
| 127 | 257 | 255 |
+-----+-----+

```

## Auto\_increment

The

`AUTO_INCREMENT`

attribute can be used to generate a unique identity for new rows. For more details, see [auto\\_increment](#).

### 4.1.1.2 TINYINT

#### Syntax

```
TINYINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

#### Description

A very small `integer`. The signed range is -128 to 127. The unsigned range is 0 to 255. For details on the attributes, see [Numeric Data Type Overview](#).

`INT1` is a synonym for

`TINYINT`

. `BOOL` and `BOOLEAN` are synonyms for

TINYINT(1)

## Examples

```
CREATE TABLE tinyints (a TINYINT,b TINYINT UNSIGNED,c TINYINT ZEROFILL);
```

With `strict_mode` set, the default from MariaDB 10.2.4 :

```
INSERT INTO tinyints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO tinyints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO tinyints VALUES (-10,10,10);

SELECT * FROM tinyints;
+----+----+----+
| a | b | c |
+----+----+----+
| -10 | 10 | 010 |
+----+----+----+

INSERT INTO tinyints VALUES (128,128,128);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO tinyints VALUES (127,128,128);

SELECT * FROM tinyints;
+----+----+----+
| a | b | c |
+----+----+----+
| -10 | 10 | 010 |
| 127 | 128 | 128 |
+----+----+----+
```

With `strict_mode` unset, the default until MariaDB 10.2.3 :

```

INSERT INTO tinyints VALUES (-10,-10,-10);
Query OK, 1 row affected, 2 warnings (0.08 sec)
Warning (Code 1264): Out of range value for column 'b' at row 1
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO tinyints VALUES (-10,10,-10);
Query OK, 1 row affected, 1 warning (0.11 sec)
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO tinyints VALUES (-10,10,10);

SELECT * FROM tinyints;
+-----+-----+
| a    | b    | c    |
+-----+-----+
| -10  | 0   | 000 |
| -10  | 10  | 000 |
| -10  | 10  | 010 |
+-----+-----+

INSERT INTO tinyints VALUES (128,128,128);
Query OK, 1 row affected, 1 warning (0.19 sec)
Warning (Code 1264): Out of range value for column 'a' at row 1

INSERT INTO tinyints VALUES (127,128,128);

SELECT * FROM tinyints;
+-----+-----+
| a    | b    | c    |
+-----+-----+
| -10  | 0   | 000 |
| -10  | 10  | 000 |
| -10  | 10  | 010 |
| 127  | 128 | 128 |
| 127  | 128 | 128 |
+-----+-----+

```

## See Also

- [Numeric Data Type Overview](#)
- [SMALLINT](#)
- [MEDIUMINT](#)
- [INTEGER](#)
- [BIGINT](#)
- [BOOLEAN](#)

## 4.1.1.3 BOOLEAN

### Syntax

BOOL, BOOLEAN

### Description

These types are synonyms for [TINYINT\(1\)](#). A value of zero is considered false. Non-zero values are considered true.

However, the values TRUE and FALSE are merely aliases for 1 and 0. See [Boolean Literals](#), as well as the [IS operator](#) for testing values against a boolean.

### Examples

```

CREATE TABLE boo (i BOOLEAN);

DESC boo;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| i     | tinyint(1) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+

```

```

SELECT IF(0, 'true', 'false');
+-----+
| IF(0, 'true', 'false') |
+-----+
| false |
+-----+

SELECT IF(1, 'true', 'false');
+-----+
| IF(1, 'true', 'false') |
+-----+
| true |
+-----+

SELECT IF(2, 'true', 'false');
+-----+
| IF(2, 'true', 'false') |
+-----+
| true |
+-----+

```

TRUE and FALSE as aliases for 1 and 0:

```

SELECT IF(0 = FALSE, 'true', 'false');
+-----+
| IF(0 = FALSE, 'true', 'false') |
+-----+
| true |
+-----+

SELECT IF(1 = TRUE, 'true', 'false');
+-----+
| IF(1 = TRUE, 'true', 'false') |
+-----+
| true |
+-----+

SELECT IF(2 = TRUE, 'true', 'false');
+-----+
| IF(2 = TRUE, 'true', 'false') |
+-----+
| false |
+-----+

SELECT IF(2 = FALSE, 'true', 'false');
+-----+
| IF(2 = FALSE, 'true', 'false') |
+-----+
| false |
+-----+

```

The last two statements display the results shown because 2 is equal to neither 1 nor 0.

## See Also

- [Boolean Literals](#)
- [IS operator](#)

## 4.1.1.4 SMALLINT

### Syntax

```
SMALLINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

### Description

A small [integer](#). The signed range is -32768 to 32767. The unsigned range is 0 to 65535.

If a column has been set to ZEROFILL, all values will be prepended by zeros so that the SMALLINT value contains a number of M digits.

**Note:** If the `ZEROFILL` attribute has been specified, the column will automatically become `UNSIGNED`.

```
INT2  
is a synonym for  
SMALLINT
```

For more details on the attributes, see [Numeric Data Type Overview](#).

## Examples

```
CREATE TABLE smallints (a SMALLINT,b SMALLINT UNSIGNED,c SMALLINT ZEROFILL);
```

With `strict_mode` set, the default from MariaDB 10.2.4 :

```
INSERT INTO smallints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO smallints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO smallints VALUES (-10,10,10);

INSERT INTO smallints VALUES (32768,32768,32768);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO smallints VALUES (32767,32768,32768);

SELECT * FROM smallints;
+----+----+----+
| a | b | c |
+----+----+----+
| -10 | 10 | 00010 |
| 32767 | 32768 | 32768 |
+----+----+----+
```

With `strict_mode` unset, the default until MariaDB 10.2.3 :

```
INSERT INTO smallints VALUES (-10,-10,-10);
Query OK, 1 row affected, 2 warnings (0.09 sec)
Warning (Code 1264): Out of range value for column 'b' at row 1
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO smallints VALUES (-10,10,-10);
Query OK, 1 row affected, 1 warning (0.08 sec)
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO smallints VALUES (-10,10,10);

INSERT INTO smallints VALUES (32768,32768,32768);
Query OK, 1 row affected, 1 warning (0.04 sec)
Warning (Code 1264): Out of range value for column 'a' at row 1

INSERT INTO smallints VALUES (32767,32768,32768);

SELECT * FROM smallints;
+----+----+----+
| a | b | c |
+----+----+----+
| -10 | 0 | 00000 |
| -10 | 10 | 00000 |
| -10 | 10 | 00010 |
| 32767 | 32768 | 32768 |
| 32767 | 32768 | 32768 |
+----+----+----+
```

## See Also

- [Numeric Data Type Overview](#)
- [TINYINT](#)
- [MEDIUMINT](#)
- [INTEGER](#)

- BIGINT

## 4.1.1.5 MEDIUMINT

### Syntax

```
MEDIUMINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

### Description

A medium-sized integer. The signed range is -8388608 to 8388607. The unsigned range is 0 to 16777215.

ZEROFILL  
pads the integer with zeroes and assumes  
UNSIGNED  
(even if  
UNSIGNED  
is not specified).

INT3  
is a synonym for  
MEDIUMINT

For details on the attributes, see [Numeric Data Type Overview](#).

### Examples

```
CREATE TABLE mediumints (a MEDIUMINT,b MEDIUMINT UNSIGNED,c MEDIUMINT ZEROFILL);

DESCRIBE mediumints;
+-----+-----+-----+-----+
| Field | Type            | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| a     | mediumint(9)    | YES  |     | NULL    |       |
| b     | mediumint(8) unsigned | YES  |     | NULL    |       |
| c     | mediumint(8) unsigned zerofill | YES  |     | NULL    |       |
+-----+-----+-----+-----+
```

With `strict_mode` set, the default from MariaDB 10.2.4 :

```
INSERT INTO mediumints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO mediumints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO mediumints VALUES (-10,10,10);

INSERT INTO mediumints VALUES (8388608,8388608,8388608);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO mediumints VALUES (8388607,8388608,8388608);

SELECT * FROM mediumints;
+-----+-----+-----+
| a     | b     | c     |
+-----+-----+-----+
| -10   | 10   | 00000010 |
| 8388607 | 8388608 | 08388608 |
+-----+-----+-----+
```

With `strict_mode` unset, the default until MariaDB 10.2.3 :

```

INSERT INTO mediumints VALUES (-10,-10,-10);
Query OK, 1 row affected, 2 warnings (0.05 sec)
Warning (Code 1264): Out of range value for column 'b' at row 1
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO mediumints VALUES (-10,10,-10);
Query OK, 1 row affected, 1 warning (0.08 sec)
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO mediumints VALUES (-10,10,10);

INSERT INTO mediumints VALUES (8388608,8388608,8388608);
Query OK, 1 row affected, 1 warning (0.05 sec)
Warning (Code 1264): Out of range value for column 'a' at row 1

INSERT INTO mediumints VALUES (8388607,8388608,8388608);

SELECT * FROM mediumints;
+-----+-----+-----+
| a    | b    | c    |
+-----+-----+-----+
| -10  | 0   | 00000000 |
| -10  | 0   | 00000000 |
| -10  | 10  | 00000000 |
| -10  | 10  | 00000010 |
| 8388607 | 8388608 | 08388608 |
| 8388607 | 8388608 | 08388608 |
+-----+-----+-----+

```

## See Also

- [Numeric Data Type Overview](#)
- [TINYINT](#)
- [SMALLINT](#)
- [INTEGER](#)
- [BIGINT](#)

## 4.1.1.6 INT

### Syntax

```

INT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
INTEGER[(M)] [SIGNED | UNSIGNED | ZEROFILL]

```

### Description

A normal-size integer. When marked UNSIGNED, it ranges from 0 to 4294967295, otherwise its range is -2147483648 to 2147483647 (SIGNED is the default). If a column has been set to ZEROFILL, all values will be prepended by zeros so that the INT value contains a number of M digits. INTEGER is a synonym for INT.

**Note:** If the ZEROFILL attribute has been specified, the column will automatically become UNSIGNED.

```

INT4
is a synonym for
INT

```

For details on the attributes, see [Numeric Data Type Overview](#).

### Examples

```

CREATE TABLE ints (a INT,b INT UNSIGNED,c INT ZEROFILL);

```

With `strict_mode` set, the default from MariaDB 10.2.4 :

```

INSERT INTO ints VALUES (-10,-10,-10);
ERROR 1264 (22003): Out of range value for column 'b' at row 1

INSERT INTO ints VALUES (-10,10,-10);
ERROR 1264 (22003): Out of range value for column 'c' at row 1

INSERT INTO ints VALUES (-10,10,10);

INSERT INTO ints VALUES (2147483648,2147483648,2147483648);
ERROR 1264 (22003): Out of range value for column 'a' at row 1

INSERT INTO ints VALUES (2147483647,2147483648,2147483648);

SELECT * FROM ints;
+-----+-----+-----+
| a      | b      | c      |
+-----+-----+-----+
| -10    | 10    | 0000000010 |
| 2147483647 | 2147483648 | 2147483648 |
+-----+-----+-----+

```

With `strict_mode` unset, the default until MariaDB 10.2.3 :

```

INSERT INTO ints VALUES (-10,-10,-10);
Query OK, 1 row affected, 2 warnings (0.10 sec)
Warning (Code 1264): Out of range value for column 'b' at row 1
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO ints VALUES (-10,10,-10);
Query OK, 1 row affected, 1 warning (0.08 sec)
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO ints VALUES (-10,10,10);

INSERT INTO ints VALUES (2147483648,2147483648,2147483648);
Query OK, 1 row affected, 1 warning (0.07 sec)
Warning (Code 1264): Out of range value for column 'a' at row 1

INSERT INTO ints VALUES (2147483647,2147483648,2147483648);

SELECT * FROM ints;
+-----+-----+-----+
| a      | b      | c      |
+-----+-----+-----+
| -10    | 0     | 0000000000 |
| -10    | 10    | 0000000000 |
| -10    | 10    | 0000000010 |
| 2147483647 | 2147483648 | 2147483648 |
| 2147483647 | 2147483648 | 2147483648 |
+-----+-----+-----+

```

## See Also

- [Numeric Data Type Overview](#)
- [TINYINT](#)
- [SMALLINT](#)
- [MEDIUMINT](#)
- [BIGINT](#)

## 4.1.1.7 INTEGER

### Syntax

```
INTEGER[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

### Description

This type is a synonym for [INT](#).

## 4.1.1.8 BIGINT

### Syntax

```
BIGINT[(M)] [SIGNED | UNSIGNED | ZEROFILL]
```

### Description

A large integer. The signed range is

```
-9223372036854775808  
to  
9223372036854775807  
. The unsigned range is  
0  
to  
18446744073709551615
```

If a column has been set to ZEROFILL, all values will be prepended by zeros so that the BIGINT value contains a number of M digits.

**Note:** If the ZEROFILL attribute has been specified, the column will automatically become UNSIGNED.

For more details on the attributes, see [Numeric Data Type Overview](#).

SERIAL

is an alias for:

```
BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE
```

INT8

is a synonym for

BIGINT

### Examples

```
CREATE TABLE bigints (a BIGINT,b BIGINT UNSIGNED,c BIGINT ZEROFILL);
```

With `strict_mode` set, the default from MariaDB 10.2.4 :

```
INSERT INTO bigints VALUES (-10,-10,-10);  
ERROR 1264 (22003): Out of range value for column 'b' at row 1  
  
INSERT INTO bigints VALUES (-10,10,-10);  
ERROR 1264 (22003): Out of range value for column 'c' at row 1  
  
INSERT INTO bigints VALUES (-10,10,10);  
  
INSERT INTO bigints VALUES (9223372036854775808,9223372036854775808,9223372036854775808);  
ERROR 1264 (22003): Out of range value for column 'a' at row 1  
  
INSERT INTO bigints VALUES (9223372036854775807,9223372036854775808,9223372036854775808);  
  
SELECT * FROM bigints;  
+-----+-----+-----+  
| a      | b      | c      |  
+-----+-----+-----+  
| -10    | 10    | 000000000000000010 |  
| 9223372036854775807 | 9223372036854775808 | 09223372036854775808 |  
+-----+-----+-----+
```

With `strict_mode` unset, the default until MariaDB 10.2.3 :

```

INSERT INTO bigints VALUES (-10,-10,-10);
Query OK, 1 row affected, 2 warnings (0.08 sec)
Warning (Code 1264): Out of range value for column 'b' at row 1
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO bigints VALUES (-10,10,-10);
Query OK, 1 row affected, 1 warning (0.08 sec)
Warning (Code 1264): Out of range value for column 'c' at row 1

INSERT INTO bigints VALUES (-10,10,10);

INSERT INTO bigints VALUES (9223372036854775808,9223372036854775808,9223372036854775808);
Query OK, 1 row affected, 1 warning (0.07 sec)
Warning (Code 1264): Out of range value for column 'a' at row 1

INSERT INTO bigints VALUES (9223372036854775807,9223372036854775808,9223372036854775808);

SELECT * FROM bigints;
+-----+-----+
| a    | b      | c      |
+-----+-----+
| -10  | 0      | 00000000000000000000000000000000 |
| -10  | 10     | 00000000000000000000000000000000 |
| -10  | 10     | 00000000000000000000000000000010 |
| 9223372036854775807 | 9223372036854775808 | 09223372036854775808 |
| 9223372036854775807 | 9223372036854775808 | 09223372036854775808 |
+-----+-----+

```

## See Also

- [Numeric Data Type Overview](#)
- [TINYINT](#)
- [SMALLINT](#)
- [MEDIUMINT](#)
- [INTEGER](#)

## 4.1.1.9 DECIMAL

### Syntax

```
DECIMAL[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

A packed "exact" fixed-point number.

M  
is the total number of digits (the precision) and

D  
is the number of digits after the decimal point (the scale).

- The decimal point and (for negative numbers) the "-" sign are not counted in

M

.

- If

D

is

0

, values have no decimal point or fractional part and on [INSERT](#) the value will be rounded to the nearest DECIMAL

- The maximum number of digits (

M

) for

- The maximum number of supported decimals ( DECIMAL ) is 65.

D  
)  
is  
30  
before [Mariadb 10.2.1](#) and  
38  
afterwards.  
• If  
D  
is omitted, the default is  
0  
. If  
M  
is omitted, the default is  
10

UNSIGNED  
, if specified, disallows negative values.

ZEROFILL  
, if specified, pads the number with zeros, up to the total number of digits specified by M

All basic calculations (+, -, \*, /) with  
DECIMAL  
columns are done with a precision of 65 digits.

For more details on the attributes, see [Numeric Data Type Overview](#).

DEC  
,  
NUMERIC  
and  
FIXED  
are synonyms, as well as  
NUMBER  
in [Oracle mode from MariaDB 10.3](#).

## Examples

```
CREATE TABLE t1 (d DECIMAL UNSIGNED ZEROFILL);

INSERT INTO t1 VALUES (1),(2),(3),(4.0),(5.2),(5.7);
Query OK, 6 rows affected, 2 warnings (0.16 sec)
Records: 6  Duplicates: 0  Warnings: 2

Note (Code 1265): Data truncated for column 'd' at row 5
Note (Code 1265): Data truncated for column 'd' at row 6
```

```
SELECT * FROM t1;
+-----+
| d      |
+-----+
| 000000001 |
| 000000002 |
| 000000003 |
| 000000004 |
| 000000005 |
| 000000006 |
+-----+
```

With `strict_mode` set, the default from MariaDB 10.2.4 :

```
INSERT INTO t1 VALUES (-7);
ERROR 1264 (22003): Out of range value for column 'd' at row 1
```

With `strict_mode` unset, the default until MariaDB 10.2.3 :

```
INSERT INTO t1 VALUES (-7);
Query OK, 1 row affected, 1 warning (0.02 sec)
Warning (Code 1264): Out of range value for column 'd' at row 1

SELECT * FROM t1;
+-----+
| d    |
+-----+
| 0000000001 |
| 0000000002 |
| 0000000003 |
| 0000000004 |
| 0000000005 |
| 0000000006 |
| 0000000000 |
+-----+
```

## See Also

- [Numeric Data Type Overview](#)
- [Oracle mode from MariaDB 10.3](#)

## 4.1.1.10 DEC, NUMERIC, FIXED

### Syntax

```
DEC[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]
NUMERIC[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]
FIXED[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]
```

### Description

These types are synonyms for [DECIMAL](#) . The

`FIXED`  
synonym is available for compatibility with other database systems.

## 4.1.1.11 NUMBER

MariaDB starting with 10.3

```
NUMBER[(M[,D])] [SIGNED | UNSIGNED | ZEROFILL]
```

In Oracle mode from MariaDB 10.3 ,  
`NUMBER`  
is a synonym for [DECIMAL](#) .

## 4.1.1.12 FLOAT

### Syntax

```
FLOAT[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
```

### Contents

1. [Syntax](#)
2. [Description](#)

### Description

A small (single-precision) floating-point number (see [DOUBLE](#) for a regular-size floating point number). Allowable values are:

- -3.402823466E+38 to -1.175494351E-38
- 0
- 1.175494351E-38 to 3.402823466E+38.

These are the theoretical limits, based on the IEEE standard. The actual range might be slightly smaller depending on your hardware or operating system.

M is the total number of digits and D is the number of digits following the decimal point. If M and D are omitted, values are stored to the limits allowed by the hardware. A single-precision floating-point number is accurate to approximately 7 decimal places.

**UNSIGNED**, if specified, disallows negative values.

Using FLOAT might give you some unexpected problems because all calculations in MariaDB are done with double precision. See [Floating Point Accuracy](#).

For more details on the attributes, see [Numeric Data Type Overview](#).

## 4.1.1.13 DOUBLE

### Syntax

```
DOUBLE[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
DOUBLE PRECISION[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
REAL[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)

### Description

A normal-size (double-precision) floating-point number (see [FLOAT](#) for a single-precision floating-point number).

Allowable values are:

- -1.7976931348623157E+308  
to  
-2.2250738585072014E-308
- 0
- 2.2250738585072014E-308  
to  
1.7976931348623157E+308

These are the theoretical limits, based on the IEEE standard. The actual range might be slightly smaller depending on your hardware or operating system.

M

is the total number of digits and

D

is the number of digits following the decimal point. If

M

and

D

are omitted, values are stored to the limits allowed by the hardware. A double-precision floating-point number is accurate to approximately 15 decimal places.

**UNSIGNED**

, if specified, disallows negative values.

**ZEROFILL**

, if specified, pads the number with zeros, up to the total number of digits specified by

M

REAL and DOUBLE PRECISION are synonyms, unless the REAL\_AS\_FLOAT [SQL mode](#) is enabled, in which case REAL is a synonym for [FLOAT](#) rather than DOUBLE.

See [Floating Point Accuracy](#) for issues when using floating-point numbers.

For more details on the attributes, see [Numeric Data Type Overview](#).

## Examples

```
CREATE TABLE t1 (d DOUBLE(5,0) zerofill);
INSERT INTO t1 VALUES (1),(2),(3),(4);
SELECT * FROM t1;
+---+
| d |
+---+
| 00001 |
| 00002 |
| 00003 |
| 00004 |
+---+
```

### 4.1.1.14 DOUBLE PRECISION

#### Syntax

```
DOUBLE PRECISION[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
REAL[(M,D)] [SIGNED | UNSIGNED | ZEROFILL]
```

#### Description

REAL and DOUBLE PRECISION are synonyms for [DOUBLE](#).

Exception: If the REAL\_AS\_FLOAT [SQL mode](#) is enabled, REAL is a synonym for [FLOAT](#) rather than [DOUBLE](#).

### 4.1.1.15 BIT

#### Syntax

```
BIT[(M)]
```

#### Description

A bit-field type.

M  
indicates the number of bits per value, from  
1  
to  
64  
. The default is  
1  
if  
M  
is omitted.

Bit values can be inserted with

b'value'  
notation, where  
value  
is the bit value in 0's and 1's.

Bit fields are automatically zero-padded from the left to the full length of the bit, so for example in a BIT(4) field, '10' is equivalent to '0010'.

Bits are returned as binary, so to display them, either add 0, or use a function such as [HEX](#), [OCT](#) or [BIN](#) to convert them.

## Examples

```
CREATE TABLE b ( b1 BIT(8) );
```

With `strict_mode` set, the default from MariaDB 10.2.4 :

```
INSERT INTO b VALUES (b'11111111');
INSERT INTO b VALUES (b'01010101');
INSERT INTO b VALUES (b'11111111111111');
ERROR 1406 (22001): Data too long for column 'b1' at row 1

SELECT b1+0, HEX(b1), OCT(b1), BIN(b1) FROM b;
+-----+-----+-----+
| b1+0 | HEX(b1) | OCT(b1) | BIN(b1) |
+-----+-----+-----+
| 255 | FF      | 377    | 11111111 |
| 85  | 55      | 125    | 1010101  |
+-----+-----+-----+
```

With `strict_mode` unset, the default until MariaDB 10.2.3 :

```
INSERT INTO b VALUES (b'11111111'),(b'01010101'),(b'111111111111');
Query OK, 3 rows affected, 1 warning (0.10 sec)
Records: 3  Duplicates: 0  Warnings: 1

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message           |
+-----+-----+
| Warning | 1264 | Out of range value for column 'b1' at row 3 |
+-----+-----+

SELECT b1+0, HEX(b1), OCT(b1), BIN(b1) FROM b;
+-----+-----+-----+
| b1+0 | HEX(b1) | OCT(b1) | BIN(b1) |
+-----+-----+-----+
| 255 | FF      | 377    | 11111111 |
| 85  | 55      | 125    | 1010101  |
| 255 | FF      | 377    | 11111111 |
+-----+-----+-----+
```

## 4.1.1.16 Floating-point Accuracy

Due to their nature, not all floating-point numbers can be stored with exact precision. Hardware architecture, the CPU or even the compiler version and optimization level may affect the precision.

If you are comparing `DOUBLES` or `FLOATS` with numeric decimals, it is not safe to use the `equality` operator.

Sometimes, changing a floating-point number from single-precision (`FLOAT`) to double-precision (`DOUBLE`) will fix the problem.

### Example

f1, f2 and f3 have seemingly identical values across each row, but due to floating point accuracy, the results may be unexpected.

```
CREATE TABLE fpn (id INT, f1 FLOAT, f2 DOUBLE, f3 DECIMAL (10,3));
INSERT INTO fpn VALUES (1,2,2,2),(2,0.1,0.1,0.1);

SELECT * FROM fpn WHERE f1*f1 = f2*f2;
+-----+-----+-----+
| id  | f1   | f2   | f3   |
+-----+-----+-----+
| 1   | 2    | 2    | 2.000 |
+-----+-----+-----+
```

The reason why only one instead of two rows was returned becomes clear when we see how the floating point squares were evaluated.

```

SELECT f1*f1, f2*f2, f3*f3 FROM fpn;
+-----+-----+-----+
| f1*f1 | f2*f2 | f3*f3 |
+-----+-----+-----+
| 4 | 4 | 4.000000 |
| 0.01000000298023226 | 0.0100000000000002 | 0.010000 |
+-----+-----+-----+

```

## 4.1.1.17 INT1

INT1

is a synonym for TINYINT .

```

CREATE TABLE t1 (x INT1);

DESC t1;
+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| x     | tinyint(4) | YES  |      | NULL    |       |
+-----+-----+-----+-----+

```

## 4.1.1.18 INT2

INT2

is a synonym for SMALLINT .

```

CREATE TABLE t1 (x INT2);

DESC t1;
+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| x     | smallint(6) | YES  |      | NULL    |       |
+-----+-----+-----+-----+

```

## 4.1.1.19 INT3

INT3

is a synonym for MEDIUMINT .

```

CREATE TABLE t1 (x INT3);

DESC t1;
+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| x     | mediumint(9) | YES  |      | NULL    |       |
+-----+-----+-----+-----+

```

## 4.1.1.20 INT4

INT4

is a synonym for INT .

```

CREATE TABLE t1 (x INT4);

DESC t1;
+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| x     | int(11)   | YES  |      | NULL    |       |
+-----+-----+-----+-----+

```

## 4.1.1.21 INT8

INT8  
is a synonym for [BIGINT](#).

```
CREATE TABLE t1 (x INT8);

DESC t1;
+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| x     | bigint(20) | YES  |      | NULL    |       |
+-----+-----+-----+-----+
```

## 4.1.2 String Data Types

### 4.1.2.1 String Literals

Strings are sequences of characters and are enclosed with quotes.

The syntax is:

```
[_charset_name]'string' [COLLATE collation_name]
```

For example:

```
'The MariaDB Foundation'
_utf8 'Foundation' COLLATE utf8_unicode_ci;
```

Strings can either be enclosed in single quotes or in double quotes (the same character must be used to both open and close the string).

The ANSI SQL-standard does not permit double quotes for enclosing strings, and although MariaDB does by default, if the MariaDB server has enabled the [ANSI\\_QUOTES\\_SQL\\_MODE](#), double quotes will be treated as being used for [identifiers](#) instead of strings.

Strings that are next to each other are automatically concatenated. For example:

```
'The ' 'MariaDB ' 'Foundation'
```

and

```
'The MariaDB Foundation'
```

are equivalent.

The

\ (backslash character) is used to escape characters (unless the [SQL\\_MODE](#) hasn't been set to [NO\\_BACKSLASH\\_ESCAPES](#)). For example:

```
'MariaDB\'s new features'
```

is not a valid string because of the single quote in the middle of the string, which is treated as if it closes the string, but is actually meant as part of the string, an apostrophe. The backslash character helps in situations like this:

```
'MariaDB\'s new features'
```

is now a valid string, and if displayed, will appear without the backslash.

```
SELECT 'MariaDB\'s new features';
+-----+
| MariaDB's new features |
+-----+
| MariaDB's new features |
+-----+
```

Another way to escape the quoting character is repeating it twice:

```
SELECT 'I''m here', """Double""";
+-----+-----+
| I'm here | "Double" |
+-----+-----+
| I'm here | "Double" |
+-----+-----+
```

## Escape Sequences

There are other escape sequences also. Here is a full list:

Escape sequence	Character
\0	ASCII NUL (0x00).
\'	Single quote ("").
\"	Double quote (""").
\b	Backspace.
\n	Newline, or linefeed.,
\r	Carriage return.
\t	Tab.
\z	ASCII 26 (Control+Z). See note following the table.
\\\	Backslash ("\").
\%	"%" character. See note following the table.
\_	A "_" character. See note following the table.

Escaping the

%  
and  
-

characters can be necessary when using the [LIKE](#) operator, which treats them as special characters.

The ASCII 26 character (

\z

) needs to be escaped when included in a batch file which needs to be executed in Windows. The reason is that ASCII 26, in Windows, is the end of file (EOF).

Backslash (

\

), if not used as an escape character, must always be escaped. When followed by a character that is not in the above table, backslashes will simply be ignored.

## 4.1.2.2 BINARY

## Syntax

BINARY(M)

### Contents

1. [Description](#)
2. [Examples](#)
3. [See Also](#)

## Description

The

BINARY

type is similar to the

CHAR

type, but stores binary byte strings rather than non-binary character strings.

M

represents the column length in bytes.

It contains no character set, and comparison and sorting are based on the numeric value of the bytes.

If the maximum length is exceeded, and [SQL strict mode](#) is not enabled, the extra characters will be dropped with a warning. If strict mode is enabled, an error will occur.

BINARY values are right-padded with

0x00

(the zero byte) to the specified length when inserted. The padding is *not* removed on select, so this needs to be taken into account when sorting and comparing, where all bytes are significant. The zero byte,

0x00

is less than a space for comparison purposes.

## Examples

Inserting too many characters, first with strict mode off, then with it on:

```
CREATE TABLE bins (a BINARY(10));

INSERT INTO bins VALUES('12345678901');
Query OK, 1 row affected, 1 warning (0.04 sec)

SELECT * FROM bins;
+-----+
| a      |
+-----+
| 1234567890 |
+-----+

SET sql_mode='STRICT_ALL_TABLES';

INSERT INTO bins VALUES('12345678901');
ERROR 1406 (22001): Data too long for column 'a' at row 1
```

Sorting is performed with the byte value:

```

TRUNCATE bins;

INSERT INTO bins VALUES('A'),('B'),('a'),('b');

SELECT * FROM bins ORDER BY a;
+-----+
| a    |
+-----+
| A    |
| B    |
| a    |
| b    |
+-----+

```

Using `CAST` to sort as a `CHAR` instead:

```

SELECT * FROM bins ORDER BY CAST(a AS CHAR);
+-----+
| a    |
+-----+
| a    |
| A    |
| b    |
| B    |
+-----+

```

The field is a `BINARY(10)`, so padding of two `\0`'s are inserted, causing comparisons that don't take this into account to fail:

```

TRUNCATE bins;

INSERT INTO bins VALUES('12345678');

SELECT a = '12345678', a = '12345678\0\0' from bins;
+-----+-----+
| a = '12345678' | a = '12345678\0\0' |
+-----+-----+
|          0      |          1      |
+-----+-----+

```

## See Also

- [CHAR](#)
- [Data Type Storage Requirements](#)

### 4.1.2.3 BLOB

## Syntax

`BLOB[(M)]`

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [Indexing](#)
  2. [Oracle Mode](#)
3. [See Also](#)

## Description

A

BLOB  
column with a maximum length of  
65,535  
(  
2

- 1  
) bytes. Each  
BLOB  
value is stored using a two-byte length prefix that indicates the number of bytes in the value.

An optional length

M  
can be given for this type. If this is done, MariaDB creates the column as the smallest  
BLOB  
type large enough to hold values  
M  
bytes long.

BLOBS can also be used to store [dynamic columns](#).

Before [MariaDB 10.2.1](#),

BLOB  
and  
TEXT  
columns could not be assigned a [DEFAULT](#) value. This restriction was lifted in [MariaDB 10.2.1](#).

## Indexing

MariaDB starting with [10.4](#)

From [MariaDB 10.4](#), it is possible to set a [unique index](#) on a column that uses the  
BLOB  
data type. In previous releases this was not possible, as the index would only guarantee the uniqueness of a fixed number of characters.

## Oracle Mode

MariaDB starting with [10.3](#)

In [Oracle mode from MariaDB 10.3](#),

BLOB  
is a synonym for  
LONGBLOB  
.

## See Also

- [BLOB and TEXT Data Types](#)
- [Data Type Storage Requirements](#)
- [Oracle mode from MariaDB 10.3](#)

### 4.1.2.4 BLOB and TEXT Data Types

## Description

A

BLOB  
is a binary large object that can hold a variable amount of data. The four  
BLOB  
types are

- [TINYBLOB](#),
- [BLOB](#),
- [MEDIUMBLOB](#), and
- [LONGBLOB](#).

These differ only in the maximum length of the values they can hold.

The

TEXT  
types are

- [TINYTEXT](#),
- [TEXT](#),
- [MEDIUMTEXT](#), and
- [LONGTEXT](#).

- [JSON](#) (alias for LONGTEXT)

These correspond to the four

BLOB

types and have the same maximum lengths and [storage requirements](#).

MariaDB starting with [10.2.1](#)

Starting from [MariaDB 10.2.1](#),

BLOB

and

TEXT

columns can have a

DEFAULT

value.

MariaDB starting with [10.4.3](#)

From [MariaDB 10.4](#), it is possible to set a [unique index](#) on columns that use the

BLOB

or

TEXT

data types.

## 4.1.2.5 CHAR

This article covers the CHAR data type. See [CHAR Function](#) for the function.

### Syntax

```
[NATIONAL] CHAR[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]
```

### Contents

- [1. Syntax](#)
- [2. Description](#)
- [3. Examples](#)
- [4. NO PAD Collations](#)
- [5. See Also](#)

### Description

A fixed-length string that is always right-padded with spaces to the specified length when stored.

M

represents the column length in characters. The range of

M

is

0

to

255

. If

M

is omitted, the length is

1

.

CHAR(0) columns can contain 2 values: an empty string or NULL. Such columns cannot be part of an index. The [CONNECT](#) storage engine does not support CHAR(0).

**Note:** Trailing spaces are removed when

CHAR

values are retrieved unless the

PAD\_CHAR\_TO\_FULL\_LENGTH

[SQL mode](#) is enabled.

Before [MariaDB 10.2](#), all collations were of type PADSPACE, meaning that CHAR (as well as [VARCHAR](#) and [TEXT](#)) values are compared without regard for trailing spaces. This does not apply to the [LIKE](#) pattern-matching operator, which takes into account trailing spaces.

If a unique index consists of a column where trailing pad characters are stripped or ignored, inserts into that column where values differ only by the number of trailing pad characters will result in a duplicate-key error.

# Examples

Trailing spaces:

```
CREATE TABLE strtest (c CHAR(10));
INSERT INTO strtest VALUES('Maria   ');

SELECT c='Maria',c='Maria   ' FROM strtest;
+-----+-----+
| c='Maria' | c='Maria   ' |
+-----+-----+
|      1    |          1   |
+-----+-----+

SELECT c LIKE 'Maria',c LIKE 'Maria   ' FROM strtest;
+-----+-----+
| c LIKE 'Maria' | c LIKE 'Maria   ' |
+-----+-----+
|          1     |          0   |
+-----+-----+
```

## NO PAD Collations

MariaDB starting with [10.2](#)

NO PAD collations regard trailing spaces as normal characters. You can get a list of all NO PAD collations by querying the [Information Schema Collations table](#), for example:

```
SELECT collation_name FROM information_schema.collations
  WHERE collation_name LIKE "%nopad%";

+-----+
| collation_name           |
+-----+
| big5_chinese_nopad_ci   |
| big5_nopad_bin           |
...

```

## See Also

- [CHAR Function](#)
- [VARCHAR](#)
- [BINARY](#)
- [Data Type Storage Requirements](#)

## 4.1.2.6 CHAR BYTE

### Description

The

`CHAR BYTE`  
data type is an alias for the

`BINARY`

data type. This is a compatibility feature.

## 4.1.2.7 ENUM

### Syntax

```
ENUM('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [NULL and empty values](#)
  2. [Numeric index](#)
3. [Examples](#)
4. [See Also](#)

## Description

An enumeration. A string object that can have only one value, chosen from the list of values 'value1', 'value2', ..., NULL or the special " error value. In theory, an

ENUM

column can have a maximum of 65,535 distinct values; in practice, the real maximum depends on many factors.

ENUM

values are represented internally as integers.

Trailing spaces are automatically stripped from ENUM values on table creation.

ENUMs require relatively little storage space compared to strings, either one or two bytes depending on the number of enumeration values.

## NULL and empty values

An ENUM can also contain NULL and empty values. If the ENUM column is declared to permit NULL values, NULL becomes a valid value, as well as the default value (see below). If strict SQL Mode is not enabled, and an invalid value is inserted into an ENUM, a special empty string, with an index value of zero (see Numeric index, below), is inserted, with a warning. This may be confusing, because the empty string is also a possible value, and the only difference if that is this case its index is not 0. Inserting will fail with an error if strict mode is active.

If a

DEFAULT

clause is missing, the default value will be:

- NULL  
if the column is nullable;
- otherwise, the first value in the enumeration.

## Numeric index

ENUM values are indexed numerically in the order they are defined, and sorting will be performed in this numeric order. We suggest not using ENUM to store numerals, as there is little to no storage space benefit, and it is easy to confuse the enum integer with the enum numeral value by leaving out the quotes.

An ENUM defined as ENUM('apple','orange','pear') would have the following index values:

Index	Value
NULL	NULL
0	"
1	'apple'
2	'orange'
3	'pear'

## Examples

```

CREATE TABLE fruits (
    id INT NOT NULL auto_increment PRIMARY KEY,
    fruit ENUM('apple','orange','pear'),
    bushels INT);

DESCRIBE fruits;
+-----+-----+-----+-----+
| Field | Type            | Null | Key | Default | Extra       |
+-----+-----+-----+-----+
| id    | int(11)          | NO   | PRI | NULL    | auto_increment |
| fruit | enum('apple','orange','pear') | YES  |     | NULL    |             |
| bushels | int(11)          | YES  |     | NULL    |             |
+-----+-----+-----+-----+

INSERT INTO fruits
(fruit,bushels) VALUES
('pear',20),
('apple',100),
('orange',25);

INSERT INTO fruits
(fruit,bushels) VALUES
('avocado',10);
ERROR 1265 (01000): Data truncated for column 'fruit' at row 1

SELECT * FROM fruits;
+-----+-----+
| id | fruit | bushels |
+-----+-----+
| 1  | pear  |      20 |
| 2  | apple  |      100 |
| 3  | orange |      25 |
+-----+-----+

```

Selecting by numeric index:

```

SELECT * FROM fruits WHERE fruit=2;
+-----+-----+
| id | fruit | bushels |
+-----+-----+
| 3  | orange |      25 |
+-----+-----+

```

Sorting is according to the index value:

```

CREATE TABLE enums (a ENUM('2','1'));

INSERT INTO enums VALUES ('1'),('2');

SELECT * FROM enums ORDER BY a ASC;
+-----+
| a   |
+-----+
| 2   |
| 1   |
+-----+

```

It's easy to get confused between returning the enum integer with the stored value, so we don't suggest using ENUM to store numerals. The first example returns the 1st indexed field ('2' has an index value of 1, as it's defined first), while the second example returns the string value '1'.

```

SELECT * FROM enums WHERE a=1;
+-----+
| a   |
+-----+
| 2   |
+-----+

SELECT * FROM enums WHERE a='1';
+-----+
| a   |
+-----+
| 1   |
+-----+

```

## See Also

- [Data Type Storage Requirements](#)

### 4.1.2.9 JSON Data Type

MariaDB starting with [10.2.7](#)

The JSON alias was added in [MariaDB 10.2.7](#). This was done to make it possible to use JSON columns in [statement based replication](#) from MySQL to MariaDB and to make it possible for MariaDB to read [mysqldumps](#) from MySQL.

#### Contents

1. [Examples](#)
2. [Replicating JSON Data Between MySQL and MariaDB](#)
3. [Converting a MySQL TABLE with JSON Fields to MariaDB](#)
4. [Differences Between MySQL JSON Strings and MariaDB JSON Strings](#)
5. [See Also](#)

JSON is an alias for [LONGTEXT](#) introduced for compatibility reasons with MySQL's JSON data type. MariaDB implements this as a LONGTEXT rather, as the JSON data type contradicts the SQL standard, and MariaDB's benchmarks indicate that performance is at least equivalent.

In order to ensure that a valid json document is inserted, the [JSON\\_VALID](#) function can be used as a [CHECK constraint](#). This constraint is automatically included for types using the JSON alias from [MariaDB 10.4.3](#).

## Examples

```
CREATE TABLE t (j JSON);

DESC t;
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| j     | longtext | YES  |      | NULL    |       |
+-----+-----+-----+-----+
```

With validation:

```
CREATE TABLE t2 (
  j JSON
  CHECK (JSON_VALID(j))
);

INSERT INTO t2 VALUES ('invalid');
ERROR 4025 (23000): CONSTRAINT `j` failed for `test`.`t2`

INSERT INTO t2 VALUES ('{"id": 1, "name": "Monty"}');
Query OK, 1 row affected (0.13 sec)
```

## Replicating JSON Data Between MySQL and MariaDB

The JSON type in MySQL stores the JSON object in a compact form, not as [LONGTEXT](#) as in MariaDB. This means that row based replication will not work for JSON types from MySQL to MariaDB.

There are a few different ways to solve this:

- Use statement based replication.
- Change the JSON column to type TEXT in MySQL
- If you must use row-based replication and cannot change the MySQL master from JSON to TEXT, you can try to introduce an intermediate MySQL slave and change the column type from JSON to TEXT on it. Then you replicate from this intermediate slave to MariaDB.

## Converting a MySQL TABLE with JSON Fields to MariaDB

MariaDB can't directly access MySQL's JSON format.

There are a few different ways to move the table to MariaDB:

- Change the JSON column to type TEXT in MySQL. After this, MariaDB can directly use the table without any need for a dump and restore.

- Use mariadb-dump/mysqldump to copy the table .

## Differences Between MySQL JSON Strings and MariaDB JSON Strings

- In MySQL, JSON is an object and is compared according to json values . In MariaDB JSON strings are normal strings and compared as strings. One exception is when using [JSON\\_EXTRACT\(\)](#) in which case strings are unescaped before comparison.

### See Also

- [JSON Functions](#)
- [CONNECT JSON Table Type](#)
- [MDEV-9144](#)

## 4.1.2.10 MEDIUMBLOB

### Syntax

```
MEDIUMBLOB
```

### Description

A [BLOB](#) column with a maximum length of 16,777,215 ( $2^{24} - 1$ ) bytes. Each MEDIUMBLOB value is stored using a three-byte length prefix that indicates the number of bytes in the value.

### See Also

- [BLOB](#)
- [BLOB and TEXT Data Types](#)
- [Data Type Storage Requirements](#)

## 4.1.2.11 MEDIUMTEXT

### Syntax

```
MEDIUMTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

### Description

A [TEXT](#) column with a maximum length of 16,777,215 (

2

$2^{24}$

- 1

) characters. The effective maximum length is less if the value contains multi-byte characters. Each MEDIUMTEXT value is stored using a three-byte length prefix that indicates the number of bytes in the value.

### See Also

- [TEXT](#)
- [BLOB and TEXT Data Types](#)
- [Data Type Storage Requirements](#)

## 4.1.2.12 LONGBLOB

### Syntax

```
LONGBLOB
```

### Description

A [BLOB](#) column with a maximum length of 4,294,967,295 bytes or 4GB ( $2^{32} - 1$ ). The effective maximum length of LONGBLOB columns depends on the configured maximum packet size in the client/server protocol and available memory. Each LONGBLOB value is stored using a four-byte length prefix that indicates the number of bytes in the value.

## Oracle Mode

MariaDB starting with [10.3](#)

In Oracle mode from MariaDB 10.3 ,

BLOB  
is a synonym for  
LONGBLOB  
.

## See Also

- [BLOB](#)
- [BLOB and TEXT Data Types](#)
- [Data Type Storage Requirements](#)
- [Oracle mode from MariaDB 10.3](#)

## 4.1.2.13 LONG and LONG VARCHAR

LONG  
and  
LONG VARCHAR  
are synonyms for [MEDIUMTEXT](#) .

```
CREATE TABLE t1 (a LONG, b LONG VARCHAR);
```

```
DESC t1;
```

Field	Type	Null	Key	Default	Extra
a	mediumtext	YES		NULL	
b	mediumtext	YES		NULL	

## 4.1.2.14 LONGTEXT

### Syntax

```
LONGTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Oracle Mode](#)
4. [See Also](#)

### Description

A [TEXT](#) column with a maximum length of 4,294,967,295 or 4GB (

2

32

- 1

) characters. The effective maximum length is less if the value contains multi-byte characters. The effective maximum length of LONGTEXT columns also depends on the configured maximum packet size in the client/server protocol and available memory. Each LONGTEXT value is stored using a four-byte length prefix that indicates the number of bytes in the value.

From [MariaDB 10.2.7](#) , JSON is an alias for LONGTEXT. See [JSON Data Type](#) for details.

# Oracle Mode

MariaDB starting with 10.3

In Oracle mode from MariaDB 10.3 ,

CLOB  
is a synonym for  
LONGTEXT

## See Also

- [TEXT](#)
- [BLOB and TEXT Data Types](#)
- [Data Type Storage Requirements](#)
- [JSON Data Type](#)
- [Oracle mode from MariaDB 10.3](#)

## 4.1.2.15 ROW

MariaDB starting with 10.3.0

The

ROW  
data type was introduced in MariaDB 10.3.0 .

## Syntax

```
ROW (<field name> <data type> [{, <field name> <data type>}... ])
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Features](#)
  1. [ROW fields as normal variables](#)
  2. [ROW type variables as FETCH targets](#)
  3. [ROW type variables as SELECT...INTO targets](#)
4. [Features not implemented](#)
5. [Examples](#)
  1. [Declaring a ROW in a stored procedure](#)
  2. [FETCH Examples](#)
  3. [SELECT...INTO Examples](#)
6. [See Also](#)

## Description

ROW

is a data type for [stored procedure](#) variables.

## Features

### ROW fields as normal variables

ROW

fields (members) act as normal variables, and are able to appear in all query parts where a stored procedure variable is allowed:

- Assignment is using the  
:=  
operator and the [SET](#) command:

```
a.x:= 10;  
a.x:= b.x;  
SET a.x= 10, a.y=20, a.z= b.z;
```

- Passing to functions and operators:

```
SELECT f1(rec.a), rec.a<10;
```

- Clauses (select list, WHERE, HAVING, LIMIT, etc...):

```
SELECT var.a, t1.b FROM t1 WHERE t1.b=var.b LIMIT var.c;
```

- INSERT  
    values:

```
INSERT INTO t1 VALUES (rec.a, rec.b, rec.c);
```

- SELECT .. INTO  
    targets

```
SELECT a,b INTO rec.a, rec.b FROM t1 WHERE t1.id=10;
```

- Dynamic SQL out parameters ( [EXECUTE](#) and [EXECUTE IMMEDIATE](#) )

```
EXECUTE IMMEDIATE 'CALL proc_with_out_param(?)' USING rec.a;
```

## ROW type variables as FETCH targets

ROW  
type variables are allowed as [FETCH](#) targets:

```
FETCH cur INTO rec;
```

where

cur  
is a  
CURSOR  
and  
rec  
is a  
ROW  
type stored procedure variable.

Note, currently an attempt to use

    FETCH  
    for a  
    ROW  
    type variable returns this error:

```
ERROR 1328 (HY000): Incorrect number of FETCH variables
```

    FETCH  
    from a cursor  
    cur  
    into a  
    ROW  
    variable  
    rec  
    works as follows:

- The number of fields in  
        cur

must match the number of fields in  
rec  
. Otherwise, an error is reported.

- Assignment is done from left to right. The first cursor field is assigned to the first variable field, the second cursor field is assigned to the second variable field, etc.
- Field names in  
rec  
are not important and can differ from field names in  
cur

See [FETCH Examples](#) (below) for examples of using this with

```
sql_mode=ORACLE
and
sql_mode=DEFAULT
```

## ROW type variables as

```
SELECT...INTO
targets
```

ROW  
type variables are allowed as  
SELECT..INTO  
targets with some differences depending on which  
sql\_mode  
is in use.

- When using

```
sql_mode=ORACLE
,
table%ROWTYPE
and
cursor%ROWTYPE
variables can be used as
SELECT...INTO
targets.
```

- Using multiple

```
ROW
variables in the
SELECT..INTO
list will report an error.
```

- Using

```
ROW
variables with a different column count than in the
SELECT..INTO
list will report an error.
```

See [SELECT...INTO Examples](#) (below) for examples of using this with

```
sql_mode=ORACLE
and
sql_mode=DEFAULT
```

## Features not implemented

The following features are planned, but not implemented yet:

- Returning a ROW type expression from a stored function (see [MDEV-12252](#) ). This will need some grammar change to support field names after parentheses:

```
SELECT f1().x FROM DUAL;
```

- Returning a ROW type expression from a built-in hybrid type function, such as  
CASE  
,  
IF  
, etc.

- ROW of ROWs

## Examples

### Declaring a ROW in a stored procedure

```
DELIMITER $$  
CREATE PROCEDURE p1()  
BEGIN  
    DECLARE r ROW (c1 INT, c2 VARCHAR(10));  
    SET r.c1= 10;  
    SET r.c2= 'test';  
    INSERT INTO t1 VALUES (r.c1, r.c2);  
END;  
$$  
DELIMITER ;  
CALL p1();
```

### FETCH Examples

A complete

```
FETCH  
example for  
sql_mode=ORACLE  
:  
  
DROP TABLE IF EXISTS t1;  
CREATE TABLE t1 (a INT, b VARCHAR(32));  
INSERT INTO t1 VALUES (10,'b10');  
INSERT INTO t1 VALUES (20,'b20');  
INSERT INTO t1 VALUES (30,'b30');  
  
SET sql_mode=oracle;  
DROP PROCEDURE IF EXISTS p1;  
DELIMITER $$  
CREATE PROCEDURE p1 AS  
    rec ROW(a INT, b VARCHAR(32));  
    CURSOR c IS SELECT a,b FROM t1;  
BEGIN  
    OPEN c;  
    LOOP  
        FETCH c INTO rec;  
        EXIT WHEN c%NOTFOUND;  
        SELECT ('rec=(\' || rec.a ||','|| rec.b||\')');  
    END LOOP;  
    CLOSE c;  
END;  
$$  
DELIMITER ;  
CALL p1();
```

A complete

```
FETCH  
example for  
sql_mode=DEFAULT  
:
```

```

DROP TABLE IF EXISTS t1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
INSERT INTO t1 VALUES (20,'b20');
INSERT INTO t1 VALUES (30,'b30');

SET sql_mode=DEFAULT;
DROP PROCEDURE IF EXISTS p1;
DELIMITER $$ 
CREATE PROCEDURE p1()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE rec ROW(a INT, b VARCHAR(32));
    DECLARE c CURSOR FOR SELECT a,b FROM t1;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    OPEN c;
    read_loop:
    LOOP
        FETCH c INTO rec;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT CONCAT('rec=(',rec.a,',',rec.b,')');
    END LOOP;
    CLOSE c;
END;
$$
DELIMITER ;
CALL p1();

```

## SELECT...INTO Examples

A

```

SELECT...INTO
example for
sql_mode=DEFAULT
:

SET sql_mode=DEFAULT;
DROP TABLE IF EXISTS t1;
DROP PROCEDURE IF EXISTS p1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
DELIMITER $$ 
CREATE PROCEDURE p1()
BEGIN
    DECLARE rec1 ROW(a INT, b VARCHAR(32));
    SELECT * FROM t1 INTO rec1;
    SELECT rec1.a, rec1.b;
END;
$$
DELIMITER ;
CALL p1();

```

The above example returns:

rec1.a	rec1.b
10	b10

A

```

SELECT...INTO
example for
sql_mode=ORACLE
:

```

```

SET sql_mode=ORACLE;
DROP TABLE IF EXISTS t1;
DROP PROCEDURE IF EXISTS p1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
DELIMITER $$ 
CREATE PROCEDURE p1 AS
    rec1 ROW(a INT, b VARCHAR(32));
BEGIN
    SELECT * FROM t1 INTO rec1;
    SELECT rec1.a, rec1.b;
END;
$$
DELIMITER ;
CALL p1();

```

The above example returns:

rec1.a	rec1.b
10	b10

An example for

```

sql_mode=ORACLE
using
table%ROWTYPE
variables as
SELECT..INTO
targets:

```

```

SET sql_mode=ORACLE;
DROP TABLE IF EXISTS t1;
DROP PROCEDURE IF EXISTS p1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
DELIMITER $$ 
CREATE PROCEDURE p1 AS
    rec1 t1%ROWTYPE;
BEGIN
    SELECT * FROM t1 INTO rec1;
    SELECT rec1.a, rec1.b;
END;
$$
DELIMITER ;
CALL p1();

```

The above example returns:

rec1.a	rec1.b
10	b10

An example for

```

sql_mode=ORACLE
using
cursor%ROWTYPE
variables as
SELECT..INTO
targets:

```

```

SET sql_mode=ORACLE;
DROP TABLE IF EXISTS t1;
DROP PROCEDURE IF EXISTS p1;
CREATE TABLE t1 (a INT, b VARCHAR(32));
INSERT INTO t1 VALUES (10,'b10');
DELIMITER $$ 
CREATE PROCEDURE p1 AS
    CURSOR cur1 IS SELECT * FROM t1;
    rec1 cur1%ROWTYPE;
BEGIN
    SELECT * FROM t1 INTO rec1;
    SELECT rec1.a, rec1.b;
END;
$$
DELIMITER ;
CALL p1();

```

The above example returns:

rec1.a	rec1.b
10	b10

## See Also

- [STORED PROCEDURES](#)
- [DECLARE Variable](#)

## 4.1.2.16 TEXT

### Syntax

```
TEXT[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [Indexing](#)
5. [Difference between VARCHAR and TEXT](#)
  1. [For Storage Engine Developers](#)
6. [See Also](#)

### Description

A

```

TEXT
column with a maximum length of
65,535
(
2

```

16

```

- 1
) characters. The effective maximum length is less if the value contains multi-byte characters. Each
TEXT
value is stored using a two-byte length prefix that indicates the number of bytes in the value. If you need a bigger storage, consider using
MEDIUMTEXT instead.

```

An optional length

```

M
can be given for this type. If this is done, MariaDB creates the column as the smallest
TEXT
type large enough to hold values
M

```

characters long.

Before MariaDB 10.2 , all MariaDB [collations](#) were of type PADSPACE, meaning that TEXT (as well as VARCHAR and CHAR values) are compared without regard for trailing spaces. This does not apply to the LIKE pattern-matching operator, which takes into account trailing spaces.

Before MariaDB 10.2.1 ,

BLOB

and

TEXT

columns could not be assigned a [DEFAULT](#) value. This restriction was lifted in MariaDB 10.2.1 .

## Examples

Trailing spaces:

```
CREATE TABLE strtest (d TEXT(10));
INSERT INTO strtest VALUES('Maria   ');

SELECT d='Maria',d='Maria   ' FROM strtest;
+-----+-----+
| d='Maria' | d='Maria   ' |
+-----+-----+
|      1 |          1 |
+-----+-----+

SELECT d LIKE 'Maria',d LIKE 'Maria   ' FROM strtest;
+-----+-----+
| d LIKE 'Maria' | d LIKE 'Maria   ' |
+-----+-----+
|          0 |          1 |
+-----+-----+
```

## Indexing

TEXT

columns can only be indexed over a specified length. This means that they cannot be used as the [primary key](#) of a table norm until MariaDB 10.4 , can a [unique index](#) be created on them.

MariaDB starting with 10.4

Starting with MariaDB 10.4 , a unique index can be created on a

TEXT

column.

Internally, this uses hash indexing to quickly check the values and if a hash collision is found, the actual stored values are compared in order to retain the uniqueness.

## Difference between VARCHAR and TEXT

- VARCHAR columns can be fully indexed.

TEXT

columns can only be indexed over a specified length.

- Using TEXT or BLOB in a SELECT query that uses temporary tables for storing intermediate results will force the temporary table to be disk based (using the [Aria storage engine](#) instead of the [memory storage engine](#) , which is a bit slower. This is not that bad as the [Aria storage engine](#) caches the rows in memory. To get the benefit of this, one should ensure that the [aria\\_pagecache\\_buffer\\_size](#) variable is big enough to hold most of the row and index data for temporary tables.

## For Storage Engine Developers

- Internally the full length of the VARCHAR column is allocated inside each TABLE objects record[] structure. As there are three such buffers, each open table will allocate 3 times max-length-to-store-varchar bytes of memory.

- 

TEXT

and

BLOB

columns are stored with a pointer (4 or 8 bytes) + a 1-4 bytes length. The

TEXT

data is only stored once. This means that internally

TEXT

uses less memory for each open table but instead has the additional overhead that each

## TEXT

object needs to be allocated and freed for each row access (with some caching in between).

## See Also

- [BLOB and TEXT Data Types](#)
- [MEDIUMTEXT](#)
- [Data Type Storage Requirements](#)

## 4.1.2.17 TINYBLOB

### Syntax

```
TINYBLOB
```

### Description

A [BLOB](#) column with a maximum length of 255 ( $2^8 - 1$ ) bytes. Each TINYBLOB value is stored using a one-byte length prefix that indicates the number of bytes in the value.

## See Also

- [BLOB](#)
- [BLOB and TEXT Data Types](#)
- [Data Type Storage Requirements](#)

## 4.1.2.18 TINYTEXT

### Syntax

```
TINYTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

### Description

A [TEXT](#) column with a maximum length of 255 (

2

8

- 1

) characters. The effective maximum length is less if the value contains multi-byte characters. Each TINYTEXT value is stored using a one-byte length prefix that indicates the number of bytes in the value.

## See Also

- [TEXT](#)
- [BLOB and TEXT Data Types](#)
- [Data Type Storage Requirements](#)

## 4.1.2.19 VARBINARY

### Syntax

```
VARBINARY(M)
```

## Contents

1. [Syntax](#)
2. [Description](#)
  1. [Oracle Mode](#)
3. [Examples](#)
4. [See Also](#)

## Description

The VARBINARY type is similar to the VARCHAR type, but stores binary byte strings rather than non-binary character strings.

M

represents the maximum column length in bytes.

It contains no [character set](#), and comparison and sorting are based on the numeric value of the bytes.

If the maximum length is exceeded, and [SQL strict mode](#) is not enabled, the extra characters will be dropped with a warning. If strict mode is enabled, an error will occur.

Unlike [BINARY](#) values, VARBINARYs are not right-padded when inserting.

## Oracle Mode

MariaDB starting with 10.3

In [Oracle mode from MariaDB 10.3](#),

RAW  
is a synonym for  
VARBINARY

## Examples

Inserting too many characters, first with strict mode off, then with it on:

```
CREATE TABLE varbins (a VARBINARY(10));

INSERT INTO varbins VALUES('12345678901');
Query OK, 1 row affected, 1 warning (0.04 sec)

SELECT * FROM varbins;
+-----+
| a      |
+-----+
| 1234567890 |
+-----+

SET sql_mode='STRICT_ALL_TABLES';

INSERT INTO varbins VALUES('12345678901');
ERROR 1406 (22001): Data too long for column 'a' at row 1
```

Sorting is performed with the byte value:

```
TRUNCATE varbins;

INSERT INTO varbins VALUES('A'),('B'),('a'),('b');

SELECT * FROM varbins ORDER BY a;
+-----+
| a    |
+-----+
| A    |
| B    |
| a    |
| b    |
+-----+
```

Using [CAST](#) to sort as a [CHAR](#) instead:

```
SELECT * FROM varbins ORDER BY CAST(a AS CHAR);
+----+
| a |
+----+
| a |
| A |
| b |
| B |
+----+
```

## See Also

- [VARCHAR](#)
- [Data Type Storage Requirements](#)
- [Oracle mode from MariaDB 10.3](#)

## 4.1.2.20 VARCHAR

### Syntax

```
[NATIONAL] VARCHAR(M) [CHARACTER SET charset_name] [COLLATE collation_name]
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [Truncation](#)
5. [Difference Between VARCHAR and TEXT](#)
6. [Oracle Mode](#)
  1. [For Storage Engine Developers](#)
7. [See Also](#)

### Description

A variable-length string. M represents the maximum column length in characters. The range of M is 0 to 65,532. The effective maximum length of a VARCHAR is subject to the maximum row size and the character set used. For example, utf8 characters can require up to three bytes per character, so a VARCHAR column that uses the utf8 character set can be declared to be a maximum of 21,844 characters.

**Note:** For the [ColumnStore](#) engine, M represents the maximum column length in bytes.

MariaDB stores VARCHAR values as a one-byte or two-byte length prefix plus data. The length prefix indicates the number of bytes in the value. A VARCHAR column uses one length byte if values require no more than 255 bytes, two length bytes if values may require more than 255 bytes.

MariaDB follows the standard SQL specification, and does not remove trailing spaces from VARCHAR values.

VARCHAR(0) columns can contain 2 values: an empty string or NULL. Such columns cannot be part of an index. The [CONNECT](#) storage engine does not support VARCHAR(0).

VARCHAR is shorthand for CHARACTER VARYING. NATIONAL VARCHAR is the standard SQL way to define that a VARCHAR column should use some predefined character set. MariaDB uses utf8 as this predefined character set, as does MySQL 4.1 and up. NVARCHAR is shorthand for NATIONAL VARCHAR.

Before [MariaDB 10.2](#), all MariaDB [collations](#) were of type

PADSPACE

, meaning that VARCHAR (as well as [CHAR](#) and [TEXT](#) values) are compared without regard for trailing spaces. This does not apply to the [LIKE](#) pattern-matching operator, which takes into account trailing spaces. From [MariaDB 10.2](#), a number of [NO PAD collations](#) are available.

If a unique index consists of a column where trailing pad characters are stripped or ignored, inserts into that column where values differ only by the number of trailing pad characters will result in a duplicate-key error.

### Examples

The following are equivalent:

```
VARCHAR(30) CHARACTER SET utf8
NATIONAL VARCHAR(30)
NVARCHAR(30)
NCHAR VARCHAR(30)
NATIONAL CHARACTER VARYING(30)
NATIONAL CHAR VARYING(30)
```

Trailing spaces:

```
CREATE TABLE strtest (v VARCHAR(10));
INSERT INTO strtest VALUES('Maria   ');

SELECT v='Maria',v='Maria   ' FROM strtest;
+-----+
| v='Maria' | v='Maria   ' |
+-----+
|      1    |          1   |
+-----+

SELECT v LIKE 'Maria',v LIKE 'Maria   ' FROM strtest;
+-----+
| v LIKE 'Maria' | v LIKE 'Maria   ' |
+-----+
|        0       |          1   |
+-----+
```

## Truncation

- Depending on whether or not `strict sql mode` is set, you will either get a warning or an error if you try to insert a string that is too long into a `VARCHAR` column. If the extra characters are spaces, the spaces that can't fit will be removed and you will always get a warning, regardless of the `sql mode` setting.

## Difference Between VARCHAR and TEXT

- `VARCHAR` columns can be fully indexed. `TEXT` columns can only be indexed over a specified length.
- Using `TEXT` or `BLOB` in a `SELECT` query that uses temporary tables for storing intermediate results will force the temporary table to be disk based (using the `Aria storage engine` instead of the `memory storage engine`, which is a bit slower. This is not that bad as the `Aria storage engine` caches the rows in memory. To get the benefit of this, one should ensure that the `aria_pagecache_buffer_size` variable is big enough to hold most of the row and index data for temporary tables.

## Oracle Mode

MariaDB starting with 10.3

In Oracle mode from MariaDB 10.3 ,

`VARCHAR2`

is a synonym.

## For Storage Engine Developers

- Internally the full length of the `VARCHAR` column is allocated inside each TABLE objects record[] structure. As there are three such buffers, each open table will allocate 3 times max-length-to-store-varchar bytes of memory.
- `TEXT` and `BLOB` columns are stored with a pointer (4 or 8 bytes) + a 1-4 bytes length. The `TEXT` data is only stored once. This means that internally

`TEXT`

uses less memory for each open table but instead has the additional overhead that each

`TEXT`

object needs to be allocated and freed for each row access (with some caching in between).

## See Also

- [VARBINARY](#)
- [TEXT](#)
- [CHAR](#)
- [Character Sets and Collations](#)
- [Data Type Storage Requirements](#)
- [Oracle mode from MariaDB 10.3](#)

## 4.1.2.21 SET Data Type

### Syntax

```
SET('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]
```

# Description

A set. A string object that can have zero or more values, each of which must be chosen from the list of values 'value1', 'value2', ... A SET column can have a maximum of 64 members. SET values are represented internally as integers.

SET values cannot contain commas.

If a SET contains duplicate values, an error will be returned if [strict mode](#) is enabled, or a warning if strict mode is not enabled.

## See Also

- [Character Sets and Collations](#)
- [Data Type Storage Requirements](#)

## 4.1.2.22 UUID Data Type

MariaDB starting with [10.7.0](#)

The UUID data type was added in a [MariaDB 10.7.0](#) preview.

## Syntax

UUID

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [Retrieval](#)
  2. [Casting](#)
3. [Examples](#)
4. [See Also](#)

## Description

The

UUID

data type is intended for the storage of 128-bit UUID (Universally Unique Identifier) data. See the [UUID function](#) page for more details on UUIDs themselves.

## Retrieval

Data retrieved by this data type is in the string representation defined in [RFC4122](#).

## Casting

String literals of hexadecimal characters and [CHAR](#) / [VARCHAR](#) / [TEXT](#) can be cast to the UUID data type. Likewise [hexadecimal literals](#) , [binary-literals](#) , and [BINARY](#) / [VARBINARY](#) / [BLOB](#) types can also be cast to UUID.

The data type will not accept a short UUID generated with the [UUID\\_SHORT](#) function, but will accept a value without the

character generated by the [SYS\\_GUID](#) function (or inserted directly). Hyphens can be partially omitted as well, or included after any group of two digits.

The type does not accept UUIDs in braces, permitted by some implementations.

## Examples

```
CREATE TABLE t1 (id UUID);
```

Directly Inserting via [string literals](#) :

```
INSERT INTO t1 VALUES ('123e4567-e89b-12d3-a456-426655440000');
```

Directly Inserting via [hexadecimal literals](#) :

```
INSERT INTO t1 VALUES (x'fffffffffffffffffffe');
```

Generating and inserting via the [UUID function](#) .

```
INSERT INTO t1 VALUES (UUID());
```

Retrieval:

```
SELECT * FROM t1;
+-----+
| id |
+-----+
| 123e4567-e89b-12d3-a456-426655440000 |
| ffffffff-ffff-ffff-ffff-fffffffffe |
| 93aac041-1a14-11ec-ab4e-f859713e4be4 |
+-----+
```

The [UUID\\_SHORT](#) function does not generate valid full-length UUID:

```
INSERT INTO t1 VALUES (UUID_SHORT());
ERROR 1292 (22007): Incorrect uuid value: '99440417627439104'
for column `test`.`t1`.`id` at row 1
```

Accepting a value without the

character, either directly or generated by the [SYS\\_GUID](#) function:

```
INSERT INTO t1 VALUES (SYS_GUID());
SELECT * FROM t1;
+-----+
| id |
+-----+
| 123e4567-e89b-12d3-a456-426655440000 |
| ffffffff-ffff-ffff-ffff-fffffffffe |
| 93aac041-1a14-11ec-ab4e-f859713e4be4 |
| ea0368d3-1a14-11ec-ab4e-f859713e4be4 |
+-----+
SELECT SYS_GUID();
+-----+
| SYS_GUID() |
+-----+
| ff5b6bcc1a1411ecab4ef859713e4be4 |
+-----+
INSERT INTO t1 VALUES ('ff5b6bcc1a1411ecab4ef859713e4be4');
SELECT * FROM t1;
+-----+
| id |
+-----+
| 123e4567-e89b-12d3-a456-426655440000 |
| ffffffff-ffff-ffff-ffff-fffffffffe |
| 93aac041-1a14-11ec-ab4e-f859713e4be4 |
| ea0368d3-1a14-11ec-ab4e-f859713e4be4 |
| ff5b6bcc-1a14-11ec-ab4e-f859713e4be4 |
+-----+
```

Valid and invalid hyphen and brace usage:

```

TRUNCATE t1;

INSERT INTO t1 VALUES ('f8aa-ed66-1a1b-11ec-ab4e-f859-713e-4be4');

INSERT INTO t1 VALUES ('1b80667f1a1c-11ecab4ef859713e4be4');

INSERT INTO t1 VALUES ('2fd6c945-1a-1c-11ec-ab4e-f859713e4be4');

INSERT INTO t1 VALUES ('49-c9-f9-59-1a-1c-11ec-ab4e-f859713e4be4');

INSERT INTO t1 VALUES ('57-96-da-c1-1a-1c-11-ec-ab-4e-f8-59-71-3e-4b-e4');

INSERT INTO t1 VALUES ('6-eb74f8f-1a1c-11ec-ab4e-f859713e4be4');

INSERT INTO t1 VALUES ('{29bad136-1a1d-11ec-ab4e-f859713e4be4}');
ERROR 1292 (22007): Incorrect uuid value: '{29bad136-1a1d-11ec-ab4e-f859713e4be4}'
  for column `test`.`t1`.`id` at row 1

SELECT * FROM t1;
+-----+
| id   |
+-----+
| f8aaed66-1a1b-11ec-ab4e-f859713e4be4 |
| 1b80667f-1a1c-11ec-ab4e-f859713e4be4 |
| 2fd6c945-1a1c-11ec-ab4e-f859713e4be4 |
| 49c9f959-1a1c-11ec-ab4e-f859713e4be4 |
| 5796dac1-1a1c-11ec-ab4e-f859713e4be4 |
| 6eb74f8f-1a1c-11ec-ab4e-f859713e4be4 |
+-----+

```

## See Also

- [10.7 preview feature: UUID Data Type](#) (mariadb.org blog post)
- [UUID function](#)
- [UUID\\_SHORT function](#)
- [SYS\\_GUID](#) - UUID without the

character for Oracle compatibility

### 4.1.2.23 Data Type Storage Requirements

#### Contents

1. [Numeric Data Types](#)
  1. [Decimal](#)
2. [String Data Types](#)
  1. [Examples](#)
3. [Date and Time Data Types](#)
  1. [Microseconds](#)
    1. [MySQL 5.6+ and MariaDB 10.1+](#)
    2. [MariaDB 5.3 - MariaDB 10.0](#)

The following tables indicate the approximate data storage requirements for each data type.

## Numeric Data Types

Data Type	Storage Requirement
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
BIGINT	8 bytes
FLOAT (p)	4 bytes if p <= 24, otherwise 8 bytes
DOUBLE	8 bytes
DECIMAL	See table below
BIT (M)	(M+7)/8 bytes

Note that MEDIUMINT columns will require 4 bytes in memory (for example, in InnoDB buffer pool).

## Decimal

Decimals are stored using a binary format, with the integer and the fraction stored separately. Each nine-digit multiple requires 4 bytes, followed by a number of bytes for whatever remains, as follows:

Remaining digits	Storage Requirement
0	0 bytes
1	1 byte
2	1 byte
3	2 bytes
4	2 bytes
5	3 bytes
6	3 bytes
7	4 bytes
8	4 bytes

## String Data Types

In the descriptions below,

M

is the declared column length (in characters or in bytes), while

len

is the actual length in bytes of the value.

Data Type	Storage Requirement
ENUM	1 byte for up to 255 enum values, 2 bytes for 256 to 65,535 enum values
CHAR(M)	M × w bytes, where w is the number of bytes required for the maximum-length character in the character set
BINARY(M)	M bytes
VARCHAR(M) , VARBINARY(M)	len + 1 bytes if column is 0 – 255 bytes, len + 2 bytes if column may require more than 255 bytes
TINYBLOB , TINYTEXT	len + 1 bytes
BLOB , TEXT	len + 2 bytes
MEDIUMBLOB , MEDIUMTEXT	len + 3 bytes
LONGBLOB , LONGTEXT	len + 4 bytes
SET	Given M members of the set, (M+7)/8 bytes, rounded up to 1, 2, 3, 4, or 8 bytes
INET6	16 bytes
UUID	16 bytes

In some [character sets](#), not all characters use the same number of bytes. utf8 encodes characters with one to three bytes per character, while utf8mb4 requires one to four bytes per character.

When using field the COMPRESSED attribute, 1 byte is reserved for metadata. For example, VARCHAR(255) will use +2 bytes instead of +1.

## Examples

Assuming a single-byte character-set:

Value	CHAR(2)	Storage Required	VARCHAR(2)	Storage Required
"	''	2 bytes	"	1 byte
'1'	'1 '	2 bytes	'1'	2 bytes
'12'	'12'	2 bytes	'12'	3 bytes

## Date and Time Data Types

Data Type	Storage Requirement
-----------	---------------------

DATE	3 bytes
TIME	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
YEAR	1 byte

## Microseconds

MariaDB 5.3 and MySQL 5.6 introduced [microseconds](#). The underlying storage implementations were different, but from MariaDB 10.1, MariaDB defaults to the MySQL format (by means of the `mysql56_temporal_format` variable). Microseconds have the following additional storage requirements:

MySQL 5.6+ and MariaDB 10.1 +

Precision	Storage Requirement
0	0 bytes
1,2	1 byte
3,4	2 bytes
5,6	3 bytes

MariaDB 5.3 - MariaDB 10.0

Precision	Storage Requirement
0	0 bytes
1,2	1 byte
3,4,5	2 bytes
6	3 bytes

## 4.1.2.24 Supported Character Sets and Collations

### Contents

1. [Character Sets](#)
2. [Collations](#)
3. [NO PAD collations](#)
4. [Changes](#)
5. [See Also](#)

## Character Sets

MariaDB 10.2 supports the following character sets:

Note that the Mroonga Storage Engine only supports a limited number of character sets. See [Mroonga available character sets](#).

Charset	Description	Default collation	Maxlen
armSCII8	ARMSCII-8 Armenian	armSCII8_general_ci	1
ascii	US ASCII	ascii_general_ci	1
big5	Big5 Traditional Chinese	big5_chinese_ci	2
binary	Binary pseudo charset	binary	1
cp1250	Windows Central European	cp1250_general_ci	1
cp1251	Windows Cyrillic	cp1251_general_ci	1
cp1256	Windows Arabic	cp1256_general_ci	1
cp1257	Windows Baltic	cp1257_general_ci	1
cp850	DOS West European	cp850_general_ci	1
cp852	DOS Central European	cp852_general_ci	1
cp866	DOS Russian	cp866_general_ci	1
cp932	SJIS for Windows Japanese	cp932_japanese_ci	2

dec8	DEC West European	dec8_swedish_ci	1
eucjpm	UJIS for Windows Japanese	eucjpm_japanese_ci	3
euckr	EUC-KR Korean	euckr_korean_ci	2
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
gbk	GBK Simplified Chinese	gbk_chinese_ci	2
geostd8	GEOSTD8 Georgian	geostd8_general_ci	1
greek	ISO 8859-7 Greek	greek_general_ci	1
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
hp8	HP West European	hp8_english_ci	1
keybcs2	DOS Kamenicky Czech-Slovak	keybcs2_general_ci	1
koi8r	KOI8-R Relcom Russian	koi8r_general_ci	1
koi8u	KOI8-U Ukrainian	koi8u_general_ci	1
latin1	cp1252 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Central European	latin2_general_ci	1
latin5	ISO 8859-9 Turkish	latin5_turkish_ci	1
latin7	ISO 8859-13 Baltic	latin7_general_ci	1
macce	Mac Central European	macce_general_ci	1
macroman	Mac West European	macroman_general_ci	1
sjis	Shift-JIS Japanese	sjis_japanese_ci	2
swe7	7bit Swedish	swe7_swedish_ci	1
tis620	TIS620 Thai	tis620_thai_ci	1
ucs2	UCS-2 Unicode	ucs2_general_ci	2
ujis	EUC-JP Japanese	ujis_japanese_ci	3
utf8	UTF-8 Unicode	utf8_general_ci	3/4 (see <a href="#">OLD_MODE</a> )
utf16	UTF-16 Unicode	utf16_general_ci	4
utf16le	UTF-16LE Unicode	utf16le_general_ci	4
utf32	UTF-32 Unicode	utf32_general_ci	4
utf8mb3	UTF-8 Unicode	utf8mb3_general_ci	3
utf8mb4	UTF-8 Unicode	utf8mb4_general_ci	4

You can see which character sets are available in a particular version by running the `SHOW CHARACTER SET` statement or by querying the [Information Schema CHARACTER\\_SETS Table](#).

## Collations

MariaDB supports the following collations:

SHOW COLLATION;					
Collation	Charset	Id	Default	Compiled	Sortlen
big5_chinese_ci	big5	1	Yes	Yes	1
big5_bin	big5	84		Yes	1
big5_chinese_nopad_ci	big5	1025		Yes	1
big5_nopad_bin	big5	1108		Yes	1
dec8_swedish_ci	dec8	3	Yes	Yes	1
dec8_bin	dec8	69		Yes	1
dec8_swedish_nopad_ci	dec8	1027		Yes	1
dec8_nopad_bin	dec8	1093		Yes	1
cp850_general_ci	cp850	4	Yes	Yes	1
cp850_bin	cp850	80		Yes	1
cp850_general_nopad_ci	cp850	1028		Yes	1
cp850_nopad_bin	cp850	1104		Yes	1
hp8_english_ci	hp8	6	Yes	Yes	1
hp8_bin	hp8	72		Yes	1
hp8_english_nopad_ci	hp8	1030		Yes	1

hp8_nopad_bin	hp8	1096	Yes	1
koi8r_general_ci	koi8r	7	Yes	1
koi8r_bin	koi8r	74	Yes	1
koi8r_general_nopad_ci	koi8r	1031	Yes	1
koi8r_nopad_bin	koi8r	1098	Yes	1
latin1_german1_ci	latin1	5	Yes	1
latin1_swedish_ci	latin1	8	Yes	1
latin1_danish_ci	latin1	15	Yes	1
latin1_german2_ci	latin1	31	Yes	2
latin1_bin	latin1	47	Yes	1
latin1_general_ci	latin1	48	Yes	1
latin1_general_cs	latin1	49	Yes	1
latin1_spanish_ci	latin1	94	Yes	1
latin1_swedish_nopad_ci	latin1	1032	Yes	1
latin1_nopad_bin	latin1	1071	Yes	1
latin2_czech_cs	latin2	2	Yes	4
latin2_general_ci	latin2	9	Yes	1
latin2_hungarian_ci	latin2	21	Yes	1
latin2_croatian_ci	latin2	27	Yes	1
latin2_bin	latin2	77	Yes	1
latin2_general_nopad_ci	latin2	1033	Yes	1
latin2_nopad_bin	latin2	1101	Yes	1
swe7_swedish_ci	swe7	10	Yes	1
swe7_bin	swe7	82	Yes	1
swe7_swedish_nopad_ci	swe7	1034	Yes	1
swe7_nopad_bin	swe7	1106	Yes	1
ascii_general_ci	ascii	11	Yes	1
ascii_bin	ascii	65	Yes	1
ascii_general_nopad_ci	ascii	1035	Yes	1
ascii_nopad_bin	ascii	1089	Yes	1
ujis_japanese_ci	ujis	12	Yes	1
ujis_bin	ujis	91	Yes	1
ujis_japanese_nopad_ci	ujis	1036	Yes	1
ujis_nopad_bin	ujis	1115	Yes	1
sjis_japanese_ci	sjis	13	Yes	1
sjis_bin	sjis	88	Yes	1
sjis_japanese_nopad_ci	sjis	1037	Yes	1
sjis_nopad_bin	sjis	1112	Yes	1
hebrew_general_ci	hebrew	16	Yes	1
hebrew_bin	hebrew	71	Yes	1
hebrew_general_nopad_ci	hebrew	1040	Yes	1
hebrew_nopad_bin	hebrew	1095	Yes	1
tis620_thai_ci	tis620	18	Yes	4
tis620_bin	tis620	89	Yes	1
tis620_thai_nopad_ci	tis620	1042	Yes	4
tis620_nopad_bin	tis620	1113	Yes	1
euckr_korean_ci	euckr	19	Yes	1
euckr_bin	euckr	85	Yes	1
euckr_korean_nopad_ci	euckr	1043	Yes	1
euckr_nopad_bin	euckr	1109	Yes	1
koi8u_general_ci	koi8u	22	Yes	1
koi8u_bin	koi8u	75	Yes	1
koi8u_general_nopad_ci	koi8u	1046	Yes	1
koi8u_nopad_bin	koi8u	1099	Yes	1
gb2312_chinese_ci	gb2312	24	Yes	1
gb2312_bin	gb2312	86	Yes	1
gb2312_chinese_nopad_ci	gb2312	1048	Yes	1
gb2312_nopad_bin	gb2312	1110	Yes	1
greek_general_ci	greek	25	Yes	1
greek_bin	greek	70	Yes	1
greek_general_nopad_ci	greek	1049	Yes	1
greek_nopad_bin	greek	1094	Yes	1
cp1250_general_ci	cp1250	26	Yes	1
cp1250_czech_cs	cp1250	34	Yes	2
cp1250_croatian_ci	cp1250	44	Yes	1
cp1250_bin	cp1250	66	Yes	1
cp1250_polish_ci	cp1250	99	Yes	1
cp1250_general_nopad_ci	cp1250	1050	Yes	1
cp1250_nopad_bin	cp1250	1090	Yes	1
gbk_chinese_ci	gbk	28	Yes	1
gbk_bin	gbk	87	Yes	1
gbk_chinese_nopad_ci	gbk	1052	Yes	1
gbk_nopad_bin	gbk	1111	Yes	1
latin5_turkish_ci	latin5	30	Yes	1
latin5_bin	latin5	78	Yes	1
latin5_turkish_nopad_ci	latin5	1054	Yes	1
latin5_nopad_bin	latin5	1102	Yes	1
armSCII8_general_ci	armSCII8	32	Yes	1

armSCII8_bin	armSCII8   64	Yes	1
armSCII8_general_nopad_ci	armSCII8   1056	Yes	1
armSCII8_nopad_bin	armSCII8   1088	Yes	1
utf8_general_ci	utf8   33   Yes	Yes	1
utf8_bin	utf8   83	Yes	1
utf8_unicode_ci	utf8   192	Yes	8
utf8_icelandic_ci	utf8   193	Yes	8
utf8_latvian_ci	utf8   194	Yes	8
utf8_romanian_ci	utf8   195	Yes	8
utf8_slovenian_ci	utf8   196	Yes	8
utf8_polish_ci	utf8   197	Yes	8
utf8_estonian_ci	utf8   198	Yes	8
utf8_spanish_ci	utf8   199	Yes	8
utf8_swedish_ci	utf8   200	Yes	8
utf8_turkish_ci	utf8   201	Yes	8
utf8_czech_ci	utf8   202	Yes	8
utf8_danish_ci	utf8   203	Yes	8
utf8_lithuanian_ci	utf8   204	Yes	8
utf8_slovak_ci	utf8   205	Yes	8
utf8_spanish2_ci	utf8   206	Yes	8
utf8_roman_ci	utf8   207	Yes	8
utf8_persian_ci	utf8   208	Yes	8
utf8_esperanto_ci	utf8   209	Yes	8
utf8_hungarian_ci	utf8   210	Yes	8
utf8_sinhala_ci	utf8   211	Yes	8
utf8_german2_ci	utf8   212	Yes	8
utf8_croatian_mysql561_ci	utf8   213	Yes	8
utf8_unicode_520_ci	utf8   214	Yes	8
utf8_vietnamese_ci	utf8   215	Yes	8
utf8_general_mysql500_ci	utf8   223	Yes	1
utf8_croatian_ci	utf8   576	Yes	8
utf8_myanmar_ci	utf8   577	Yes	8
utf8_thai_520_w2	utf8   578	Yes	4
utf8_general_nopad_ci	utf8   1057	Yes	1
utf8_nopad_bin	utf8   1107	Yes	1
utf8_unicode_nopad_ci	utf8   1216	Yes	8
utf8_unicode_520_nopad_ci	utf8   1238	Yes	8
ucs2_general_ci	ucs2   35   Yes	Yes	1
ucs2_bin	ucs2   90	Yes	1
ucs2_unicode_ci	ucs2   128	Yes	8
ucs2_icelandic_ci	ucs2   129	Yes	8
ucs2_latvian_ci	ucs2   130	Yes	8
ucs2_romanian_ci	ucs2   131	Yes	8
ucs2_slovenian_ci	ucs2   132	Yes	8
ucs2_polish_ci	ucs2   133	Yes	8
ucs2_estonian_ci	ucs2   134	Yes	8
ucs2_spanish_ci	ucs2   135	Yes	8
ucs2_swedish_ci	ucs2   136	Yes	8
ucs2_turkish_ci	ucs2   137	Yes	8
ucs2_czech_ci	ucs2   138	Yes	8
ucs2_danish_ci	ucs2   139	Yes	8
ucs2_lithuanian_ci	ucs2   140	Yes	8
ucs2_slovak_ci	ucs2   141	Yes	8
ucs2_spanish2_ci	ucs2   142	Yes	8
ucs2_roman_ci	ucs2   143	Yes	8
ucs2_persian_ci	ucs2   144	Yes	8
ucs2_esperanto_ci	ucs2   145	Yes	8
ucs2_hungarian_ci	ucs2   146	Yes	8
ucs2_sinhala_ci	ucs2   147	Yes	8
ucs2_german2_ci	ucs2   148	Yes	8
ucs2_croatian_mysql561_ci	ucs2   149	Yes	8
ucs2_unicode_520_ci	ucs2   150	Yes	8
ucs2_vietnamese_ci	ucs2   151	Yes	8
ucs2_general_mysql500_ci	ucs2   159	Yes	1
ucs2_croatian_ci	ucs2   640	Yes	8
ucs2_myanmar_ci	ucs2   641	Yes	8
ucs2_thai_520_w2	ucs2   642	Yes	4
ucs2_general_nopad_ci	ucs2   1059	Yes	1
ucs2_nopad_bin	ucs2   1114	Yes	1
ucs2_unicode_nopad_ci	ucs2   1152	Yes	8
ucs2_unicode_520_nopad_ci	ucs2   1174	Yes	8
cp866_general_ci	cp866   36   Yes	Yes	1
cp866_bin	cp866   68	Yes	1
cp866_general_nopad_ci	cp866   1060	Yes	1
cp866_nopad_bin	cp866   1092	Yes	1
keybcs2_general_ci	keybcs2   37   Yes	Yes	1
keybcs2_bin	keybcs2   73	Yes	1

keybcs2_general_nopad_ci	keybcs2	1061		Yes		1
keybcs2_nopad_bin	keybcs2	1097		Yes		1
macce_general_ci	macce	38	Yes	Yes		1
macce_bin	macce	43		Yes		1
macce_general_nopad_ci	macce	1062		Yes		1
macce_nopad_bin	macce	1067		Yes		1
macroman_general_ci	macroman	39	Yes	Yes		1
macroman_bin	macroman	53		Yes		1
macroman_general_nopad_ci	macroman	1063		Yes		1
macroman_nopad_bin	macroman	1077		Yes		1
cp852_general_ci	cp852	40	Yes	Yes		1
cp852_bin	cp852	81		Yes		1
cp852_general_nopad_ci	cp852	1064		Yes		1
cp852_nopad_bin	cp852	1105		Yes		1
latin7_estonian_cs	latin7	20		Yes		1
latin7_general_ci	latin7	41	Yes	Yes		1
latin7_general_cs	latin7	42		Yes		1
latin7_bin	latin7	79		Yes		1
latin7_general_nopad_ci	latin7	1065		Yes		1
latin7_nopad_bin	latin7	1103		Yes		1
utf8mb4_general_ci	utf8mb4	45	Yes	Yes		1
utf8mb4_bin	utf8mb4	46		Yes		1
utf8mb4_unicode_ci	utf8mb4	224		Yes		8
utf8mb4_icelandic_ci	utf8mb4	225		Yes		8
utf8mb4_latvian_ci	utf8mb4	226		Yes		8
utf8mb4_romanian_ci	utf8mb4	227		Yes		8
utf8mb4_slovenian_ci	utf8mb4	228		Yes		8
utf8mb4_polish_ci	utf8mb4	229		Yes		8
utf8mb4_estonian_ci	utf8mb4	230		Yes		8
utf8mb4_spanish_ci	utf8mb4	231		Yes		8
utf8mb4_swedish_ci	utf8mb4	232		Yes		8
utf8mb4_turkish_ci	utf8mb4	233		Yes		8
utf8mb4_czech_ci	utf8mb4	234		Yes		8
utf8mb4_danish_ci	utf8mb4	235		Yes		8
utf8mb4_lithuanian_ci	utf8mb4	236		Yes		8
utf8mb4_slovak_ci	utf8mb4	237		Yes		8
utf8mb4_spanish2_ci	utf8mb4	238		Yes		8
utf8mb4_roman_ci	utf8mb4	239		Yes		8
utf8mb4_persian_ci	utf8mb4	240		Yes		8
utf8mb4_esperanto_ci	utf8mb4	241		Yes		8
utf8mb4_hungarian_ci	utf8mb4	242		Yes		8
utf8mb4_sinhala_ci	utf8mb4	243		Yes		8
utf8mb4_german2_ci	utf8mb4	244		Yes		8
utf8mb4_croatian_mysql561_ci	utf8mb4	245		Yes		8
utf8mb4_unicode_520_ci	utf8mb4	246		Yes		8
utf8mb4_vietnamese_ci	utf8mb4	247		Yes		8
utf8mb4_croatian_ci	utf8mb4	608		Yes		8
utf8mb4_myanmar_ci	utf8mb4	609		Yes		8
utf8mb4_thai_520_w2	utf8mb4	610		Yes		4
utf8mb4_general_nopad_ci	utf8mb4	1069		Yes		1
utf8mb4_nopad_bin	utf8mb4	1070		Yes		1
utf8mb4_unicode_nopad_ci	utf8mb4	1248		Yes		8
utf8mb4_unicode_520_nopad_ci	utf8mb4	1270		Yes		8
cp1251_bulgarian_ci	cp1251	14		Yes		1
cp1251_ukrainian_ci	cp1251	23		Yes		1
cp1251_bin	cp1251	50		Yes		1
cp1251_general_ci	cp1251	51	Yes	Yes		1
cp1251_general_cs	cp1251	52		Yes		1
cp1251_nopad_bin	cp1251	1074		Yes		1
cp1251_general_nopad_ci	cp1251	1075		Yes		1
utf16_general_ci	utf16	54	Yes	Yes		1
utf16_bin	utf16	55		Yes		1
utf16_unicode_ci	utf16	101		Yes		8
utf16_icelandic_ci	utf16	102		Yes		8
utf16_latvian_ci	utf16	103		Yes		8
utf16_romanian_ci	utf16	104		Yes		8
utf16_slovenian_ci	utf16	105		Yes		8
utf16_polish_ci	utf16	106		Yes		8
utf16_estonian_ci	utf16	107		Yes		8
utf16_spanish_ci	utf16	108		Yes		8
utf16_swedish_ci	utf16	109		Yes		8
utf16_turkish_ci	utf16	110		Yes		8
utf16_czech_ci	utf16	111		Yes		8
utf16_danish_ci	utf16	112		Yes		8
utf16_lithuanian_ci	utf16	113		Yes		8
utf16_slovak_ci	utf16	114		Yes		8
utf16_spanish2_ci	utf16	115		Yes		8
utf16_roman_ci	utf16	116		Yes		8

utf16_persian_ci	utf16	117	Yes	8
utf16_esperanto_ci	utf16	118	Yes	8
utf16_hungarian_ci	utf16	119	Yes	8
utf16_sinhala_ci	utf16	120	Yes	8
utf16_german2_ci	utf16	121	Yes	8
utf16_croatian_mysql1561_ci	utf16	122	Yes	8
utf16_unicode_520_ci	utf16	123	Yes	8
utf16_vietnamese_ci	utf16	124	Yes	8
utf16_croatian_ci	utf16	672	Yes	8
utf16_myanmar_ci	utf16	673	Yes	8
utf16_thai_520_w2	utf16	674	Yes	4
utf16_general_nopad_ci	utf16	1078	Yes	1
utf16_nopad_bin	utf16	1079	Yes	1
utf16_unicode_nopad_ci	utf16	1125	Yes	8
utf16_unicode_520_nopad_ci	utf16	1147	Yes	8
utf16le_general_ci	utf16le	56	Yes	1
utf16le_bin	utf16le	62	Yes	1
utf16le_general_nopad_ci	utf16le	1080	Yes	1
utf16le_nopad_bin	utf16le	1086	Yes	1
cp1256_general_ci	cp1256	57	Yes	1
cp1256_bin	cp1256	67	Yes	1
cp1256_general_nopad_ci	cp1256	1081	Yes	1
cp1256_nopad_bin	cp1256	1091	Yes	1
cp1257_lithuanian_ci	cp1257	29	Yes	1
cp1257_bin	cp1257	58	Yes	1
cp1257_general_ci	cp1257	59	Yes	1
cp1257_nopad_bin	cp1257	1082	Yes	1
cp1257_general_nopad_ci	cp1257	1083	Yes	1
utf32_general_ci	utf32	60	Yes	1
utf32_bin	utf32	61	Yes	1
utf32_unicode_ci	utf32	160	Yes	8
utf32_icelandic_ci	utf32	161	Yes	8
utf32_latvian_ci	utf32	162	Yes	8
utf32_romanian_ci	utf32	163	Yes	8
utf32_slovenian_ci	utf32	164	Yes	8
utf32_polish_ci	utf32	165	Yes	8
utf32_estonian_ci	utf32	166	Yes	8
utf32_spanish_ci	utf32	167	Yes	8
utf32_swedish_ci	utf32	168	Yes	8
utf32_turkish_ci	utf32	169	Yes	8
utf32_czech_ci	utf32	170	Yes	8
utf32_danish_ci	utf32	171	Yes	8
utf32_lithuanian_ci	utf32	172	Yes	8
utf32_slovak_ci	utf32	173	Yes	8
utf32_spanish2_ci	utf32	174	Yes	8
utf32_roman_ci	utf32	175	Yes	8
utf32_persian_ci	utf32	176	Yes	8
utf32_esperanto_ci	utf32	177	Yes	8
utf32_hungarian_ci	utf32	178	Yes	8
utf32_sinhala_ci	utf32	179	Yes	8
utf32_german2_ci	utf32	180	Yes	8
utf32_croatian_mysql1561_ci	utf32	181	Yes	8
utf32_unicode_520_ci	utf32	182	Yes	8
utf32_vietnamese_ci	utf32	183	Yes	8
utf32_croatian_ci	utf32	736	Yes	8
utf32_myanmar_ci	utf32	737	Yes	8
utf32_thai_520_w2	utf32	738	Yes	4
utf32_general_nopad_ci	utf32	1084	Yes	1
utf32_nopad_bin	utf32	1085	Yes	1
utf32_unicode_nopad_ci	utf32	1184	Yes	8
utf32_unicode_520_nopad_ci	utf32	1206	Yes	8
binary	binary	63	Yes	1
geostd8_general_ci	geostd8	92	Yes	1
geostd8_bin	geostd8	93	Yes	1
geostd8_general_nopad_ci	geostd8	1116	Yes	1
geostd8_nopad_bin	geostd8	1117	Yes	1
cp932_japanese_ci	cp932	95	Yes	1
cp932_bin	cp932	96	Yes	1
cp932_japanese_nopad_ci	cp932	1119	Yes	1
cp932_nopad_bin	cp932	1120	Yes	1
eucjpms_japanese_ci	eucjpms	97	Yes	1
eucjpms_bin	eucjpms	98	Yes	1
eucjpms_japanese_nopad_ci	eucjpms	1121	Yes	1
eucjpms_nopad_bin	eucjpms	1122	Yes	1

322 rows in set (0.00 sec)

From MariaDB 10.6.1 , the

```
utf8*
collations listed above are renamed
utf8mb3*
```

A '

```
ci
' at the end of a collation name indicates the collation is case insensitive. A '
cs
' at the end of a collation name indicates the collation is case sensitive. A '
nopad
' as part of the name indicates that the collation is of type
NO PAD
as opposed to
PADSPACE
(see below).
```

## NO PAD collations

MariaDB starting with 10.2

Until MariaDB 10.1 , all collations were of type

```
PADSPACE
. From MariaDB 10.2 , 88 new
NO PAD
collations are available.
NO PAD
collations regard trailing spaces as normal characters. You can get a list of all of these by querying the Information Schema COLLATIONS Table as follows:
```

```
SELECT collation_name FROM information_schema.COLLATIONS
WHERE collation_name LIKE "%nopad%";
```

collation_name
big5_chinese_nopad_ci
big5_nopad_bin
...

## Changes

- MariaDB 10.6.1 changed the  
utf8  
character set by default to be an alias for utf8mb3 rather than the other way around. It can be set to imply  
utf8mb4  
by changing the value of the `old_mode` system variable.
- MariaDB 10.2.2 added 88  
NO PAD  
collations.
- MariaDB 10.1.15 added the  
utf8\_thai\_520\_w2  
,  
utf8mb4\_thai\_520\_w2  
,  
ucs2\_thai\_520\_w2  
,  
utf16\_thai\_520\_w2  
and  
utf32\_thai\_520\_w2  
collations.
- MariaDB 10.0.7 added the  
utf8\_myanmar\_ci  
,  
ucs2\_myanmar\_ci  
,  
utf8mb4\_myanmar\_ci  
,  
utf16\_myanmar\_ci

- and  
utf32\_myanmar\_ci  
collations.
- MariaDB 10.0.5 added the  
utf8\_german2\_ci  
,  
utf8mb4\_german2\_ci  
,  
ucs2\_german2\_ci  
,  
utf16\_german2\_ci  
and  
utf32\_german2\_ci  
collations.
  - MariaDB 5.1.41 added a Croatian collation patch from [Alexander Barkov](#) to fix some problems with the Croatian character set and LIKE queries. This patch added  
utf8\_croatian\_ci  
and  
ucs2\_croatian\_ci  
collations to MariaDB.

## See Also

- [Information Schema CHARACTER\\_SETS Table](#)
- [Information Schema COLLATIONS Table](#)

### 4.1.2.25 Character Sets and Collations

Simply put, a character set defines how and which characters are stored to support a particular language or languages. A collation, on the other hand, defines the order used when comparing strings (i.e. the position of any given character within the alphabet of that language)



#### Character Set and Collation Overview

*Introduction to character sets and collations.*



#### Supported Character Sets and Collations

*MariaDB supports the following character sets and collations.*



#### Setting Character Sets and Collations

*Changing from the default character set and collation.*



#### Unicode

*Unicode support.*



#### SHOW CHARACTER SET

*Available character sets.*



#### SHOW COLLATION

*Supported collations.*



#### Information Schema CHARACTER\_SETS Table

*Supported character sets.*



#### Information Schema COLLATIONS Table

*Supported collations.*



#### Internationalization and Localization

*Character sets, collations, time zones and locales.*



#### SET CHARACTER SET

*Maps all strings sent between the current client and the server with the given mapping.*



#### SET NAMES

*The character set used to send statements to the server, and results back to the client.*

There are 3 related questions .

### 4.1.3 Data and Time Data Types

## 4.1.3.1 DATE

### Contents

1. [Syntax](#)
2. [Description](#)
  1. [Oracle Mode](#)
3. [Examples](#)
4. [See Also](#)

## Syntax

DATE

## Description

A date. The supported range is '

1000-01-01

' to '

9999-12-31

'. MariaDB displays

DATE

values in '

YYYY-MM-DD

' format, but can be assigned dates in looser formats, including strings or numbers, as long as they make sense. These include a short year,

YY-MM-DD

, no delimiters,

YYMMDD

, or any other acceptable delimiter, for example

YYYY/MM/DD

. For details, see [date and time literals](#).

0000-00-00

' is a permitted special value (zero-date), unless the [NO\\_ZERO\\_DATE SQL\\_MODE](#) is used. Also, individual components of a date can be set to 0 (for example: '

2015-00-12

'), unless the [NO\\_ZERO\\_IN\\_DATE SQL\\_MODE](#) is used. In many cases, the result of an expression involving a zero-date, or a date with zero-parts, is

NULL

. If the [ALLOW\\_INVALID\\_DATES SQL\\_MODE](#) is enabled, if the day part is in the range between 1 and 31, the date does not produce any error, even for months that have less than 31 days.

## Oracle Mode

MariaDB starting with 10.3

In [Oracle mode from MariaDB 10.3](#),

DATE

with a time portion is a synonym for [DATETIME](#). See also [mariadb\\_schema](#).

## Examples

```
CREATE TABLE t1 (d DATE);

INSERT INTO t1 VALUES ("2010-01-12"), ("2011-2-28"), ('120314'),('13*04*21');

SELECT * FROM t1;
+-----+
| d      |
+-----+
| 2010-01-12 |
| 2011-02-28 |
| 2012-03-14 |
| 2013-04-21 |
+-----+
```

## See Also

- [mariadb\\_schema](#) data type qualifier

### 4.1.3.2 TIME

#### Syntax

```
TIME [(<microsecond precision>)]
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Internal Format](#)
4. [Examples](#)
5. [See also](#)

#### Description

A time. The range is

```
'-838:59:59.999999'  
to  
'838:59:59.999999'  
. Microsecond precision can be from 0-6; if not specified 0 is used. Microseconds have been available since MariaDB 5.3 .
```

MariaDB displays

```
TIME  
values in  
'HH:MM:SS.#####'  
format, but allows assignment of times in looser formats, including 'D HH:MM:SS', 'HH:MM:SS', 'HH:MM', 'D HH:MM', 'D HH', 'SS', or  
'HHMMSS', as well as permitting dropping of any leading zeros when a delimiter is provided, for example '3:9:10'. For details, see date and time literals .
```

MariaDB starting with [10.1.2](#)

[MariaDB 10.1.2](#) introduced the [--mysql56-temporal-format](#) option, on by default, which allows MariaDB to store TIMES using the same low-level format MySQL 5.6 uses.

#### Internal Format

In [MariaDB 10.1.2](#) a new temporal format was introduced from MySQL 5.6 that alters how the

```
TIME  
,  
DATETIME  
and  
TIMESTAMP  
columns operate at lower levels. These changes allow these temporal data types to have fractional parts and negative values. You can disable  
this feature using the
```

[mysql56\\_temporal\\_format](#)

system variable.

Tables that include

```
TIMESTAMP  
values that were created on an older version of MariaDB or that were created while the
```

[mysql56\\_temporal\\_format](#)

system variable was disabled continue to store data using the older data type format.

In order to update table columns from the older format to the newer format, execute an

[ALTER TABLE... MODIFY COLUMN](#)

statement that changes the column to the \*same\* data type. This change may be needed if you want to export the table's tablespace and import it onto a server that has

```
mysql56_temporal_format=ON  
set (see MDEV-15225 ).
```

For instance, if you have a

TIME

column in your table:

```
SHOW VARIABLES LIKE 'mysql156_temporal_format';

+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| mysql156_temporal_format | ON      |
+-----+-----+

ALTER TABLE example_table MODIFY ts_col TIME;
```

When MariaDB executes the

ALTER TABLE

statement, it converts the data from the older temporal format to the newer one.

In the event that you have several tables and columns using temporal data types that you want to switch over to the new format, make sure the system variable is enabled, then perform a dump and restore using

mysqldump

. The columns using relevant temporal data types are restored using the new temporal format.

Starting from MariaDB 10.5.1 columns with old temporal formats are marked with a

/\* mariadb-5.3 \*/

comment in the output of

SHOW CREATE TABLE

,

SHOW COLUMNS

,

DESCRIBE

statements, as well as in the

COLUMN\_TYPE

column of the

INFORMATION\_SCHEMA.COLUMNS Table

```
SHOW CREATE TABLE mariadb5312_time\G
***** 1. row *****
Table: mariadb5312_time
Create Table: CREATE TABLE `mariadb5312_time` (
  `t0` time /* mariadb-5.3 */ DEFAULT NULL,
  `t6` time(6) /* mariadb-5.3 */ DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

Note, columns with the current format are not marked with a comment.

## Examples

```

INSERT INTO time VALUES ('90:00:00'), ('800:00:00'), (800), (22), (151413), ('9:6:3'), ('12 09');

SELECT * FROM time;
+-----+
| t    |
+-----+
| 90:00:00 |
| 800:00:00 |
| 00:08:00 |
| 00:00:22 |
| 15:14:13 |
| 09:06:03 |
| 297:00:00 |
+-----+

```

## See also

- [Data Type Storage Requirements](#)

### 4.1.3.3 DATETIME

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Supported Values](#)
4. [Time Zones](#)
5. [Oracle Mode](#)
6. [Internal Format](#)
7. [Examples](#)
8. [See Also](#)

## Syntax

```
DATETIME [(microsecond precision)]
```

## Description

A date and time combination.

MariaDB displays

```

DATETIME
values in '
YYYY-MM-DD HH:MM:SS.fffffff
' format, but allows assignment of values to
DATETIME
columns using either strings or numbers. For details, see date and time literals.

```

DATETIME columns also accept `CURRENT_TIMESTAMP` as the default value.

[MariaDB 10.1.2](#) introduced the `--mysql56-temporal-format` option, on by default, which allows MariaDB to store DATETIMEs using the same low-level format MySQL 5.6 uses. For more information, see [Internal Format](#), below.

For storage requirements, see [Data Type Storage Requirements](#).

## Supported Values

MariaDB stores values that use the

```

DATETIME
data type in a format that supports values between
1000-01-01 00:00:00.000000
and
9999-12-31 23:59:59.999999

```

MariaDB can also store `microseconds` with a precision between 0 and 6. If no microsecond precision is specified, then 0 is used by default.

MariaDB also supports '

```

0000-00-00
' as a special zero-date value, unless NO_ZERO_DATE is specified in the SQL_MODE. Similarly, individual components of a date can be set to
```

0  
(for example: '2015-00-12')  
'), unless `NO_ZERO_DATE` is specified in the `SQL_MODE`. In many cases, the result of an expression involving a zero-date, or a date with zero-parts, is  
NULL  
. If the `ALLOW_INVALID_DATES` `SQL_MODE` is enabled, if the day part is in the range between 1 and 31, the date does not produce any error, even for months that have less than 31 days.

## Time Zones

If a column uses the

`DATETIME`

data type, then any inserted values are stored as-is, so no automatic time zone conversions are performed.

MariaDB also does not currently support time zone literals that contain time zone identifiers. See [MDEV-11829](#) for more information.

MariaDB validates

`DATETIME`

literals against the session's time zone. For example, if a specific time range never occurred in a specific time zone due to daylight savings time, then

`DATETIME`

values within that range would be invalid for that time zone.

For example, daylight savings time started on March 10, 2019 in the US, so the time range between 02:00:00 and 02:59:59 is invalid for that day in US time zones:

```
SET time_zone = 'America/New_York';
Query OK, 0 rows affected (0.000 sec)

INSERT INTO timestamp_test VALUES ('2019-03-10 02:55:05');
ERROR 1292 (22007): Incorrect datetime value: '2019-03-10 02:55:05' for column `db1`.`timestamp_test`.`timestamp_test` at row 1
```

But that same time range is fine in other time zones, such as [Coordinated Universal Time \(UTC\)](#). For example:

```
SET time_zone = 'UTC';
Query OK, 0 rows affected (0.000 sec)

INSERT INTO timestamp_test VALUES ('2019-03-10 02:55:05');
Query OK, 1 row affected (0.002 sec)
```

## Oracle Mode

MariaDB starting with 10.3

In [Oracle mode from MariaDB 10.3](#),

`DATE`

with a time portion is a synonym for

`DATETIME`

. See also [mariadb\\_schema](#).

## Internal Format

In [MariaDB 10.1.2](#) a new temporal format was introduced from MySQL 5.6 that alters how the

`TIME`

,

`DATETIME`

and

`TIMESTAMP`

columns operate at lower levels. These changes allow these temporal data types to have fractional parts and negative values. You can disable this feature using the `mysql56_temporal_format` system variable.

Tables that include

`TIMESTAMP`

values that were created on an older version of MariaDB or that were created while the `mysql56_temporal_format` system variable was disabled continue to store data using the older data type format.

In order to update table columns from the older format to the newer format, execute an `ALTER TABLE... MODIFY COLUMN` statement that changes the column to the \*same\* data type. This change may be needed if you want to export the table's tablespace and import it onto a server that has

`mysql56_temporal_format=ON`

set (see [MDEV-15225](#) ).

For instance, if you have a

DATETIME

column in your table:

```
SHOW VARIABLES LIKE 'mysql56_temporal_format';

+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| mysql56_temporal_format | ON    |
+-----+-----+

ALTER TABLE example_table MODIFY ts_col DATETIME;
```

When MariaDB executes the `ALTER TABLE` statement, it converts the data from the older temporal format to the newer one.

In the event that you have several tables and columns using temporal data types that you want to switch over to the new format, make sure the system variable is enabled, then perform a dump and restore using

`mysqldump`

. The columns using relevant temporal data types are restored using the new temporal format.

Starting from [MariaDB 10.5.1](#) columns with old temporal formats are marked with a

```
/* mariadb-5.3 */
comment in the output of SHOW CREATE TABLE , SHOW COLUMNS , DESCRIBE statements, as well as in the
COLUMN_TYPE
column of the INFORMATION_SCHEMA.COLUMNS Table .
```

```
SHOW CREATE TABLE mariadb5312_datetime\G
***** 1. row *****
Table: mariadb5312_datetime
Create Table: CREATE TABLE `mariadb5312_datetime` (
`dt0` datetime /* mariadb-5.3 */ DEFAULT NULL,
`dt6` datetime(6) /* mariadb-5.3 */ DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

## Examples

```
CREATE TABLE t1 (d DATETIME);

INSERT INTO t1 VALUES ("2011-03-11"), ("2012-04-19 13:08:22"),
("2013-07-18 13:44:22.123456");

SELECT * FROM t1;
+-----+
| d          |
+-----+
| 2011-03-11 00:00:00 |
| 2012-04-19 13:08:22 |
| 2013-07-18 13:44:22 |
+-----+
```

```
CREATE TABLE t2 (d DATETIME(6));

INSERT INTO t2 VALUES ("2011-03-11"), ("2012-04-19 13:08:22"),
("2013-07-18 13:44:22.123456");

SELECT * FROM t2;
+-----+
| d          |
+-----+
| 2011-03-11 00:00:00.000000 |
| 2012-04-19 13:08:22.000000 |
| 2013-07-18 13:44:22.123456 |
+-----+
```

Strings used in datetime context are automatically converted to datetime(6). If you want to have a datetime without seconds, you should use `CONVERT(..,datetime)` .

```

SELECT CONVERT('2007-11-30 10:30:19',datetime);
+-----+
| CONVERT('2007-11-30 10:30:19',datetime) |
+-----+
| 2007-11-30 10:30:19 |
+-----+

SELECT CONVERT('2007-11-30 10:30:19',datetime(6));
+-----+
| CONVERT('2007-11-30 10:30:19',datetime(6)) |
+-----+
| 2007-11-30 10:30:19.000000 |
+-----+

```

## See Also

- [Data Type Storage Requirements](#)
- [CONVERT\(\)](#)
- [Oracle mode from MariaDB 10.3](#)
- [mariadb\\_schema data type qualifier](#)

## 4.1.3.4 TIMESTAMP

### Syntax

`TIMESTAMP [(<microsecond precision>)]`

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Supported Values](#)
4. [Automatic Values](#)
5. [Time Zones](#)
6. [Limitations](#)
7. [SQL\\_MODE=MAXDB](#)
8. [Internal Format](#)
9. [Examples](#)
10. [See Also](#)

### Description

A timestamp in the format

`YYYY-MM-DD HH:MM:SS.fffffff`

The timestamp field is generally used to define at which moment in time a row was added or updated and by default will automatically be assigned the current datetime when a record is inserted or updated. The automatic properties only apply to the first TIMESTAMP in the record; subsequent TIMESTAMP columns will not be changed.

MariaDB starting with [10.1.2](#)

MariaDB [10.1.2](#) introduced the `--mysql56-temporal-format` option, on by default, which allows MariaDB to store TIMESTAMPs using the same low-level format MySQL 5.6 uses.

For more information, see [Internal Format](#).

### Supported Values

MariaDB stores values that use the

`TIMESTAMP` data type as the number of seconds since '1970-01-01 00:00:00' ( [UTC](#) ). This means that the

`TIMESTAMP` data type can hold values between '1970-01-01 00:00:01' ( [UTC](#) ) and '2038-01-19 03:14:07' ( [UTC](#) ).

MariaDB can also store `microseconds` with a precision between 0 and 6. If no microsecond precision is specified, then 0 is used by default.

### Automatic Values

MariaDB has special behavior for the first column that uses the  
TIMESTAMP  
data type in a specific table. For the first column that uses the  
TIMESTAMP  
data type in a specific table, MariaDB automatically assigns the following properties to the column:

- DEFAULT CURRENT\_TIMESTAMP
- ON UPDATE CURRENT\_TIMESTAMP

This means that if the column is not explicitly assigned a value in an  
INSERT  
or  
UPDATE  
query, then MariaDB will automatically initialize the column's value with the current date and time.

This automatic initialization for

INSERT  
and  
UPDATE  
queries can also be **explicitly enabled** for a column that uses the  
TIMESTAMP  
data type by specifying the  
DEFAULT CURRENT\_TIMESTAMP  
and  
ON UPDATE CURRENT\_TIMESTAMP  
clauses for the column. In these clauses, any synonym of

CURRENT\_TIMESTAMP

is accepted, including

CURRENT\_TIMESTAMP()

,

NOW()

,

LOCALTIME

,

LOCALTIME()

,

LOCALTIMESTAMP

, and

LOCALTIMESTAMP()

.

This automatic initialization for

INSERT  
queries can also be **explicitly disabled** for a column that uses the  
TIMESTAMP  
data type by specifying a constant  
DEFAULT  
value. For example,  
DEFAULT 0

.

This automatic initialization for

UPDATE  
queries can also be **explicitly disabled** for a column that uses the  
TIMESTAMP  
data type by specifying a  
DEFAULT  
clause for the column, but no  
ON UPDATE  
clause. If a  
DEFAULT  
clause is explicitly specified for a column that uses the  
TIMESTAMP  
data type, but an  
ON UPDATE  
clause is not specified for the column, then the timestamp value will not automatically change when an  
UPDATE

statement is executed.

MariaDB also has special behavior if

NULL  
is assigned to column that uses the  
TIMESTAMP  
data type. If the column is assigned the  
NULL  
value in an  
INSERT  
or  
UPDATE  
query, then MariaDB will automatically initialize the column's value with the current date and time. For details, see [NULL values in MariaDB](#).

This automatic initialization for

NULL  
values can also be **explicitly disabled** for a column that uses the  
TIMESTAMP  
data type by specifying the  
NULL  
attribute for the column. In this case, if the column's value is set to  
NULL  
, then the column's value will actually be set to  
NULL

## Time Zones

If a column uses the

TIMESTAMP  
data type, then any inserted values are converted from the session's time zone to [Coordinated Universal Time \(UTC\)](#) when stored, and converted back to the session's time zone when retrieved.

MariaDB does not currently store any time zone identifier with the value of the

TIMESTAMP  
data type. See [MDEV-10018](#) for more information.

MariaDB does not currently support time zone literals that contain time zone identifiers. See [MDEV-11829](#) for more information.

## Limitations

- Because the TIMESTAMP value is stored as Epoch Seconds, the timestamp value '1970-01-01 00:00:00' (UTC) is reserved since the second #0 is used to represent '0000-00-00 00:00:00'.
- In [MariaDB 5.5](#) and before there could only be one TIMESTAMP column per table that had CURRENT\_TIMESTAMP defined as its default value. This limit has no longer applied since [MariaDB 10.0](#).

## SQL\_MODE=MAXDB

If the `SQL_MODE` is set to

MAXDB  
, TIMESTAMP fields will be silently converted to [DATETIME](#).

## Internal Format

In [MariaDB 10.1.2](#) a new temporal format was introduced from MySQL 5.6 that alters how the

TIME  
,  
DATETIME  
and  
TIMESTAMP  
columns operate at lower levels. These changes allow these temporal data types to have fractional parts and negative values. You can disable this feature using the

`mysql56_temporal_format`

system variable.

Tables that include

TIMESTAMP  
values that were created on an older version of MariaDB or that were created while the

## mysql156\_temporal\_format

system variable was disabled continue to store data using the older data type format.

In order to update table columns from the older format to the newer format, execute an

```
ALTER TABLE... MODIFY COLUMN
```

statement that changes the column to the \*same\* data type. This change may be needed if you want to export the table's tablespace and import it onto a server that has

```
mysql156_temporal_format=ON  
set (see MDEV-15225 ).
```

For instance, if you have a

```
TIMESTAMP  
column in your table:
```

```
SHOW VARIABLES LIKE 'mysql156_temporal_format';  
  
+-----+  
| Variable_name      | Value |  
+-----+  
| mysql156_temporal_format | ON    |  
+-----+  
  
ALTER TABLE example_table MODIFY ts_col TIMESTAMP;
```

When MariaDB executes the

```
ALTER TABLE
```

statement, it converts the data from the older temporal format to the newer one.

In the event that you have several tables and columns using temporal data types that you want to switch over to the new format, make sure the system variable is enabled, then perform a dump and restore using

```
mysqldump  
. The columns using relevant temporal data types are restored using the new temporal format.
```

Starting from MariaDB 10.5.1 columns with old temporal formats are marked with a

```
/* mariadb-5.3 */  
comment in the output of
```

```
SHOW CREATE TABLE
```

```
,  
SHOW COLUMNS
```

```
,  
DESCRIBE
```

statements, as well as in the

```
COLUMN_TYPE  
column of the
```

```
INFORMATION_SCHEMA.COLUMNS Table
```

```
SHOW CREATE TABLE mariadb5312_timestamp\G  
***** 1. row *****  
Table: mariadb5312_timestamp  
Create Table: CREATE TABLE `mariadb5312_timestamp` (  
  `ts0` timestamp /* mariadb-5.3 */ NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),  
  `ts6` timestamp(6) /* mariadb-5.3 */ NOT NULL DEFAULT '0000-00-00 00:00:00.000000'  
) ENGINE=MyISAM DEFAULT CHARSET=latin1
```

**Note:** Prior to MySQL 4.1 a different format for the TIMESTAMP datatype was used. This format is unsupported in MariaDB 5.1 and upwards.

# Examples

```
CREATE TABLE t (id INT, ts TIMESTAMP);

DESC t;
+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default            | Extra          |
+-----+-----+-----+-----+
| id    | int(11)   | YES  |     | NULL               |                |
| ts    | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-----+-----+-----+-----+

INSERT INTO t(id) VALUES (1),(2);

SELECT * FROM t;
+-----+-----+
| id  | ts        |
+-----+-----+
| 1   | 2013-07-22 12:50:05 |
| 2   | 2013-07-22 12:50:05 |
+-----+-----+

INSERT INTO t VALUES (3,NULL),(4,'2001-07-22 12:12:12');

SELECT * FROM t;
+-----+-----+
| id  | ts        |
+-----+-----+
| 1   | 2013-07-22 12:50:05 |
| 2   | 2013-07-22 12:50:05 |
| 3   | 2013-07-22 12:51:56 |
| 4   | 2001-07-22 12:12:12 |
+-----+-----+
```

Converting to Unix epoch:

```
SELECT ts, UNIX_TIMESTAMP(ts) FROM t;
+-----+-----+
| ts           | UNIX_TIMESTAMP(ts) |
+-----+-----+
| 2013-07-22 12:50:05 | 1374490205 |
| 2013-07-22 12:50:05 | 1374490205 |
| 2013-07-22 12:51:56 | 1374490316 |
| 2001-07-22 12:12:12 | 995796732 |
+-----+-----+
```

Update also changes the timestamp:

```
UPDATE t set id=5 WHERE id=1;

SELECT * FROM t;
+-----+-----+
| id  | ts        |
+-----+-----+
| 5   | 2013-07-22 14:52:33 |
| 2   | 2013-07-22 12:50:05 |
| 3   | 2013-07-22 12:51:56 |
| 4   | 2001-07-22 12:12:12 |
+-----+-----+
```

Default NULL:

```

CREATE TABLE t2 (id INT, ts TIMESTAMP NULL ON UPDATE CURRENT_TIMESTAMP);

INSERT INTO t2(id) VALUES (1),(2);

SELECT * FROM t2;

INSERT INTO t2(id) VALUES (1),(2);

SELECT * FROM t2;
+-----+
| id   | ts    |
+-----+
| 1    | NULL  |
| 2    | NULL  |
+-----+

UPDATE t2 SET id=3 WHERE id=1;

SELECT * FROM t2;
+-----+
| id   | ts      |
+-----+
| 3    | 2013-07-22 15:32:22 |
| 2    | NULL    |
+-----+

```

Only the first timestamp is automatically inserted and updated:

```

CREATE TABLE t3 (id INT, ts1 TIMESTAMP, ts2 TIMESTAMP);

INSERT INTO t3(id) VALUES (1),(2);

SELECT * FROM t3;
+-----+
| id   | ts1           | ts2           |
+-----+
| 1    | 2013-07-22 15:35:07 | 0000-00-00 00:00:00 |
| 2    | 2013-07-22 15:35:07 | 0000-00-00 00:00:00 |
+-----+

DESC t3;
+-----+
| Field | Type      | Null | Key | Default          | Extra          |
+-----+
| id    | int(11)   | YES  |      | NULL             |                |
| ts1   | timestamp  | NO   |      | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| ts2   | timestamp  | NO   |      | 0000-00-00 00:00:00 |
+-----+

```

Explicitly setting a timestamp with the `CURRENT_TIMESTAMP` function:

```

INSERT INTO t3(id,ts2) VALUES (3,CURRENT_TIMESTAMP());

SELECT * FROM t3;
+-----+
| id   | ts1           | ts2           |
+-----+
| 1    | 2013-07-22 15:35:07 | 0000-00-00 00:00:00 |
| 2    | 2013-07-22 15:35:07 | 0000-00-00 00:00:00 |
| 3    | 2013-07-22 15:38:52 | 2013-07-22 15:38:52 |
+-----+

```

Specifying the timestamp as NOT NULL:

```

CREATE TABLE t4 (id INT, ts TIMESTAMP NOT NULL);

INSERT INTO t4(id) VALUES (1);
SELECT SLEEP(1);
INSERT INTO t4(id,ts) VALUES (2,NULL);

SELECT * FROM t4;

```

## See Also

- Data Type Storage Requirements

## 4.1.3.5 YEAR Data Type

### Syntax

```
YEAR[(4)]
```

### Description

A year in two-digit or four-digit format. The default is four-digit format. Note that the two-digit format has been deprecated since [MariaDB 5.5.27](#).

In four-digit format, the allowable values are 1901 to 2155, and 0000. In two-digit format, the allowable values are 70 to 69, representing years from 1970 to 2069. MariaDB displays YEAR values in YYYY format, but allows you to assign values to YEAR columns using either strings or numbers.

Inserting numeric zero has a different result for YEAR(4) and YEAR(2). For YEAR(2), the value

00

reflects the year 2000. For YEAR(4), the value

0000

reflects the year zero. This only applies to numeric zero. String zero always reflects the year 2000.

### Examples

Accepting a string or a number:

```
CREATE TABLE y(y YEAR);

INSERT INTO y VALUES (1990),('2012');

SELECT * FROM y;
+---+
| y   |
+---+
| 1990 |
| 2012 |
+---+
```

With `strict_mode` set, the default from [MariaDB 10.2.4](#):

Out of range:

```
INSERT INTO y VALUES (1005),('3080');
ERROR 1264 (22003): Out of range value for column 'y' at row 1

INSERT INTO y VALUES ('2013-12-12');
ERROR 1265 (01000): Data truncated for column 'y' at row 1

SELECT * FROM y;
+---+
| y   |
+---+
| 1990 |
| 2012 |
+---+
```

With `strict_mode` unset, the default until [MariaDB 10.2.3](#):

Out of range:

```

INSERT INTO y VALUES (1005),('3080');
Query OK, 2 rows affected, 2 warnings (0.05 sec)
Records: 2 Duplicates: 0 Warnings: 2

SHOW WARNINGS;
+-----+
| Level | Code | Message |
+-----+
| Warning | 1264 | Out of range value for column 'y' at row 1 |
| Warning | 1264 | Out of range value for column 'y' at row 2 |
+-----+

SELECT * FROM y;
+---+
| y |
+---+
| 1990 |
| 2012 |
| 0000 |
| 0000 |
+---+

```

Truncating:

```

INSERT INTO y VALUES ('2013-12-12');
Query OK, 1 row affected, 1 warning (0.05 sec)

SHOW WARNINGS;
+-----+
| Level | Code | Message |
+-----+
| Warning | 1265 | Data truncated for column 'y' at row 1 |
+-----+

SELECT * FROM y;
+---+
| y |
+---+
| 1990 |
| 2012 |
| 0000 |
| 0000 |
| 2013 |
+---+

```

Difference between YEAR(2) and YEAR(4), and string and numeric zero:

```

CREATE TABLE y2(y YEAR(4), y2 YEAR(2));
Query OK, 0 rows affected, 1 warning (0.40 sec)

Note (Code 1287): 'YEAR(2)' is deprecated and will be removed in a future release.
Please use YEAR(4) instead

INSERT INTO y2 VALUES(0,0),('0','0');

SELECT YEAR(y),YEAR(y2) FROM y2;
+-----+
| YEAR(y) | YEAR(y2) |
+-----+
| 0 | 2000 |
| 2000 | 2000 |
+-----+

```

## See Also

- [YEAR\(\) function](#)

## 4.1.4 Geometry Types

## Contents

1. Description
2. Examples
  1. POINT
  2. LINESTRING
  3. POLYGON
  4. MULTIPOINT
  5. MULTILINESTRING
  6. MULTIPOLYGON
  7. GEOMETRYCOLLECTION
  8. GEOMETRY

## Description

MariaDB provides a standard way of creating spatial columns for geometry types, for example, with `CREATE TABLE` or `ALTER TABLE`. Currently, spatial columns are supported for `MyISAM`, `InnoDB`, `NDB`, and `ARCHIVE` tables. See also `SPATIAL INDEX`.

The basic geometry type is

`GEOMETRY`

. But the type can be more specific. The following types are supported:

Geometry Types
<code>POINT</code>
<code>LINESTRING</code>
<code>POLYGON</code>
<code>MULTIPOINT</code>
<code>MULTILINESTRING</code>
<code>MULTIPOLYGON</code>
<code>GEOMETRYCOLLECTION</code>
<code>GEOMETRY</code>

## Examples

**Note:** For clarity, only one type is listed per table in the examples below, but a table row can contain multiple types. For example:

```
CREATE TABLE object (shapeA POLYGON, shapeB LINESTRING);
```

## POINT

```
CREATE TABLE gis_point (g POINT);
SHOW FIELDS FROM gis_point;
INSERT INTO gis_point VALUES
(PointFromText('POINT(10 10)'), 
(PointFromText('POINT(20 10)'), 
(PointFromText('POINT(20 20)'), 
(PointFromWKB(AsWKB(PointFromText('POINT(10 20))));
```

## LINESTRING

```
CREATE TABLE gis_line (g LINESTRING);
SHOW FIELDS FROM gis_line;
INSERT INTO gis_line VALUES
(LineFromText('LINESTRING(0 0,0 10,10 0)'), 
(LineStringFromText('LINESTRING(10 10,20 10,20 20,10 20,10 10)'), 
(LineStringFromWKB(AsWKB(LineString(Point(10, 10), Point(40, 10))));
```

## POLYGON

```

CREATE TABLE gis_polygon  (g POLYGON);
SHOW FIELDS FROM gis_polygon;
INSERT INTO gis_polygon VALUES
  (PolygonFromText('POLYGON((10 10,20 10,20 20,10 20,10 10))')),
  (PolyFromText('POLYGON((0 0,50 0,50 50,0 50,0 0), (10 10,20 10,20 20,10 20,10 10))')),
  (PolyFromWKB(AsWKB(Polygon(LineString(Point(0, 0), Point(30, 0), Point(30, 30), Point(0, 0))))));

```

## MULTIPOINT

```

CREATE TABLE gis_multi_point (g MULTIPOINT);
SHOW FIELDS FROM gis_multi_point;
INSERT INTO gis_multi_point VALUES
  (MultiPointFromText('MULTIPOINT(0 0,10 10,20,20 20)')),
  (MPointFromText('MULTIPOINT(1 1,11 11,11 21,21 21)')),
  (MPointFromWKB(AsWKB(MultiPoint(Point(3, 6), Point(4, 10))))));

```

## MULTILINESTRING

```

CREATE TABLE gis_multi_line (g MULTILINESTRING);
SHOW FIELDS FROM gis_multi_line;
INSERT INTO gis_multi_line VALUES
  (MultiLineStringFromText('MULTILINESTRING((10 48,10 21,10 0),(16 0,16 23,16 48))'),
  (MLineFromText('MULTILINESTRING((10 48,10 21,10 0))')),
  (MLineFromWKB(AsWKB(MultiLineString(LineString(Point(1, 2), Point(3, 5)), LineString(Point(2, 5), Point(5, 8), Point(21, 7))))));

```

## MULTIPOLYGON

```

CREATE TABLE gis_multi_polygon (g MULTIPOLYGON);
SHOW FIELDS FROM gis_multi_polygon;
INSERT INTO gis_multi_polygon VALUES
  (MultiPolygonFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67 13,59 13,
  (MPolyFromText('MULTIPOLYGON(((28 26,28 0,84 0,84 42,28 26),(52 18,66 23,73 9,48 6,52 18)),((59 18,67 18,67 13,59 13,
  (MPolyFromWKB(AsWKB(MultiPolygon(Polygon(LineString(Point(0, 3), Point(3, 3), Point(3, 0), Point(0, 3))))));

```

## GEOMETRYCOLLECTION

```

CREATE TABLE gis_geometrycollection (g GEOMETRYCOLLECTION);
SHOW FIELDS FROM gis_geometrycollection;
INSERT INTO gis_geometrycollection VALUES
  (GeomCollFromText('GEOMETRYCOLLECTION(POINT(0 0), LINESTRING(0 0,10 10))'),
  (GeometryFromWKB(AsWKB(GeometryCollection(Point(44, 6), LineString(Point(3, 6), Point(7, 9))))),
  (GeomFromText('GeometryCollection()')),
  (GeomFromText('GeometryCollection EMPTY')));

```

## GEOMETRY

```

CREATE TABLE gis_geometry (g GEOMETRY);
SHOW FIELDS FROM gis_geometry;
INSERT into gis_geometry SELECT * FROM gis_point;
INSERT into gis_geometry SELECT * FROM gis_line;
INSERT into gis_geometry SELECT * FROM gis_polygon;
INSERT into gis_geometry SELECT * FROM gis_multi_point;
INSERT into gis_geometry SELECT * FROM gis_multi_line;
INSERT into gis_geometry SELECT * FROM gis_multi_polygon;
INSERT into gis_geometry SELECT * FROM gis_geometrycollection;

```

## 4.1.5 AUTO\_INCREMENT

## Contents

1. Description
2. Setting or Changing the Auto\_Increment Value
3. InnoDB
4. Setting Explicit Values
5. Missing Values
6. Replication
7. CHECK Constraints, DEFAULT Values and Virtual Columns
8. Generating Auto\_Increment Values When Adding the Attribute
9. See Also

## Description

The

AUTO\_INCREMENT

attribute can be used to generate a unique identity for new rows. When you insert a new record to the table (or upon adding an AUTO\_INCREMENT attribute with the ALTER TABLE statement), and the auto\_increment field is NULL or DEFAULT (in the case of an INSERT), the value will automatically be incremented. This also applies to 0, unless the NO\_AUTO\_VALUE\_ON\_ZERO SQL\_MODE is enabled.

AUTO\_INCREMENT

columns start from 1 by default. The automatically generated value can never be lower than 0.

Each table can have only one

AUTO\_INCREMENT

column. It must be defined as a key (not necessarily the

PRIMARY KEY

or

UNIQUE

key). In some storage engines (including the default InnoDB ), if the key consists of multiple columns, the

AUTO\_INCREMENT

column must be the first column. Storage engines that permit the column to be placed elsewhere are Aria , MyISAM , MERGE , Spider , TokuDB , BLACKHOLE , FederatedX and Federated .

```
CREATE TABLE animals (
    id MEDIUMINT NOT NULL AUTO_INCREMENT,
    name CHAR(30) NOT NULL,
    PRIMARY KEY (id)
);

INSERT INTO animals (name) VALUES
    ('dog'), ('cat'), ('penguin'),
    ('fox'), ('whale'), ('ostrich');
```

```
SELECT * FROM animals;
```

id	name
1	dog
2	cat
3	penguin
4	fox
5	whale
6	ostrich

SERIAL

is an alias for

BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT UNIQUE

```

CREATE TABLE t (id SERIAL, c CHAR(1)) ENGINE=InnoDB;

SHOW CREATE TABLE t
***** 1. row *****
  Table: t
Create Table: CREATE TABLE `t` (
  `id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
  `c` char(1) DEFAULT NULL,
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1

```

## Setting or Changing the Auto\_Increment Value

You can use an

`ALTER TABLE`

statement to assign a new value to the `auto_increment` table option, or set the `insert_id` server system variable to change the next `AUTO_INCREMENT` value inserted by the current session.

`LAST_INSERT_ID()`

can be used to see the last `AUTO_INCREMENT` value inserted by the current session.

```

ALTER TABLE animals AUTO_INCREMENT=8;

INSERT INTO animals (name) VALUES ('aardvark');

SELECT * FROM animals;
+----+-----+
| id | name   |
+----+-----+
| 1  | dog    |
| 2  | cat    |
| 3  | penguin |
| 4  | fox    |
| 5  | whale   |
| 6  | ostrich |
| 8  | aardvark|
+----+-----+

SET insert_id=12;

INSERT INTO animals (name) VALUES ('gorilla');

SELECT * FROM animals;
+----+-----+
| id | name   |
+----+-----+
| 1  | dog    |
| 2  | cat    |
| 3  | penguin |
| 4  | fox    |
| 5  | whale   |
| 6  | ostrich |
| 8  | aardvark|
| 12 | gorilla|
+----+-----+

```

## InnoDB

Until [MariaDB 10.2.3](#), InnoDB used an auto-increment counter that is stored in memory. When the server restarts, the counter is re-initialized to the highest value used in the table, which cancels the effects of any `AUTO_INCREMENT = N` option in the table statements.

From [MariaDB 10.2.4](#), this restriction has been lifted and `AUTO_INCREMENT` is persistent.

See also [AUTO\\_INCREMENT Handling in InnoDB](#).

## Setting Explicit Values

It is possible to specify a value for an

`AUTO_INCREMENT`

column. If the key is primary or unique, the value must not already exist in the key.

If the new value is higher than the current maximum value, the

`AUTO_INCREMENT`

value is updated, so the next value will be higher. If the new value is lower than the current maximum value, the

`AUTO_INCREMENT`

value remains unchanged.

The following example demonstrates these behaviors:

```
CREATE TABLE t (id INTEGER UNSIGNED AUTO_INCREMENT PRIMARY KEY) ENGINE = InnoDB;

INSERT INTO t VALUES (NULL);
SELECT id FROM t;
+----+
| id |
+----+
| 1 |
+----+

INSERT INTO t VALUES (10); -- higher value
SELECT id FROM t;
+----+
| id |
+----+
| 1 |
| 10 |
+----+

INSERT INTO t VALUES (2); -- Lower value
INSERT INTO t VALUES (NULL); -- auto value
SELECT id FROM t;
+----+
| id |
+----+
| 1 |
| 2 |
| 10 |
| 11 |
+----+
```

The [ARCHIVE](#) storage engine does not allow to insert a value that is lower than the current maximum.

## Missing Values

An AUTO\_INCREMENT column normally has missing values. This happens because if a row is deleted, or an AUTO\_INCREMENT value is explicitly updated, old values are never re-used. The REPLACE statement also deletes a row, and its value is wasted. With InnoDB, values can be reserved by a transaction; but if the transaction fails (for example, because of a ROLLBACK) the reserved value will be lost.

Thus AUTO\_INCREMENT values can be used to sort results in a chronological order, but not to create a numeric sequence.

## Replication

To make master-master or Galera safe to use

`AUTO_INCREMENT`

one should use the system variables `auto_increment_increment` and `auto_increment_offset` to generate unique values for each server.

## CHECK Constraints, DEFAULT Values and Virtual Columns

MariaDB starting with 10.2.6

From MariaDB 10.2.6 auto\_increment columns are no longer permitted in CHECK constraints , DEFAULT value expressions and virtual columns . They were permitted in earlier versions, but did not work correctly. See [MDEV-11117](#) .

## Generating Auto\_Increment Values When Adding the Attribute

```
CREATE OR REPLACE TABLE t1 (a INT);
INSERT t1 VALUES (0),(0),(0);
ALTER TABLE t1 MODIFY a INT NOT NULL AUTO_INCREMENT PRIMARY KEY;
SELECT * FROM t1;
+---+
| a |
+---+
| 1 |
| 2 |
| 3 |
+---+
```

```
CREATE OR REPLACE TABLE t1 (a INT);
INSERT t1 VALUES (5),(0),(8),(0);
ALTER TABLE t1 MODIFY a INT NOT NULL AUTO_INCREMENT PRIMARY KEY;
SELECT * FROM t1;
+---+
| a |
+---+
| 5 |
| 6 |
| 8 |
| 9 |
+---+
```

If the `NO_AUTO_VALUE_ON_ZERO SQL_MODE` is set, zero values will not be automatically incremented:

```
SET SQL_MODE='no_auto_value_on_zero';
CREATE OR REPLACE TABLE t1 (a INT);
INSERT t1 VALUES (3), (0);
ALTER TABLE t1 MODIFY a INT NOT NULL AUTO_INCREMENT PRIMARY KEY;
SELECT * FROM t1;
+---+
| a |
+---+
| 0 |
| 3 |
+---+
```

## See Also

- [Getting Started with Indexes](#)
- [Sequences](#) - an alternative to auto\_increment available from [MariaDB 10.3](#)
- [AUTO\\_INCREMENT FAQ](#)
- [LAST\\_INSERT\\_ID\(\)](#)
- [AUTO\\_INCREMENT handling in InnoDB](#)
- [BLACKHOLE and AUTO\\_INCREMENT](#)
- [UUID\\_SHORT\(\)](#) - Generate unique ids
- [Generating Identifiers – from AUTO\\_INCREMENT to Sequence \(percona.com\)](#)

### 4.1.6 Data Type Storage Requirements

### 4.1.7 AUTO\_INCREMENT FAQ

## Contents

1. How do I get the last inserted auto\_increment value?
2. What if someone else inserts before I select my id?
3. How do I get the next value to be inserted?
4. How do I change what number auto\_increment starts with?
5. How do I renumber rows once I've deleted some in the middle?
6. Can I do group-wise auto\_increment?
7. How do I get the auto\_increment value in a BEFORE INSERT trigger?
8. How do I assign two fields the same auto\_increment value in one query?
9. Does the auto\_increment field have to be primary key?
10. InnoDB and AUTO\_INCREMENT
11. General Information To Read
12. Manual Notes
13. How to start a table with a set AUTO\_INCREMENT value?
14. See Also

## How do I get the last inserted auto\_increment value?

Use the [LAST\\_INSERT\\_ID\(\)](#) function:

```
SELECT LAST_INSERT_ID();
```

## What if someone else inserts before I select my id?

[LAST\\_INSERT\\_ID\(\)](#) is connection specific, so there is no problem from race conditions.

## How do I get the next value to be inserted?

You don't. Insert, then find out what you did with [LAST\\_INSERT\\_ID\(\)](#).

## How do I change what number auto\_increment starts with?

```
ALTER TABLE yourTable AUTO_INCREMENT = x;  
— Next insert will contain
```

x  
or

MAX(autoField) + 1  
, whichever is higher

or

```
INSERT INTO yourTable (autoField) VALUES (x);
```

— Next insert will contain

x+1  
or  
MAX(autoField) + 1  
, whichever is higher

Issuing [TRUNCATE TABLE](#) will delete all the rows in the table, and will reset the auto\_increment value to 0 in most cases (some earlier versions mapped TRUNCATE to DELETE for InnoDB tables, meaning the auto\_increment value would not be reset).

## How do I renumber rows once I've deleted some in the middle?

Typically, you don't want to. Gaps are hardly ever a problem; if your application can't handle gaps in the sequence, you probably should rethink your application.

## Can I do group-wise auto\_increment?

Yes, if you use the MyISAM engine .

## How do I get the auto\_increment value in a BEFORE INSERT trigger?

You don't. It's only available after insert.

## How do I assign two fields the same auto\_increment value in one query?

You can't, not even with an AFTER INSERT trigger. Insert, then go back and update using

```
LAST_INSERT_ID()  
. Those two statements could be wrapped into one stored procedure if you wish.
```

However, you can mimic this behavior with a BEFORE INSERT trigger and a second table to store the sequence position:

```
CREATE TABLE sequence (table_name VARCHAR(255), position INT UNSIGNED);  
INSERT INTO sequence VALUES ('testTable', 0);  
CREATE TABLE testTable (firstAuto INT UNSIGNED, secondAuto INT UNSIGNED);  
DELIMITER //  
CREATE TRIGGER testTable_BI BEFORE INSERT ON testTable FOR EACH ROW BEGIN  
    UPDATE sequence SET position = LAST_INSERT_ID(position + 1) WHERE table_name = 'testTable';  
    SET NEW.firstAuto = LAST_INSERT_ID();  
    SET NEW.secondAuto = LAST_INSERT_ID();  
END//  
DELIMITER ;  
INSERT INTO testTable VALUES (NULL, NULL), (NULL, NULL);  
SELECT * FROM testTable;  
  
+-----+-----+  
| firstAuto | secondAuto |  
+-----+-----+  
| 1 | 1 |  
| 2 | 2 |  
+-----+-----+
```

The same sequence table can maintain separate sequences for multiple tables (or separate sequences for different fields in the same table) by adding extra rows.

## Does the auto\_increment field have to be primary key?

No, it only has to be indexed. It doesn't even have to be unique.

## InnoDB and AUTO\_INCREMENT

See [AUTO\\_INCREMENT handling in XtraDB/InnoDB](#)

## General Information To Read

[AUTO\\_INCREMENT](#)

## Manual Notes

There can be only one

```
AUTO_INCREMENT  
column per table, it must be indexed, and it cannot have a  
DEFAULT  
value. An  
AUTO_INCREMENT
```

column works properly only if it contains only positive values. Inserting a negative number is regarded as inserting a very large positive number. This is done to avoid precision problems when numbers wrap over from positive to negative and also to ensure that you do not accidentally get an

```
AUTO_INCREMENT  
column that contains 0.
```

## How to start a table with a set AUTO\_INCREMENT value?

```

CREATE TABLE autoinc_test (
    h INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    m INT UNSIGNED
) AUTO_INCREMENT = 100;

INSERT INTO autoinc_test (m) VALUES (1);

SELECT * FROM autoinc_test;
+-----+-----+
| h    | m    |
+-----+-----+
| 100 | 1   |
+-----+-----+

```

## See Also

- [AUTO\\_INCREMENT](#)
- [AUTO\\_INCREMENT handling in XtraDB/InnoDB](#)
- [LAST\\_INSERT\\_ID\(\)](#)
- [BLACKHOLE and AUTO\\_INCREMENT](#)
- [Sequences](#) - an alternative to auto\_increment available from [MariaDB 10.3](#)

The initial version of this article was copied, with permission, from [http://hashmysql.org/wiki/Autoincrement\\_FAQ](http://hashmysql.org/wiki/Autoincrement_FAQ) on 2012-10-05.

## 4.1.8 NULL Values

### Contents

1. [Syntax](#)
2. [Comparison Operators](#)
3. [Ordering](#)
4. [Functions](#)
5. [AUTO\\_INCREMENT, TIMESTAMP and Virtual Columns](#)
6. [Inserting](#)
  1. [Examples](#)
7. [Primary Keys and UNIQUE Indexes](#)
8. [Oracle Compatibility](#)
9. [See Also](#)

NULL represents an unknown value. It is *not* an empty string (by default), or a zero value. These are all valid values, and are not NULLs.

When a table is [created](#) or the format [altered](#), columns can be specified as accepting NULL values, or not accepting them, with the

NULL  
and  
NOT NULL  
clauses respectively.

For example, a customer table could contain dates of birth. For some customers, this information is unknown, so the value could be NULL.

The same system could allocate a customer ID for each customer record, and in this case a NULL value would not be permitted.

```

CREATE TABLE customer (
    id INT NOT NULL,
    date_of_birth DATE NULL
    ...
)

```

User-defined variables are NULL until a value is explicitly assigned.

Stored routines parameters and local variables can always be set to NULL. If no DEFAULT value is specified for a local variable, its initial value will be NULL. If no value is assigned to an OUT parameter in a stored procedure, NULL is assigned at the end of the procedure.

## Syntax

The case of

NULL  
is not relevant.  
\N  
(uppercase) is an alias for  
NULL

The

IS

operator accepts  
UNKNOWN  
as an alias for  
NULL  
, which is meant for [boolean contexts](#).

## Comparison Operators

NULL values cannot be used with most [comparison operators](#). For example, =, >, >=, <, <=, or != cannot be used, as any comparison with a NULL always returns a NULL value, never true (1) or false (0).

```
SELECT NULL = NULL;
+-----+
| NULL = NULL |
+-----+
|      NULL |
+-----+  
  
SELECT 99 = NULL;
+-----+
| 99 = NULL |
+-----+
|      NULL |
+-----+
```

To overcome this, certain operators are specifically designed for use with NULL values. To cater for testing equality between two values that may contain NULLs, there's [<=>](#), NULL-safe equal.

```
SELECT 99 <=> NULL, NULL <=> NULL;
+-----+
| 99 <=> NULL | NULL <=> NULL |
+-----+
|          0 |           1 |
+-----+
```

Other operators for working with NULLs include [IS NULL](#) and [IS NOT NULL](#), [ISNULL](#) (for testing an expression) and [COALESCE](#) (for returning the first non-NULL parameter).

## Ordering

When you order by a field that may contain NULL values, any NULLs are considered to have the lowest value. So ordering in DESC order will see the NULLs appearing last. To force NULLs to be regarded as highest values, one can add another column which has a higher value when the main field is NULL. Example:

```
SELECT col1 FROM tab ORDER BY ISNULL(col1), col1;
```

Descending order, with NULLs first:

```
SELECT col1 FROM tab ORDER BY IF(col1 IS NULL, 0, 1), col1 DESC;
```

All NULL values are also regarded as equivalent for the purposes of the DISTINCT and GROUP BY clauses.

## Functions

In most cases, functions will return NULL if any of the parameters are NULL. There are also functions specifically for handling NULLs. These include [IFNULL\(\)](#), [NULLIF\(\)](#) and [COALESCE\(\)](#).

```

SELECT IFNULL(1,0);
+-----+
| IFNULL(1,0) |
+-----+
|      1      |
+-----+

SELECT IFNULL(NULL,10);
+-----+
| IFNULL(NULL,10) |
+-----+
|      10      |
+-----+

SELECT COALESCE(NULL,NULL,1);
+-----+
| COALESCE(NULL,NULL,1) |
+-----+
|      1      |
+-----+

```

Aggregate functions, such as `SUM` and `AVG` ignore NULLs.

```

CREATE TABLE t(x INT);

INSERT INTO t VALUES (1),(9),(NULL);

SELECT SUM(x) FROM t;
+-----+
| SUM(x) |
+-----+
|    10   |
+-----+

SELECT AVG(x) FROM t;
+-----+
| AVG(x) |
+-----+
| 5.0000 |
+-----+

```

The one exception is `COUNT(*)`, which counts rows, and doesn't look at whether a value is NULL or not. Compare for example, `COUNT(x)`, which ignores the NULL, and `COUNT(*)`, which counts it:

```

SELECT COUNT(x) FROM t;
+-----+
| COUNT(x) |
+-----+
|      2   |
+-----+

SELECT COUNT(*) FROM t;
+-----+
| COUNT(*) |
+-----+
|      3   |
+-----+

```

## AUTO\_INCREMENT, TIMESTAMP and Virtual Columns

MariaDB handles NULL values in a special way if the field is an `AUTO_INCREMENT`, a `TIMESTAMP` or a `virtual column`. Inserting a NULL value into a numeric `AUTO_INCREMENT` column will result in the next number in the auto increment sequence being inserted instead. This technique is frequently used with `AUTO_INCREMENT` fields, which are left to take care of themselves.

```

CREATE TABLE t2(id INT PRIMARY KEY AUTO_INCREMENT, letter CHAR(1));

INSERT INTO t2(letter) VALUES ('a'),('b');

SELECT * FROM t2;
+-----+
| id | letter |
+-----+
| 1  | a      |
| 2  | b      |
+-----+

```

Similarly, if a NULL value is assigned to a TIMESTAMP field, the current date and time is assigned instead.

```

CREATE TABLE t3 (x INT, ts TIMESTAMP);

INSERT INTO t3(x) VALUES (1),(2);

```

After a pause,

```

INSERT INTO t3(x) VALUES (3);

SELECT* FROM t3;
+-----+
| x    | ts          |
+-----+
| 1   | 2013-09-05 10:14:18 |
| 2   | 2013-09-05 10:14:18 |
| 3   | 2013-09-05 10:14:29 |
+-----+

```

If a

NULL  
is assigned to a  
VIRTUAL  
or  
PERSISTENT  
column, the default value is assigned instead.

```

CREATE TABLE virt (c INT, v INT AS (c+10) PERSISTENT) ENGINE=InnoDB;

INSERT INTO virt VALUES (1, NULL);

SELECT c, v FROM virt;
+-----+
| c    | v    |
+-----+
| 1    | 11  |
+-----+

```

In all these special cases,

NULL  
is equivalent to the  
DEFAULT  
keyword.

## Inserting

If a NULL value is single-row inserted into a column declared as NOT NULL, an error will be returned. However, if the [SQL mode](#) is not `strict` (default until [MariaDB 10.2.3](#)), if a NULL value is multi-row inserted into a column declared as NOT NULL, the implicit default for the column type will be inserted (and NOT the default value in the table definition). The implicit defaults are an empty string for string types, and the zero value for numeric, date and time types.

Since [MariaDB 10.2.4](#), by default both cases will result in an error.

## Examples

```
CREATE TABLE nulltest (
  a INT(11),
  x VARCHAR(10) NOT NULL DEFAULT 'a',
  y INT(11) NOT NULL DEFAULT 23
);
```

Single-row insert:

```
INSERT INTO nulltest (a,x,y) VALUES (1,NULL,NULL);
ERROR 1048 (23000): Column 'x' cannot be null
```

Multi-row insert with [SQL mode not strict](#) (default until MariaDB 10.2.3):

```
show variables like 'sql_mode%';
+-----+-----+
| Variable_name | Value           |
+-----+-----+
| sql_mode      | NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----+-----+

INSERT INTO nulltest (a,x,y) VALUES (1,NULL,NULL),(2,NULL,NULL);
Query OK, 2 rows affected, 4 warnings (0.08 sec)
Records: 2  Duplicates: 0  Warnings: 4
```

The specified defaults have not been used; rather, the implicit column type defaults have been inserted

```
SELECT * FROM nulltest;
+---+---+---+
| a | x | y |
+---+---+---+
| 1 | 0 |   |
| 2 | 0 |   |
+---+---+---+
```

## Primary Keys and UNIQUE Indexes

UNIQUE indexes can contain multiple NULL values.

Primary keys are never nullable.

MariaDB starting with 10.3

## Oracle Compatibility

In [Oracle mode](#), NULL can be used as a statement:

```
IF a=10 THEN NULL; ELSE NULL; END IF
```

In [Oracle mode](#), CONCAT and the Logical OR operator || ignore NULL .

When setting `sql_mode=EMPTY_STRING_IS_NULL`, empty strings and NULLs are the same thing. For example:

```
SET sql_mode=EMPTY_STRING_IS_NULL;
SELECT '' IS NULL; -- returns TRUE
INSERT INTO t1 VALUES (''); -- inserts NULL
```

## See Also

- [Primary Keys with Nullable Columns](#)
- [IS NULL operator](#)
- [IS NOT NULL operator](#)
- [ISNULL function](#)
- [COALESCE function](#)
- [IFNULL function](#)
- [NULLIF function](#)
- [CONNECT data types](#)
- [Oracle mode from MariaDB 10.3](#)

## 4.2 Character Sets and Collations

### 4.2.1 Character Set and Collation Overview

#### Contents

1. [What Are Character Sets and Collations](#)
2. [Viewing Character Sets and Collations](#)
3. [Changing Character Sets and Collations](#)

## What Are Character Sets and Collations

A character set is a set of characters while a collation is the rules for comparing and sorting a particular character set.

For example, a subset of a character set could consist of the letters

A  
,  
B  
and  
C  
. A default collation could define these as appearing in an ascending order of  
A, B, C  
.

If we consider different case characters, more complexity is added. A binary collation would evaluate the characters

A  
and  
a  
differently, ordering them in a particular way. A case-insensitive collation would evaluate  
A  
and  
a  
equivalently, while the German phone book collation evaluates the characters  
ue  
and  
ü  
equivalently.

A character set can have many collations associated with it, while each collation is only associated with one character set. In MariaDB, the character set name is always part of the collation name. For example, the

latin1\_german1\_ci  
collation applies only to the  
latin1  
character set. Each character set also has one default collation. The  
latin1  
default collation is  
latin1\_swedish\_ci  
.

As an example, by default, the character

y  
comes between  
x  
and  
z  
, while in Lithuanian, it's sorted between  
i  
and  
k  
. Similarly, the German phone book order is different to the German dictionary order, so while they share the same character set, the collation is different.

## Viewing Character Sets and Collations

In MariaDB, the default character set is latin1, and the default collation is latin1\_swedish\_ci (however this may differ in some distros, see for example [Differences in MariaDB in Debian](#)). You can view a full list of character sets and collations supported by MariaDB at [Supported Character Sets and Collations](#), or see what's supported on your server with the `SHOW CHARACTER SET` and `SHOW COLLATION` commands.

By default,

A  
comes before

Z

, so the following evaluates to true:

```
SELECT "A" < "Z";
+-----+
| "A" < "Z" |
+-----+
|      1 |
+-----+
```

By default, comparisons are case-insensitive:

```
SELECT "A" < "a", "A" = "a";
+-----+-----+
| "A" < "a" | "A" = "a" |
+-----+-----+
|      0 |      1 |
+-----+-----+
```

## Changing Character Sets and Collations

Character sets and collations can be set from the server level right down to the column level, as well as for client-server communication.

For example,

ue  
and  
ü

are by default evaluated differently.

```
SELECT 'Mueller' = 'Müller';
+-----+
| 'Müller' = 'Mueller' |
+-----+
|          0 |
+-----+
```

By using the [collation\\_connection](#) system variable to change the connection character set to

latin1\_german2\_ci  
, or German phone book, the same two characters will evaluate as equivalent.

```
SET collation_connection = latin1_german2_ci;

SELECT 'Mueller' = 'Müller';
+-----+
| 'Mueller' = 'Müller' |
+-----+
|          1 |
+-----+
```

See [Setting Character Sets and Collations](#) for more.

### 4.2.2 Supported Character Sets and Collations

### 4.2.3 Setting Character Sets and Collations

## Contents

1. Server Level
2. Database Level
3. Table Level
4. Column Level
5. Filenames
6. Literals
  1. Examples
  2. N
7. Stored Programs and Views
  1. Illegal Collation Mix
8. Example: Changing the Default Character Set To UTF-8
9. See Also

In MariaDB, the default [character set](#) is latin1, and the default collation is latin1\_swedish\_ci (however this may differ in some distros, see for example [Differences in MariaDB in Debian](#)). Both character sets and collations can be specified from the server right down to the column level, as well as for client-server connections. When changing a character set and not specifying a collation, the default collation for the new character set is always used.

Character sets and collations always cascade down, so a column without a specified collation will look for the table default, the table for the database, and the database for the server. It's therefore possible to have extremely fine-grained control over all the character sets and collations used in your data.

Default collations for each character set can be viewed with the [SHOW COLLATION](#) statement, for example, to find the default collation for the latin2 character set:

```
SHOW COLLATION LIKE 'latin2%';
+-----+-----+-----+-----+-----+
| Collation      | Charset | Id | Default | Compiled | Sortlen |
+-----+-----+-----+-----+-----+
| latin2_czech_cs | latin2   | 2  |        | Yes      |     4    |
| latin2_general_ci | latin2   | 9  | Yes    | Yes      |     1    |
| latin2_hungarian_ci | latin2   | 21 |        | Yes      |     1    |
| latin2_croatian_ci | latin2   | 27 |        | Yes      |     1    |
| latin2_bin       | latin2   | 77 |        | Yes      |     1    |
+-----+-----+-----+-----+-----+
```

## Server Level

The [character\\_set\\_server](#) system variable can be used to change the default server character set. It can be set both on startup or dynamically, with the [SET](#) command:

```
SET character_set_server = 'latin2';
```

Similarly, the [collation\\_server](#) variable is used for setting the default server collation.

```
SET collation_server = 'latin2_czech_cs';
```

## Database Level

The [CREATE DATABASE](#) and [ALTER DATABASE](#) statements have optional character set and collation clauses. If these are left out, the server defaults are used.

```
CREATE DATABASE czech_slovak_names
  CHARACTER SET = 'keybcs2'
  COLLATE = 'keybcs2_bin';
```

```
ALTER DATABASE czech_slovak_names COLLATE = 'keybcs2_general_ci';
```

To determine the default character set used by a database, use:

```
SHOW CREATE DATABASE czech_slovak_names;
+-----+-----+
| Database      | Create Database           |
+-----+-----+
| czech_slovak_names | CREATE DATABASE `czech_slovak_names` /*!40100 DEFAULT CHARACTER SET keybcs2 */ |
```

or alternatively, for the character set and collation:

CATALOG_NAME	SCHEMA_NAME	DEFAULT_CHARACTER_SET_NAME	DEFAULT_COLLATION_NAME	SQL_PATH
def	czech_slovak_names	keybcs2	keybcs2_general_ci	NULL
def	information_schema	utf8	utf8_general_ci	NULL
def	mysql	latin1	latin1_swedish_ci	NULL
def	performance_schema	utf8	utf8_general_ci	NULL
def	test	latin1	latin1_swedish_ci	NULL

It is also possible to specify only the collation, and, since each collation only applies to one character set, the associated character set will automatically be specified.

CREATE DATABASE danish_names COLLATE 'utf8_danish_ci';
SHOW CREATE DATABASE danish_names;
+-----+   Database   Create Database   +-----+   danish_names   CREATE DATABASE `danish_names` /*!40100 DEFAULT CHARACTER SET utf8 COLLATE utf8_danish_ci */   +-----+

Although there are `character_set_database` and `collation_database` system variables which can be set dynamically, these are used for determining the character set and collation for the default database, and should only be set by the server.

## Table Level

The `CREATE TABLE` and `ALTER TABLE` statements support optional character set and collation clauses, a MariaDB and MySQL extension to standard SQL.

```
CREATE TABLE english_names (id INT, name VARCHAR(40))
  CHARACTER SET 'utf8'
  COLLATE 'utf8_icelandic_ci';
```

If neither character set nor collation is provided, the database default will be used. If only the character set is provided, the default collation for that character set will be used . If only the collation is provided, the associated character set will be used. See [Supported Character Sets and Collations](#) .

```
ALTER TABLE table_name
  CONVERT TO CHARACTER SET charset_name [COLLATE collation_name];
```

If no collation is provided, the collation will be set to the default collation for that character set. See [Supported Character Sets and Collations](#) .

For `VARCHAR` or `TEXT` columns, `CONVERT TO CHARACTER SET` changes the data type if needed to ensure the new column is long enough to store as many characters as the original column.

For example, an ascii `TEXT` column requires a single byte per character, so the column can hold up to 65,535 characters. If the column is converted to `utf8`, 3 bytes can be required for each character, so the column will be converted to `MEDIUMTEXT` to be able to hold the same number of characters.

CONVERT TO CHARACTER SET `binary`  
 will convert `CHAR` , `VARCHAR` and `TEXT` columns to `BINARY` , `VARBINARY` and `BLOB` respectively, and from that point will no longer have a character set, or be affected by future  
`CONVERT TO CHARACTER SET`  
 statements.

To avoid data type changes resulting from

```
CONVERT TO CHARACTER SET
, use
```

```
MODIFY
```

on the individual columns instead. For example:

```
ALTER TABLE table_name MODIFY ascii_text_column TEXT CHARACTER SET utf8;
ALTER TABLE table_name MODIFY ascii_varchar_column VARCHAR(M) CHARACTER SET utf8;
```

## Column Level

Character sets and collations can also be specified for columns that are character types `CHAR`, `TEXT` or `VARCHAR`. The `CREATE TABLE` and `ALTER TABLE` statements support optional character set and collation clauses for this purpose - unlike those at the table level, the column level definitions are standard SQL.

```
CREATE TABLE european_names (
    croatian_names VARCHAR(40) COLLATE 'cp1250_croatian_ci',
    greek_names VARCHAR(40) CHARACTER SET 'greek');
```

If neither collation nor character set is provided, the table default is used. If only the character set is specified, that character set's default collation is used, while if only the collation is specified, the associated character set is used.

When using [ALTER TABLE](#) to change a column's character set, you need to ensure the character sets are compatible with your data. MariaDB will map the data as best it can, but it's possible to lose data if care is not taken.

The [SHOW CREATE TABLE](#) statement or INFORMATION SCHEMA database can be used to determine column character sets and collations.

```
SHOW CREATE TABLE european_names\nG
*****
1. row *****

Table: european_names
Create Table: CREATE TABLE `european_names` (
`croatian_names` varchar(40) CHARACTER SET cp1250 COLLATE cp1250_croatian_ci DEFAULT NULL,
`greek_names` varchar(40) CHARACTER SET greek DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_danish_ci
```

```
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME LIKE 'european%'\nG
*****
1. row *****

TABLE_CATALOG: def
TABLE_SCHEMA: danish_names
TABLE_NAME: european_names
COLUMN_NAME: croatian_names
ORDINAL_POSITION: 1
COLUMN_DEFAULT: NULL
IS_NULLABLE: YES
DATA_TYPE: varchar
CHARACTER_MAXIMUM_LENGTH: 40
CHARACTER_OCTET_LENGTH: 40
NUMERIC_PRECISION: NULL
NUMERIC_SCALE: NULL
DATETIME_PRECISION: NULL
CHARACTER_SET_NAME: cp1250
COLLATION_NAME: cp1250_croatian_ci
COLUMN_TYPE: varchar(40)
COLUMN_KEY:
EXTRA:
PRIVILEGES: select,insert,update,references
COLUMN_COMMENT:
*****
2. row *****

TABLE_CATALOG: def
TABLE_SCHEMA: danish_names
TABLE_NAME: european_names
COLUMN_NAME: greek_names
ORDINAL_POSITION: 2
COLUMN_DEFAULT: NULL
IS_NULLABLE: YES
DATA_TYPE: varchar
CHARACTER_MAXIMUM_LENGTH: 40
CHARACTER_OCTET_LENGTH: 40
NUMERIC_PRECISION: NULL
NUMERIC_SCALE: NULL
DATETIME_PRECISION: NULL
CHARACTER_SET_NAME: greek
COLLATION_NAME: greek_general_ci
COLUMN_TYPE: varchar(40)
COLUMN_KEY:
EXTRA:
PRIVILEGES: select,insert,update,references
COLUMN_COMMENT:
```

## Filenames

Since [MariaDB 5.1](#), the `character_set_filesystem` system variable has controlled interpretation of file names that are given as literal strings. This affects the following statements and functions:

- [SELECT INTO DUMPFILE](#)
- [SELECT INTO OUTFILE](#)
- [LOAD DATA INFILE](#)

- [LOAD XML](#)
- [LOAD\\_FILE\(\)](#)

## Literals

By default, the character set and collation used for literals is determined by the [character\\_set\\_connection](#) and [collation\\_connection](#) system variables. However, they can also be specified explicitly:

```
[_charset_name]'string' [COLLATE collation_name]
```

The character set of string literals that do not have a character set introducer is determined by the [character\\_set\\_connection](#) system variable.

This query:

```
SELECT CHARSET('a'), @@character_set_connection;
```

always returns the same character set name in both columns.

[character\\_set\\_client](#) and [character\\_set\\_connection](#) are normally (e.g. during handshake, or after a SET NAMES query) are set to equal values. However, it's possible to set to different values.

## Examples

Examples when setting @@character\_set\_client and @@character\_set\_connection to different values can be useful:

Example 1:

Suppose, we have a utf8 database with this table:

```
CREATE TABLE t1 (a VARCHAR(10)) CHARACTER SET utf8 COLLATE utf8_general_ci;
INSERT INTO t1 VALUES ('oe'),('ö');
```

Now we connect to it using "mysql.exe", which uses the DOS character set (cp850 on a West European machine), and want to fetch all records that are equal to 'ö' according to the German phonebook rules.

It's possible with the following:

```
SET @@character_set_client=cp850, @@character_set_connection=utf8;
SELECT a FROM t1 WHERE a='ö' COLLATE utf8_german2_ci;
```

This will return:

```
+----+
| a   |
+----+
| oe  |
| ö   |
+----+
```

It works as follows:

1. The client sends the query using cp850.
2. The server, when parsing the query, creates a utf8 string literal by converting 'ö' from @@character\_set\_client (cp850) to @@character\_set\_connection (utf8)
3. The server applies the collation "utf8\_german2\_ci" to this string literal.
4. The server uses utf8\_german2\_ci for comparison.

Note, if we rewrite the script like this:

```
SET NAMES cp850;
SELECT a FROM t1 WHERE a='ö' COLLATE utf8_german2_ci;
```

we'll get an error:

```
ERROR 1253 (42000): COLLATION 'utf8_german2_ci' is not valid for CHARACTER SET 'cp850'
```

because:

- on step #2, the literal is not converted to utf8 any more and is created using cp850.
- on step #3, the server fails to apply utf8\_german2\_ci to an cp850 string literal.

Example 2:

Suppose we have a utf8 database and use "mysql.exe" from a West European machine again.

We can do this:

```
SET @@character_set_client=cp850, @@character_set_connection=utf8;
CREATE TABLE t2 AS SELECT 'ö';
```

It will create a table with a column of the type

VARCHAR(1) CHARACTER SET utf8

Note, if we rewrite the query like this:

```
SET NAMES cp850;
CREATE TABLE t2 AS SELECT 'ö';
```

It will create a table with a column of the type

VARCHAR(1) CHARACTER SET cp850  
, which is probably not a good idea.

## N

Also,

N  
or  
n

can be used as prefix to convert a literal into the National Character set (which in MariaDB is always utf8).

For example:

```
SELECT _latin2 'Müller';
+-----+
| MÃ¶ller   |
+-----+
| MÃ¶ller   |
+-----+
```

```
SELECT CHARSET(N'a string');
+-----+
| CHARSET(N'a string') |
+-----+
| utf8                |
+-----+
```

```
SELECT 'Mueller' = 'Müller' COLLATE 'latin1_german2_ci';
+-----+
| 'Mueller' = 'Müller' COLLATE 'latin1_german2_ci' |
+-----+
|                      1 |
+-----+
```

## Stored Programs and Views

The literals which occur in stored programs and views, by default, use the character set and collation which was specified by the `character_set_connection` and `collation_connection` system variables when the stored program was created. These values can be seen using the SHOW CREATE statements. To change the character sets used for literals in an existing stored program, it is necessary to drop and recreate the stored program.

For stored routines parameters and return values, a character set and a collation can be specified via the CHARACTER SET and COLLATE clauses. Before 5.5, specifying a collation was not supported.

The following example shows that the character set and collation are determined at the time of creation:

```

SET @@local.character_set_connection='latin1';

DELIMITER ||
CREATE PROCEDURE `test`.`x`()
BEGIN
  SELECT CHARSET('x');
END;
||
Query OK, 0 rows affected (0.00 sec)

DELIMITER ;
SET @@local.character_set_connection='utf8';

CALL `test`.`x`();
+-----+
| CHARSET('x') |
+-----+
| latin1        |
+-----+

```

The following example shows how to specify a function parameters character set and collation:

```

CREATE FUNCTION `test`.`y`(`str` TEXT CHARACTER SET utf8 COLLATE utf8_bin)
  RETURNS TEXT CHARACTER SET latin1 COLLATE latin1_bin
BEGIN
  SET @param_coll = COLLATION(`str`);
  RETURN `str`;
END;

-- return value's collation:
SELECT COLLATION(`test`.`y`('Hello, planet!'));
+-----+
| COLLATION(`test`.`y`('Hello, planet!')) |
+-----+
| latin1_bin                                |
+-----+

-- parameter's collation:
SELECT @param_coll;
+-----+
| @param_coll | 
+-----+
| utf8_bin   | 
+-----+

```

## Illegal Collation Mix

MariaDB 10.1.28 - 10.1.29

In MariaDB 10.1.28 , you may encounter Error 1267 when performing comparison operations in views on tables that use binary constants. For instance,

```

CREATE TABLE test.t1 (
  a TEXT CHARACTER SET gbk
) ENGINE=InnoDB
CHARSET=latin1
COLLATE=latin1_general_ci;

INSERT INTO t1 VALUES ('user_a');

CREATE VIEW v1 AS
SELECT a <> 0xEE5D FROM t1;

SELECT * FROM v1;
Error 1267 (HY000): Illegal mix of collations (gbk_chinese_ci, IMPLICIT)
and (latin_swedish_ci, COERCIBLE) for operation

```

When the view query is written to file, MariaDB converts the binary character into a string literal, which causes it to be misinterpreted when you execute the

```
SELECT
```

statement. If you encounter this issue, set the character set in the view to force it to the value you want.

MariaDB throws this error due to a bug that was fixed in [MariaDB 10.1.29](#) . Later releases do not throw errors in this situation.

## Example: Changing the Default Character Set To UTF-8

To change the default character set from latin1 to UTF-8, the following settings should be specified in the my.cnf configuration file.

```
[mysql]
...
default-character-set=utf8mb4
...
[mysqld]
...
collation-server = utf8mb4_unicode_ci
init-connect='SET NAMES utf8mb4'
character-set-server = utf8mb4
...
```

Note that the

```
default-character-set
option is a client option, not a server option.
```

## See Also

- [String literals](#)
- [CAST\(\)](#)
- [CONVERT\(\)](#)

## 4.2.4 Unicode

Unicode is a standard for encoding text across multiple writing systems. MariaDB supports a number of [character sets](#) for storing Unicode data:

Character Set	Description
ucs2	UCS-2, each character is represented by a 2-byte code with the most significant byte first. Fixed-length 16-bit encoding.
utf8	Until <a href="#">MariaDB 10.5</a> , this was a UTF-8 encoding using one to three bytes per character. Basic Latin letters, numbers and punctuation use one byte. European and Middle East letters mostly fit into 2 bytes. Korean, Chinese, and Japanese ideographs use 3-bytes. No supplementary characters are stored. From <a href="#">MariaDB 10.6</a> , utf8 is an alias for utf8mb3 , but this can changed to ut8mb4 by changing the default value of the <code>old_mode</code> system variable.
utf8mb3	UTF-8 encoding using one to three bytes per character. Basic Latin letters, numbers and punctuation use one byte. European and Middle East letters mostly fit into 2 bytes. Korean, Chinese, and Japanese ideographs use 3-bytes. No supplementary characters are stored. Until <a href="#">MariaDB 10.5</a> , this was an alias for utf8 . From <a href="#">MariaDB 10.6</a> , utf8 is by default an alias for utf8mb3 , but this can changed to ut8mb4 by changing the default value of the <code>old_mode</code> system variable.
utf8mb4	UTF-8 encoding the same as utf8mb3 but which stores supplementary characters in four bytes.
utf16	UTF-16, same as ucs2, but stores supplementary characters in 32 bits. 16 or 32-bits.
utf32	UTF-32, fixed-length 32-bit encoding.

## 4.2.5 SHOW CHARACTER SET

## 4.2.6 SHOW COLLATION

## Syntax

```
SHOW COLLATION  
[LIKE 'pattern' | WHERE expr]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

The output from

```
SHOW COLLATION  
includes all available collations. The  
LIKE  
clause, if present on its own, indicates which collation names to match. The  
WHERE  
and  
LIKE  
clauses can be given to select rows using more general conditions, as discussed in Extended SHOW.
```

The same information can be queried from the [Information Schema COLLATIONS](#) table.

See [Setting Character Sets and Collations](#) for details on specifying the collation at the server, database, table and column levels.

## Examples

```
SHOW COLLATION LIKE 'latin1%';  
+-----+-----+-----+-----+-----+  
| Collation      | Charset | Id | Default | Compiled | Sortlen |  
+-----+-----+-----+-----+-----+  
| latin1_german1_ci | latin1   | 5  |       | Yes     |      1 |  
| latin1_swedish_ci | latin1   | 8  | Yes   | Yes    |      1 |  
| latin1_danish_ci | latin1   | 15 |       | Yes    |      1 |  
| latin1_german2_ci | latin1   | 31 |       | Yes    |      2 |  
| latin1_bin        | latin1   | 47 |       | Yes    |      1 |  
| latin1_general_ci | latin1   | 48 |       | Yes    |      1 |  
| latin1_general_cs | latin1   | 49 |       | Yes    |      1 |  
| latin1_spanish_ci | latin1   | 94 |       | Yes    |      1 |  
+-----+-----+-----+-----+-----+
```

```
SHOW COLLATION WHERE Sortlen LIKE '8' AND Charset LIKE 'utf8';  
+-----+-----+-----+-----+-----+  
| Collation      | Charset | Id | Default | Compiled | Sortlen |  
+-----+-----+-----+-----+-----+  
| utf8_unicode_ci | utf8    | 192 |       | Yes    |      8 |  
| utf8_icelandic_ci | utf8    | 193 |       | Yes    |      8 |  
| utf8_latvian_ci | utf8    | 194 |       | Yes    |      8 |  
| utf8_romanian_ci | utf8    | 195 |       | Yes    |      8 |  
| utf8_slovenian_ci | utf8    | 196 |       | Yes    |      8 |  
| utf8_polish_ci | utf8    | 197 |       | Yes    |      8 |  
| utf8_estonian_ci | utf8    | 198 |       | Yes    |      8 |  
| utf8_spanish_ci | utf8    | 199 |       | Yes    |      8 |  
| utf8_swedish_ci | utf8    | 200 |       | Yes    |      8 |  
| utf8_turkish_ci | utf8    | 201 |       | Yes    |      8 |  
| utf8_czech_ci | utf8    | 202 |       | Yes    |      8 |  
| utf8_danish_ci | utf8    | 203 |       | Yes    |      8 |  
| utf8_lithuanian_ci | utf8    | 204 |       | Yes    |      8 |  
| utf8_slovak_ci | utf8    | 205 |       | Yes    |      8 |  
| utf8_spanish2_ci | utf8    | 206 |       | Yes    |      8 |  
| utf8_roman_ci | utf8    | 207 |       | Yes    |      8 |  
| utf8_persian_ci | utf8    | 208 |       | Yes    |      8 |  
| utf8_esperanto_ci | utf8    | 209 |       | Yes    |      8 |  
| utf8_hungarian_ci | utf8    | 210 |       | Yes    |      8 |  
| utf8_sinhala_ci | utf8    | 211 |       | Yes    |      8 |  
| utf8_croatian_ci | utf8    | 213 |       | Yes    |      8 |  
+-----+-----+-----+-----+-----+
```

## See Also

- Supported Character Sets and Collations
- Setting Character Sets and Collations
- Information Schema COLLATIONS

## 4.2.7 Information Schema CHARACTER\_SETS Table

## 4.2.8 Information Schema COLLATIONS Table

## 4.2.9 Internationalization and Localization



### Server Locale

*Server locale.*



### Time Zones

*MariaDB time zones.*



### Setting the Language for Error Messages

*Specifying the language for the server error messages.*



### Coordinated Universal Time

*Coordinated Universal Time.*



### Locales Plugin

*List compiled-in locales.*



### mysql\_tzinfo\_to\_sql

*Load time zones to the mysql time zone tables.*

## 4.2.10 SET CHARACTER SET

## 4.2.11 SET NAMES

## 4.3 Storage Engines

### 4.3.1 Choosing the Right Storage Engine

#### Contents

1. Topic List
  1. General Purpose
  2. Scaling, Partitioning
  3. Compression / Archive
  4. Connecting to Other Data Sources
  5. Search Optimized
  6. Cache, Read-only
  7. Other Specialized Storage Engines
2. Alphabetical List

A high-level overview of the main reasons for choosing a particular storage engine:

#### Topic List

##### General Purpose

- InnoDB is a good general transaction storage engine, and, from MariaDB 10.2, the best choice in most cases. It is the default storage engine from MariaDB 10.2. For earlier releases, XtraDB was a performance enhanced fork of InnoDB and is usually preferred.
- XtraDB is the best choice in MariaDB 10.1 and earlier in the majority of cases. It is a performance-enhanced fork of InnoDB and is MariaDB's default engine until MariaDB 10.1.
- Aria, MariaDB's more modern improvement on MyISAM, has a small footprint and allows for easy copying between systems.
- MyISAM has a small footprint and allows for easy copying between systems. MyISAM is MySQL's oldest storage engine. There is usually little reason to use it except for legacy purposes. Aria is MariaDB's more modern improvement.

##### Scaling, Partitioning

When you want to split your database load on several servers or optimize for scaling. We also suggest looking at [Galera](#), a synchronous multi-master cluster.

- [Spider](#) uses partitioning to provide data sharding through multiple servers.
- [ColumnStore](#) utilizes a massively parallel distributed data architecture and is designed for big data scaling to process petabytes of data.
- The [MERGE](#) storage engine is a collection of identical [MyISAM](#) tables that can be used as one. "Identical" means that all tables have identical column and index information.
- [TokuDB](#) is a transactional storage engine which is optimized for workloads that do not fit in memory, and provides a good compression ratio. TokuDB has been deprecated by its upstream developers, and is disabled in [MariaDB 10.5](#), and removed in [MariaDB 10.6](#).

## Compression / Archive

- [MyRocks](#) enables greater compression than InnoDB, as well as less write amplification giving better endurance of flash storage and improving overall throughput.
- The [Archive](#) storage engine is, unsurprisingly, best used for archiving.
- [TokuDB](#) is a transactional storage engine which is optimized for workloads that do not fit in memory, and provides a good compression ratio. TokuDB has been deprecated by its upstream developers, and is disabled in [MariaDB 10.5](#), and removed in [MariaDB 10.6](#).

## Connecting to Other Data Sources

When you want to use data not stored in a MariaDB database.

- [CONNECT](#) allows access to different kinds of text files and remote resources as if they were regular MariaDB tables.
- The [CSV](#) storage engine can read and append to files stored in CSV (comma-separated-values) format. However, since [MariaDB 10.0](#), CONNECT is a better choice and is more flexibly able to read and write such files.
- [FederatedX](#) uses libmysql to talk to the data source, the data source being a remote RDBMS. Currently, since FederatedX only uses libmysql, it can only talk to another MySQL RDBMS.
- [CassandraSE](#) is a storage engine allowing access to an older version of Apache Cassandra NoSQL DBMS. It was relatively experimental, is no longer being actively developed and has been removed in [MariaDB 10.6](#).

## Search Optimized

Search engines optimized for search.

- [SphinxSE](#) is used as a proxy to run statements on a remote Sphinx database server (mainly useful for advanced fulltext searches).
- [Mroonga](#) provides fast CJK-ready full text searching using column store.

## Cache, Read-only

- [MEMORY](#) does not write data on-disk (all rows are lost on crash) and is best-used for read-only caches of data from other tables, or for temporary work areas. With the default [InnoDB](#) and other storage engines having good caching, there is less need for this engine than in the past.

## Other Specialized Storage Engines

- [S3 Storage Engine](#) is a read-only storage engine that stores its data in Amazon S3.
- [Sequence](#) allows the creation of ascending or descending sequences of numbers (positive integers) with a given starting value, ending value and increment, creating virtual, ephemeral tables automatically when you need them.
- The [BLACKHOLE](#) storage engine accepts data but does not store it and always returns an empty result. This can be useful in [replication](#) environments, for example, if you want to run complex filtering rules on a slave without incurring any overhead on a master.
- [OQGRAPH](#) allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).

## Alphabetical List

- The [Archive](#) storage engine is, unsurprisingly, best used for archiving.
- [Aria](#), MariaDB's more modern improvement on MyISAM, has a small footprint and allows for easy copy between systems.
- The [BLACKHOLE](#) storage engine accepts data but does not store it and always returns an empty result. This can be useful in [replication](#) environments, for example, if you want to run complex filtering rules on a slave without incurring any overhead on a master.
- [CassandraSE](#) is a storage engine allowing access to an older version of Apache Cassandra NoSQL DBMS. It was relatively experimental, is no longer being actively developed and has been removed in [MariaDB 10.6](#).
- [ColumnStore](#) utilizes a massively parallel distributed data architecture and is designed for big data scaling to process petabytes of data.
- [CONNECT](#) allows access to different kinds of text files and remote resources as if they were regular MariaDB tables.
- The [CSV](#) storage engine can read and append to files stored in CSV (comma-separated-values) format. However, since [MariaDB 10.0](#), CONNECT is a better choice and is more flexibly able to read and write such files.
- [FederatedX](#) uses libmysql to talk to the data source, the data source being a remote RDBMS. Currently, since FederatedX only uses libmysql, it can only talk to another MySQL RDBMS.
- [InnoDB](#) is a good general transaction storage engine, and, from [MariaDB 10.2](#), the best choice in most cases. It is the default storage engine from [MariaDB 10.2](#). For earlier releases, XtraDB was a performance enhanced fork of InnoDB and is usually preferred.
- The [MERGE](#) storage engine is a collection of identical MyISAM tables that can be used as one. "Identical" means that all tables have identical column and index information.
- [MEMORY](#) does not write data on-disk (all rows are lost on crash) and is best-used for read-only caches of data from other tables, or for temporary work areas. With the default [InnoDB](#) and other storage engines having good caching, there is less need for this engine than in the past.

- [Moonga](#) provides fast CJK-ready full text searching using column store.
- [MyISAM](#) has a small footprint and allows for easy copying between systems. MyISAM is MySQL's oldest storage engine. There is usually little reason to use it except for legacy purposes. Aria is MariaDB's more modern improvement.
- [MyRocks](#) enables greater compression than InnoDB, as well as less write amplification giving better endurance of flash storage and improving overall throughput.
- [OQGRAPH](#) allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).
- [S3 Storage Engine](#) is a read-only storage engine that stores its data in Amazon S3.
- [Sequence](#) allows the creation of ascending or descending sequences of numbers (positive integers) with a given starting value, ending value and increment, creating virtual, ephemeral tables automatically when you need them.
- [SphinxSE](#) is used as a proxy to run statements on a remote Sphinx database server (mainly useful for advanced fulltext searches).
- [Spider](#) uses partitioning to provide data sharding through multiple servers.
- [TokuDB](#) is a transactional storage engine which is optimized for workloads that do not fit in memory, and provides a good compression ratio. TokuDB has been deprecated by its upstream developers, and is disabled in [MariaDB 10.5](#), and removed in [MariaDB 10.6](#)
- [XtraDB](#) is the best choice in [MariaDB 10.1](#) and earlier in the majority of cases. It is a performance-enhanced fork of InnoDB and is MariaDB's default engine until [MariaDB 10.1](#).

## 4.3.2 InnoDB

### 4.3.2.1 InnoDB Versions

#### MariaDB starting with 10.3.7

In [MariaDB 10.3.7](#) and later, the InnoDB implementation has diverged substantially from the InnoDB in MySQL. Therefore, in these versions, the InnoDB version is no longer associated with a MySQL release version.

#### MariaDB starting with 10.2

In [MariaDB 10.2](#) and later, the default InnoDB implementation is based on InnoDB from MySQL 5.7. See [Why MariaDB uses InnoDB instead of XtraDB from MariaDB 10.2](#) for more information.

#### MariaDB until 10.1

In [MariaDB 10.1](#) and before, the default InnoDB implementation is based on Percona's XtraDB. XtraDB is a performance enhanced fork of InnoDB. For compatibility reasons, the [system variables](#) still retain their original

`innodb`

prefixes. If the documentation says that something applies to InnoDB, then it usually also applies to the XtraDB fork, unless explicitly stated otherwise. In these versions, it is still possible to use InnoDB instead of XtraDB. See [Using InnoDB instead of XtraDB](#) for more information.

## Divergences

Some examples of divergences between MariaDB's InnoDB and MySQL's InnoDB are:

- [MariaDB 10.1](#) (which is based on MySQL 5.6) included encryption and variable-size page compression before MySQL 5.7 introduced them.
- [MariaDB 10.2](#) (based on MySQL 5.7) introduced persistent AUTO\_INCREMENT ([MDEV-6076](#)) in a GA release before MySQL 8.0.
- [MariaDB 10.3](#) (based on MySQL 5.7) introduced instant ADD COLUMN ([MDEV-11369](#)) before MySQL.

## InnoDB Versions Included in MariaDB Releases

### MariaDB 10.3

InnoDB Version	Introduced
No longer reported	<a href="#">MariaDB 10.3.7</a>
InnoDB 5.7.20	<a href="#">MariaDB 10.3.3</a>
InnoDB 5.7.19	<a href="#">MariaDB 10.3.1</a>
InnoDB 5.7.14	<a href="#">MariaDB 10.3.0</a>

### MariaDB 10.2

InnoDB Version	Introduced
InnoDB 5.7.29	<a href="#">MariaDB 10.2.33</a>
InnoDB 5.7.23	<a href="#">MariaDB 10.2.17</a>

InnoDB 5.7.22	<a href="#">MariaDB 10.2.15</a>
InnoDB 5.7.21	<a href="#">MariaDB 10.2.13</a>
InnoDB 5.7.20	<a href="#">MariaDB 10.2.10</a>
InnoDB 5.7.19	<a href="#">MariaDB 10.2.8</a>
InnoDB 5.7.18	<a href="#">MariaDB 10.2.7</a>
InnoDB 5.7.14	<a href="#">MariaDB 10.2.2</a>

## MariaDB 10.1

InnoDB Version	Introduced
InnoDB 5.6.49	<a href="#">MariaDB 10.1.46</a>
InnoDB 5.6.47	<a href="#">MariaDB 10.1.44</a>
InnoDB 5.6.44	<a href="#">MariaDB 10.1.39</a>
InnoDB 5.6.42	<a href="#">MariaDB 10.1.37</a>
InnoDB 5.6.39	<a href="#">MariaDB 10.1.31</a>
InnoDB 5.6.37	<a href="#">MariaDB 10.1.26</a>
InnoDB 5.6.36	<a href="#">MariaDB 10.1.24</a>
InnoDB 5.6.35	<a href="#">MariaDB 10.1.21</a>
InnoDB 5.6.33	<a href="#">MariaDB 10.1.18</a>
InnoDB 5.6.32	<a href="#">MariaDB 10.1.17</a>
InnoDB 5.6.31	<a href="#">MariaDB 10.1.16</a>
InnoDB 5.6.30	<a href="#">MariaDB 10.1.14</a>
InnoDB 5.6.29	<a href="#">MariaDB 10.1.12</a>

## MariaDB 10.0

InnoDB Version	Introduced
InnoDB 5.6.43	<a href="#">MariaDB 10.0.38</a>
InnoDB 5.6.42	<a href="#">MariaDB 10.0.37</a>
InnoDB 5.6.40	<a href="#">MariaDB 10.0.35</a>
InnoDB 5.6.39	<a href="#">MariaDB 10.0.34</a>
InnoDB 5.6.38	<a href="#">MariaDB 10.0.33</a>
InnoDB 5.6.37	<a href="#">MariaDB 10.0.32</a>
InnoDB 5.6.36	<a href="#">MariaDB 10.0.31</a>
InnoDB 5.6.35	<a href="#">MariaDB 10.0.29</a>
InnoDB 5.6.33	<a href="#">MariaDB 10.0.28</a>
InnoDB 5.6.32	<a href="#">MariaDB 10.0.27</a>
InnoDB 5.6.31	<a href="#">MariaDB 10.0.26</a>
InnoDB 5.6.30	<a href="#">MariaDB 10.0.25</a>
InnoDB 5.6.29	<a href="#">MariaDB 10.0.24</a>
InnoDB 5.6.28	<a href="#">MariaDB 10.0.23</a>
InnoDB 5.6.27	<a href="#">MariaDB 10.0.22</a>
InnoDB 5.6.26	<a href="#">MariaDB 10.0.21</a>
InnoDB 5.6.25	<a href="#">MariaDB 10.0.20</a>
InnoDB 5.6.24	<a href="#">MariaDB 10.0.18</a>
InnoDB 5.6.23	<a href="#">MariaDB 10.0.17</a>
InnoDB 5.6.22	<a href="#">MariaDB 10.0.16</a>

InnoDB 5.6.21	MariaDB 10.0.15
InnoDB 5.6.20	MariaDB 10.0.14
InnoDB 5.6.19	MariaDB 10.0.13
InnoDB 5.6.17	MariaDB 10.0.11
InnoDB 5.6.15	MariaDB 10.0.9
InnoDB 5.6.14	MariaDB 10.0.8

## See Also

- [Why MariaDB uses InnoDB instead of XtraDB from MariaDB 10.2](#)
- [XtraDB Versions](#)

### 4.3.2.2 InnoDB Limitations

#### Contents

1. [Limitations on Schema](#)
2. [Limitations on Size](#)
  1. [Page Sizes](#)
  2. [Large Prefix Size](#)
3. [Limitations on Tables](#)
  1. [Table Analysis](#)
  2. [Table Status](#)
  3. [Auto-incrementing Columns](#)
4. [Transactions and Locks](#)

The [InnoDB storage engine](#) has the following limitations.

## Limitations on Schema

- InnoDB tables can have a maximum of 1,017 columns. This includes [virtual generated columns](#).
- InnoDB tables can have a maximum of 64 secondary indexes.
- A multicolumn index on InnoDB can use a maximum of 16 columns. If you attempt to create a multicolumn index that uses more than 16 columns, MariaDB returns an Error 1070.

## Limitations on Size

- With the exception of variable-length columns (that is, [VARBINARY](#), [VARCHAR](#), [BLOB](#) and [TEXT](#)), rows in InnoDB have a maximum length of roughly half the page size for 4KB, 8KB, 16KB and 32KB page sizes.
- The maximum size for [BLOB](#) and [TEXT](#) columns is 4GB. This also applies to [LONGBLOB](#) and [LONGTEXT](#).
- MariaDB imposes a row-size limit of 65,535 bytes for the combined sizes of all columns. If the table contains [BLOB](#) or [TEXT](#) columns, these only count for 9 - 12 bytes in this calculation, given that their content is stored separately.
- 32-bit operating systems have a maximum file size limit of 2GB. When working with large tables using this architecture, configure InnoDB to use smaller data files.
- The maximum size for the combined InnoDB log files is 512GB.
- With tablespaces, the minimum size is 10MB, the maximum varies depending on the InnoDB Page Size.

InnoDB Page Size	Maximum Tablespace Size
4KB	16TB
8KB	32TB
16KB	64TB
32KB	128TB
64KB	256TB

## Page Sizes

Using the [innodb\\_page\\_size](#) system variable, you can configure the size in bytes for InnoDB pages. Pages default to 16KB. There are certain limitations on how you use this variable.

- MariaDB instances using one page size cannot use data files or log files from an instance using a different page size.
- When using a Page Size of 4KB or 8KB, the maximum index key length is lowered proportionately.

InnoDB Page Size	Index Key Length
------------------	------------------

4KB	768B
8KB	1536B
16KB	3072B

## Large Prefix Size

Until MariaDB 10.3.1 , the `innodb_large_prefix` system variable enabled large prefix sizes. That is, when enabled (the default from MariaDB 10.2 ), InnoDB uses 3072B index key prefixes for

DYNAMIC

and

COMPRESSED

row formats. When disabled, it uses 787B key prefixes for tables of any row format. Using an index key that exceeds this limit throws an error.

From MariaDB 10.3.1 , InnoDB always uses large index key prefixes.

## Limitations on Tables

InnoDB has the following table-specific limitations.

- When you issue a `DELETE` statement, InnoDB doesn't regenerate the table, rather it deletes each row from the table one by one.
- When running MariaDB on Windows, InnoDB stores databases and tables in lowercase. When moving databases and tables in a binary format from Windows to a Unix-like system or from a Unix system to Windows, you need to rename these to use lowercase.
- When using cascading `foreign keys` , operations in the cascade don't activate triggers.

## Table Analysis

MariaDB supports the use of the `ANALYZE TABLE` SQL statement to analyze and store table key distribution. When MariaDB executes this statement, it calculates index cardinality through random dives on index trees. This makes it fast, but not always accurate as it does not check all rows. The data is only an estimate and repeated executions of this statement may return different results.

In situations where accurate data from `ANALYZE TABLE` statements is important, enable the `innodb_stats_persistent` system variable. Additionally, you can use the `innodb_stats_transient_sample_pages` system variable to change the number of random dives it performs.

When running `ANALYZE TABLE` twice on a table in which statements or transactions are running, MariaDB blocks the second `ANALYZE TABLE` until the statement or transaction is complete. This occurs because the statement or transaction blocks the second `ANALYZE TABLE` statement from reloading the table definition, which it must do since the old one was marked as obsolete after the first statement.

## Table Status

Similar to the `ANALYZE TABLE` statement, `SHOW TABLE STATUS` statements do not provide accurate statistics for InnoDB, except for the physical table size.

The InnoDB storage engine does not maintain internal row counts. Transactions isolate writes, which means that concurrent transactions will not have the same row counts.

## Auto-incrementing Columns

- When defining an index on an auto-incrementing column, it must be defined in a way that allows the equivalent of
 

```
SELECT MAX(col)
      lookups on the table.
```
- Restarting MariaDB may cause InnoDB to reuse old auto-increment values, such as in the case of a transaction that was rolled back.
- When auto-incrementing columns run out of values, `INSERT` statements generate duplicate-key errors.

## Transactions and Locks

- You can modify data on a maximum of 96 \* 1023 concurrent transactions that generate undo records.
- Of the 128 rollback segments, InnoDB assigns 32 to non-redo logs for transactions that modify temporary tables and related objects, reducing the maximum number of concurrent data-modifying transactions to 96,000, from 128,000.
- The limit is 32,000 concurrent transactions when all data-modifying transactions also modify temporary tables.
- Issuing a `LOCK TABLES` statement sets two locks on each table when the `innodb_table_locks` system variable is enabled (the default).
- When you commit or roll back a transaction, any locks set in the transaction are released. You don't need to issue `LOCK TABLES` statements when the `autocommit` variable is enabled, as InnoDB would immediately release the table locks.

### 4.3.2.3 InnoDB Troubleshooting

#### 4.3.2.3.1 InnoDB Troubleshooting Overview

## Contents

### 1. See Also

As with most errors, first take a look at the contents of the [MariaDB error log](#). If dealing with a deadlock, setting the `innodb_print_all_deadlocks` option (off by default) will output details of all deadlocks to the error log.

It can also help to enable the various [InnoDB Monitors](#) relating to the problem you are experiencing. There are four types: the standard InnoDB monitor, the InnoDB Lock Monitor, InnoDB Tablespace Monitor and the InnoDB Table Monitor.

Running `CHECK TABLE` will help determine whether there are errors in the table.

For problems with the InnoDB Data Dictionary, see [InnoDB Data Dictionary Troubleshooting](#).

## See Also

- [InnoDB Data Dictionary Troubleshooting](#)
- [InnoDB Recovery Modes](#)
- [Error Codes](#)

### 4.3.2.3.2 InnoDB Data Dictionary Troubleshooting

## Contents

### 1. Can't Open File 2. Removing Orphan Intermediate Tables 3. See Also

## Can't Open File

If InnoDB returns something like the following error:

```
ERROR 1016: Can't open file: 'x.ibd'. (errno: 1)
```

it may be that an orphan

`.frm`

file exists. Something like the following may also appear in the [error log](#):

```
InnoDB: Cannot find table test/x from the internal data dictionary
InnoDB: of InnoDB though the .frm file for the table exists. Maybe you
InnoDB: have deleted and recreated InnoDB data files but have forgotten
InnoDB: to delete the corresponding .frm files of InnoDB tables?
```

If this is the case, as the text describes, delete the orphan

`.frm`

file on the filesystem.

## Removing Orphan Intermediate Tables

An orphan intermediate table may prevent you from dropping the tablespace even if it is otherwise empty, and generally takes up unnecessary space.

It may come about if MariaDB exits in the middle of an `ALTER TABLE ... ALGORITHM=INPLACE` operation. They will be listed in the `INFORMATION_SCHEMA.INNODB_SYS_TABLES` table, and always start with an

```
#sql-ib
prefix. The accompanying
.frm
file also begins with
#sql-
, but has a different name.
```

To identify orphan tables, run:

```
SELECT * FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE NAME LIKE '%#sql%';
```

When `innodb_file_per_table` is set, the

```
#sql-* .ibd
file will also be visible in the database directory.
```

To remove an orphan intermediate table:

- Rename the  
`#sql-* .frm`  
file (in the database directory) to match the base name of the orphan intermediate table, for example:

```
mv #sql-36ab_2.frm #sql-ib87-856498050.frm
```

- Drop the table, for example:

```
DROP TABLE `#mysql150##sql-ib87-856498050`;
```

## See Also

- [InnoDB Troubleshooting Overview](#)

### 4.3.2.3.3 InnoDB Recovery Modes

The InnoDB recovery mode is a mode used for recovering from emergency situations. You should ensure you have a backup of your database before making changes in case you need to restore it. The `innodb_force_recovery` server system variable sets the recovery mode. A mode of 0 is normal use, while the higher the mode, the more stringent the restrictions. Higher modes incorporate all limitations of the lower modes.

The recovery mode should never be set to a value other than zero except in an emergency situation.

Generally, it is best to start with a recovery mode of 1, and increase in single increments if needs be. With a recovery mode < 4, only corrupted pages should be lost. With 4, secondary indexes could be corrupted. With 5, results could be inconsistent and secondary indexes could be corrupted (even if they were not with 4). A value of 6 leaves pages in an obsolete state, which might cause more corruption.

Until [MariaDB 10.2.7](#), mode

0

was the only mode permitting changes to the data. From [MariaDB 10.2.7](#), write transactions are permitted with mode

3

or less.

To recover the tables, you can execute `SELECTs` to dump data, and `DROP TABLE` (when write transactions are permitted) to remove corrupted tables.

The following modes are available:

## Recovery Modes

Recovery mode behaviour differs per version (server/storage/innobase/include/srv0srv.h)

[MariaDB 10.4](#) and before:

Mode	Description
0	The default mode while InnoDB is running normally. Until <a href="#">MariaDB 10.2.7</a> , it was the only mode permitting changes to the data. From <a href="#">MariaDB 10.2.7</a> , write transactions are permitted with <code>innodb_force_recovery&lt;=3</code> .
1	( <code>SRV_FORCE_IGNORE_CORRUPT</code> ) allows the the server to keep running even if corrupt pages are detected. It does so by making redo log based recovery ignore certain errors, such as missing data files or corrupted data pages. Any redo log for affected files or pages will be skipped. You can facilitate dumping tables by getting the <code>SELECT * FROM table_name</code> statement to jump over corrupt indexes and pages.
2	( <code>SRV_FORCE_NO_BACKGROUND</code> ) stops the master thread from running, preventing a crash that occurs during a purge. No purge will be performed, so the undo logs will keep growing.
3	( <code>SRV_FORCE_NO_TRX_UNDO</code> ) does not roll back transactions after the crash recovery. Does not affect rollback of currently active transactions. Starting with <a href="#">MariaDB 10.2.7</a> , will also prevent some undo-generating background tasks from running. These tasks could hit a lock wait due to the recovered incomplete transactions whose rollback is being prevented.
4	( <code>SRV_FORCE_NO_IBUF_MERGE</code> ) does not calculate tables statistics and prevents insert buffer merges.
5	( <code>SRV_FORCE_NO_UNDO_LOG_SCAN</code> ) treats incomplete transactions as committed, and does not look at the <code>undo logs</code> when starting.
6	( <code>SRV_FORCE_NO_LOG_REDO</code> ) does not perform redo log roll-forward as part of recovery. Running queries that require indexes are likely to fail with this mode active. However, if a table dump still causes a crash, you can try using a <code>SELECT * FROM tab ORDER BY primary_key DESC</code> to dump all the data portion after the corrupted part.

From [MariaDB 10.5](#) to [MariaDB 10.6.4](#):

Mode	Description
0	The default mode while InnoDB is running normally. Write transactions are permitted with <code>innodb_force_recovery&lt;=4</code> .
1	( <code>SRV_FORCE_IGNORE_CORRUPT</code> ) allows the the server to keep running even if corrupt pages are detected. It does so by making redo log based recovery ignore certain errors, such as missing data files or corrupted data pages. Any redo log for affected files or pages will be skipped. You can facilitate dumping tables by getting the <code>SELECT * FROM table_name</code> statement to jump over corrupt indexes and pages.

2	(SRV_FORCE_NO_BACKGROUND) stops the master thread from running, preventing a crash that occurs during a purge. No purge will be performed, so the undo logs will keep growing.
3	(SRV_FORCE_NO_TRX_UNDO) does not roll back transactions after the crash recovery. Does not affect rollback of currently active transactions. Will also prevent some undo-generating background tasks from running. These tasks could hit a lock wait due to the recovered incomplete transactions whose rollback is being prevented.
4	(SRV_FORCE_NO_IBUF_MERGE) The same as 3.
5	(SRV_FORCE_NO_UNDO_LOG_SCAN) treats incomplete transactions as committed, and does not look at the <a href="#">undo logs</a> when starting.
6	(SRV_FORCE_NO_LOG_REDO) does not perform redo log roll-forward as part of recovery. Running queries that require indexes are likely to fail with this mode active. However, if a table dump still causes a crash, you can try using a SELECT * FROM tab ORDER BY primary_key DESC to dump all the data portion after the corrupted part.

From [MariaDB 10.6.5](#)

Mode	Description
0	The default mode while InnoDB is running normally. Write transactions are permitted with innodb_force_recovery<=4.
1	(SRV_FORCE_IGNORE_CORRUPT) allows the server to keep running even if corrupt pages are detected. It does so by making redo log based recovery ignore certain errors, such as missing data files or corrupted data pages. Any redo log for affected files or pages will be skipped. You can facilitate dumping tables by getting the SELECT * FROM table_name statement to jump over corrupt indexes and pages.
2	(SRV_FORCE_NO_BACKGROUND) stops the master thread from running, preventing a crash that occurs during a purge. No purge will be performed, so the undo logs will keep growing.
3	(SRV_FORCE_NO_TRX_UNDO) does not roll back DML transactions after the crash recovery. Does not affect rollback of currently active DML transactions. Will also prevent some undo-generating background tasks from running. These tasks could hit a lock wait due to the recovered incomplete transactions whose rollback is being prevented.
4	(SRV_FORCE_NO_DDL_UNDO) does not roll back transactions after the crash recovery. Does not affect rollback of currently active transactions. Will also prevent some undo-generating background tasks from running. These tasks could hit a lock wait due to the recovered incomplete transactions whose rollback is being prevented.
5	(SRV_FORCE_NO_UNDO_LOG_SCAN) treats incomplete transactions as committed, and does not look at the <a href="#">undo logs</a> when starting. Any DDL log for InnoDB tables will be essentially ignored by InnoDB, but the server will start up
6	(SRV_FORCE_NO_LOG_REDO) does not perform redo log roll-forward as part of recovery. Running queries that require indexes are likely to fail with this mode active. However, if a table dump still causes a crash, you can try using a SELECT * FROM tab ORDER BY primary_key DESC to dump all the data portion after the corrupted part.

Note also that XtraDB (<= MariaDB 10.2.6 ) by default will crash the server when it detects corrupted data in a single-table tablespace. This behaviour can be changed - see the [innodb\\_corrupt\\_table\\_action](#) system variable.

## Fixing Things

Try to set innodb\_force\_recovery to 1 and start mariadb. If that fails, try a value of "2". If a value of 2 works, then there is a chance the only corruption you have experienced is within the innodb "undo logs". If that gets mariadb started, you should be able to dump your database with mysqldump. You can verify any other issues with any tables by running "mysqlcheck --all-databases".

If you were able to successfully dump your databases, or had previously known good backups, drop your database(s) from the mariadb command line like "[DROP DATABASE](#) yourdatabase". Stop mariadb. Go to /var/lib/mysql (or wherever your mysql data directory is located) and "rm -i ib\*". Start mariadb, create the database(s) you dropped (" [CREATE DATABASE](#) yourdatabase"), and then import your most recent dumps: "mysql < mydatabasedump.sql"

### 4.3.2.3.4 Troubleshooting Row Size Too Large Errors with InnoDB

With InnoDB, users can see the following message as an error or warning:

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored
inline.
```

And they can also see the following message as an error or warning in the [error log](#) :

```
[Warning] InnoDB: Cannot add field col in table db1.tab because after adding it,
the row size is 8478 which is greater than maximum allowed size (8126) for a
record on index leaf page.
```

## Contents

1. Example of the Problem
2. Root Cause of the Problem
3. Checking Existing Tables for the Problem
4. Finding All Tables That Currently Have the Problem
5. Solving the Problem
  1. Converting the Table to the DYNAMIC Row Format
  2. Fitting More Columns on Overflow Pages
    1. Converting Some Columns to BLOB or TEXT
    2. Increasing the Length of VARBINARY Columns
    3. Increasing the Length of VARCHAR Columns
6. Working Around the Problem
  1. Refactoring the Table into Multiple Tables
  2. Refactoring Some Columns into JSON
  3. Disabling InnoDB Strict Mode

These messages indicate that the table's definition allows rows that the table's InnoDB row format can't actually store.

These messages are raised in the following cases:

- If **InnoDB strict mode** is **enabled** and if a **DDL** statement is executed that touches the table, such as

`CREATE TABLE`

or

`ALTER TABLE`

, then InnoDB will raise an **error** with this message

- If **InnoDB strict mode** is **disabled** and if a **DDL** statement is executed that touches the table, such as

`CREATE TABLE`

or

`ALTER TABLE`

, then InnoDB will raise a **warning** with this message.

- Regardless of whether **InnoDB strict mode** is enabled, if a **DML** statement is executed that attempts to write a row that the table's InnoDB row format can't store, then InnoDB will raise an **error** with this message.

## Example of the Problem

Here is an example of the problem:

```
SET GLOBAL innodb_default_row_format='dynamic';

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    col1 varchar(40) NOT NULL,
    col2 varchar(40) NOT NULL,
    col3 varchar(40) NOT NULL,
    col4 varchar(40) NOT NULL,
    col5 varchar(40) NOT NULL,
    col6 varchar(40) NOT NULL,
    col7 varchar(40) NOT NULL,
    col8 varchar(40) NOT NULL,
    col9 varchar(40) NOT NULL,
    col10 varchar(40) NOT NULL,
    col11 varchar(40) NOT NULL,
    col12 varchar(40) NOT NULL,
    col13 varchar(40) NOT NULL,
    col14 varchar(40) NOT NULL,
    col15 varchar(40) NOT NULL)
```

```
col15 varchar(40) NOT NULL,
col16 varchar(40) NOT NULL,
col17 varchar(40) NOT NULL,
col18 varchar(40) NOT NULL,
col19 varchar(40) NOT NULL,
col20 varchar(40) NOT NULL,
col21 varchar(40) NOT NULL,
col22 varchar(40) NOT NULL,
col23 varchar(40) NOT NULL,
col24 varchar(40) NOT NULL,
col25 varchar(40) NOT NULL,
col26 varchar(40) NOT NULL,
col27 varchar(40) NOT NULL,
col28 varchar(40) NOT NULL,
col29 varchar(40) NOT NULL,
col30 varchar(40) NOT NULL,
col31 varchar(40) NOT NULL,
col32 varchar(40) NOT NULL,
col33 varchar(40) NOT NULL,
col34 varchar(40) NOT NULL,
col35 varchar(40) NOT NULL,
col36 varchar(40) NOT NULL,
col37 varchar(40) NOT NULL,
col38 varchar(40) NOT NULL,
col39 varchar(40) NOT NULL,
col40 varchar(40) NOT NULL,
col41 varchar(40) NOT NULL,
col42 varchar(40) NOT NULL,
col43 varchar(40) NOT NULL,
col44 varchar(40) NOT NULL,
col45 varchar(40) NOT NULL,
col46 varchar(40) NOT NULL,
col47 varchar(40) NOT NULL,
col48 varchar(40) NOT NULL,
col49 varchar(40) NOT NULL,
col50 varchar(40) NOT NULL,
col51 varchar(40) NOT NULL,
col52 varchar(40) NOT NULL,
col53 varchar(40) NOT NULL,
col54 varchar(40) NOT NULL,
col55 varchar(40) NOT NULL,
col56 varchar(40) NOT NULL,
col57 varchar(40) NOT NULL,
col58 varchar(40) NOT NULL,
col59 varchar(40) NOT NULL,
col60 varchar(40) NOT NULL,
col61 varchar(40) NOT NULL,
col62 varchar(40) NOT NULL,
col63 varchar(40) NOT NULL,
col64 varchar(40) NOT NULL,
col65 varchar(40) NOT NULL,
col66 varchar(40) NOT NULL,
col67 varchar(40) NOT NULL,
col68 varchar(40) NOT NULL,
col69 varchar(40) NOT NULL,
col70 varchar(40) NOT NULL,
col71 varchar(40) NOT NULL,
col72 varchar(40) NOT NULL,
col73 varchar(40) NOT NULL,
col74 varchar(40) NOT NULL,
col75 varchar(40) NOT NULL,
col76 varchar(40) NOT NULL,
col77 varchar(40) NOT NULL,
col78 varchar(40) NOT NULL,
col79 varchar(40) NOT NULL,
col80 varchar(40) NOT NULL,
col81 varchar(40) NOT NULL,
col82 varchar(40) NOT NULL,
col83 varchar(40) NOT NULL,
col84 varchar(40) NOT NULL,
col85 varchar(40) NOT NULL,
col86 varchar(40) NOT NULL,
col87 varchar(40) NOT NULL,
col88 varchar(40) NOT NULL,
col89 varchar(40) NOT NULL,
col90 varchar(40) NOT NULL,
col91 varchar(40) NOT NULL,
col92 varchar(40) NOT NULL,
```

```
col93 varchar(40) NOT NULL,
col94 varchar(40) NOT NULL,
col95 varchar(40) NOT NULL,
col96 varchar(40) NOT NULL,
col97 varchar(40) NOT NULL,
col98 varchar(40) NOT NULL,
col99 varchar(40) NOT NULL,
col100 varchar(40) NOT NULL,
col101 varchar(40) NOT NULL,
col102 varchar(40) NOT NULL,
col103 varchar(40) NOT NULL,
col104 varchar(40) NOT NULL,
col105 varchar(40) NOT NULL,
col106 varchar(40) NOT NULL,
col107 varchar(40) NOT NULL,
col108 varchar(40) NOT NULL,
col109 varchar(40) NOT NULL,
col110 varchar(40) NOT NULL,
col111 varchar(40) NOT NULL,
col112 varchar(40) NOT NULL,
col113 varchar(40) NOT NULL,
col114 varchar(40) NOT NULL,
col115 varchar(40) NOT NULL,
col116 varchar(40) NOT NULL,
col117 varchar(40) NOT NULL,
col118 varchar(40) NOT NULL,
col119 varchar(40) NOT NULL,
col120 varchar(40) NOT NULL,
col121 varchar(40) NOT NULL,
col122 varchar(40) NOT NULL,
col123 varchar(40) NOT NULL,
col124 varchar(40) NOT NULL,
col125 varchar(40) NOT NULL,
col126 varchar(40) NOT NULL,
col127 varchar(40) NOT NULL,
col128 varchar(40) NOT NULL,
col129 varchar(40) NOT NULL,
col130 varchar(40) NOT NULL,
col131 varchar(40) NOT NULL,
col132 varchar(40) NOT NULL,
col133 varchar(40) NOT NULL,
col134 varchar(40) NOT NULL,
col135 varchar(40) NOT NULL,
col136 varchar(40) NOT NULL,
col137 varchar(40) NOT NULL,
col138 varchar(40) NOT NULL,
col139 varchar(40) NOT NULL,
col140 varchar(40) NOT NULL,
col141 varchar(40) NOT NULL,
col142 varchar(40) NOT NULL,
col143 varchar(40) NOT NULL,
col144 varchar(40) NOT NULL,
col145 varchar(40) NOT NULL,
col146 varchar(40) NOT NULL,
col147 varchar(40) NOT NULL,
col148 varchar(40) NOT NULL,
col149 varchar(40) NOT NULL,
col150 varchar(40) NOT NULL,
col151 varchar(40) NOT NULL,
col152 varchar(40) NOT NULL,
col153 varchar(40) NOT NULL,
col154 varchar(40) NOT NULL,
col155 varchar(40) NOT NULL,
col156 varchar(40) NOT NULL,
col157 varchar(40) NOT NULL,
col158 varchar(40) NOT NULL,
col159 varchar(40) NOT NULL,
col160 varchar(40) NOT NULL,
col161 varchar(40) NOT NULL,
col162 varchar(40) NOT NULL,
col163 varchar(40) NOT NULL,
col164 varchar(40) NOT NULL,
col165 varchar(40) NOT NULL,
col166 varchar(40) NOT NULL,
col167 varchar(40) NOT NULL,
col168 varchar(40) NOT NULL,
col169 varchar(40) NOT NULL,
col170 varchar(40) NOT NULL
```

```

col118 varchar(40) NOT NULL,
col119 varchar(40) NOT NULL,
col120 varchar(40) NOT NULL,
col121 varchar(40) NOT NULL,
col122 varchar(40) NOT NULL,
col123 varchar(40) NOT NULL,
col124 varchar(40) NOT NULL,
col125 varchar(40) NOT NULL,
col126 varchar(40) NOT NULL,
col127 varchar(40) NOT NULL,
col128 varchar(40) NOT NULL,
col129 varchar(40) NOT NULL,
col130 varchar(40) NOT NULL,
col131 varchar(40) NOT NULL,
col132 varchar(40) NOT NULL,
col133 varchar(40) NOT NULL,
col134 varchar(40) NOT NULL,
col135 varchar(40) NOT NULL,
col136 varchar(40) NOT NULL,
col137 varchar(40) NOT NULL,
col138 varchar(40) NOT NULL,
col139 varchar(40) NOT NULL,
col140 varchar(40) NOT NULL,
col141 varchar(40) NOT NULL,
col142 varchar(40) NOT NULL,
col143 varchar(40) NOT NULL,
col144 varchar(40) NOT NULL,
col145 varchar(40) NOT NULL,
col146 varchar(40) NOT NULL,
col147 varchar(40) NOT NULL,
col148 varchar(40) NOT NULL,
PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.

```

## Root Cause of the Problem

The root cause is that InnoDB has a maximum row size that is roughly equivalent to half of the value of the

[innodb\\_page\\_size](#)

system variable. See [InnoDB Row Formats Overview: Maximum Row Size](#) for more information.

InnoDB's row formats work around this limit by storing certain kinds of variable-length columns on overflow pages. However, different row formats can store different types of data on overflow pages. Some row formats can store more data in overflow pages than others. For example, the

[DYNAMIC](#)

and

[COMPRESSED](#)

row formats can store the most data in overflow pages. To learn how the various InnoDB row formats use overflow pages, see the following pages:

- [InnoDB REDUNDANT Row Format: Overflow Pages with the REDUNDANT Row Format](#)
- [InnoDB COMPACT Row Format: Overflow Pages with the COMPACT Row Format](#)
- [InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format](#)
- [InnoDB COMPRESSED Row Format: Overflow Pages with the COMPRESSED Row Format](#)

## Checking Existing Tables for the Problem

InnoDB does not currently have an easy way to check all existing tables to determine which tables have this problem. See [MDEV-20400](#) for more information.

One method to check a single existing table for this problem is to enable [InnoDB strict mode](#), and then try to create a duplicate of the table with

[CREATE TABLE ... LIKE](#)

. If the table has this problem, then the operation will fail. For example:

```

SET SESSION innodb_strict_mode=ON;

CREATE TABLE tab_dup LIKE tab;
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.

```

## Finding All Tables That Currently Have the Problem

The following shell script will read through a MariaDB server to identify every table that has a row size definition that is too large for its row format and the server's page size. It runs on most common distributions of Linux.

To run the script, copy the code below to a shell-script named

```

rowcount.sh
, make it executable with the command
chmod 755 ./rowsize.sh
, and invoke it with the following parameters:

```

```
./rowsize.sh host user password
```

When the script runs, it displays the name of the temporary database it creates, so that if the script is interrupted before cleaning up, the database can be easily identified and removed manually.

As the script runs it will output one line reporting the database and tablename for each table it finds that has the oversize row problem. If it finds none, it will print the following message: "No tables with rows size too big found."

In either case, the script prints one final line to announce when it's done:

```
./rowsize.sh done.
```

```

#!/bin/bash

[ -z "$3" ] && echo "Usage: $0 host user password" >&2 && exit 1

dt="tmp_$(RANDOM$RANDOM"

mysql -h $1 -u $2 -p$p3 -ABNe "create database $dt;" 
[ $? -ne 0 ] && echo "Error: $0 terminating" >&2 exit 1

echo
echo "Created temporary database ${dt} on host $1"
echo

c=0
for d in $(mysql -h $1 -u $2 -p$p3 -ABNe "show databases;" | egrep -iv "information_schema|mysql|performance_schema|$dt")
do
  for t in $(mysql -h $1 -u $2 -p$p3 -ABNe "show tables;" $d)
  do
    tc=$(mysql -h $1 -u $2 -p$p3 -ABNe "show create table $t\G" $d | egrep -iv "^\*|^$")

    echo $tc | grep -iq "ROW_FORMAT"
    if [ $? -ne 0 ]
    then
      tf=$(mysql -h $1 -u $2 -p$p3 -ABNe "select row_format from information_schema.innodb_sys_tables where name = '$d/$t';")
      tc="$tc ROW_FORMAT=$tf"
    fi

    ef="/tmp/e$RANDOM$RANDOM"
    mysql -h $1 -u $2 -p$p3 -ABNe "set innodb_strict_mode=1; set foreign_key_checks=0; ${tc};" $dt >/dev/null 2>$ef
    [ $? -ne 0 ] && cat $ef | grep -q "Row size too large" && echo "${d}.${t}" && let c++ || mysql -h $1 -u $2 -p$p3 -ABNe "drop database $dt"
    rm -f $ef
  done
done
mysql -h $1 -u $2 -p$p3 -ABNe "set innodb_strict_mode=1; drop database $dt;"
[ $c -eq 0 ] && echo "No tables with rows size too large found." || echo && echo "$c tables found with row size too large."
echo
echo "$0 done."

```

## Solving the Problem

There are several potential solutions available to solve this problem.

## Converting the Table to the DYNAMIC Row Format

If the table is using either the

REDUNDANT

or the

COMPACT

row format, then one potential solution to this problem is to convert the table to use the

DYNAMIC

row format instead.

If your tables were originally created on an older version of MariaDB or MySQL, then your table may be using one of InnoDB's older row formats:

- In [MariaDB 10.1](#) and before, and in MySQL 5.6 and before, the

COMPACT

row format was the default row format.

- In MySQL 4.1 and before, the

REDUNDANT

row format was the default row format.

The

DYNAMIC

row format can store more data on overflow pages than these older row formats, so this row format may actually be able to store the table's data safely. See [InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format](#) for more information.

Therefore, a potential solution to the *Row size too large* error is to convert the table to use the

DYNAMIC

row format. For example:

```
ALTER TABLE tab ROW_FORMAT=DYNAMIC;
```

You can use the

INNODB\_SYS\_TABLES

table in the

information\_schema

database to find all tables that use the

REDUNDANT

or the

COMPACT

row formats. This is helpful if you would like to convert all of your tables that you still use the older row formats to the

DYNAMIC

row format. For example, the following query can find those tables, while excluding InnoDB's internal system tables :

```
SELECT NAME, ROW_FORMAT
FROM information_schema.INNODB_SYS_TABLES
WHERE ROW_FORMAT IN('Redundant', 'Compact')
AND NAME NOT IN('SYS_DATAFILES', 'SYS_FOREIGN', 'SYS_FOREIGN_COLS', 'SYS_TABLESPACES', 'SYS_VIRTUAL', 'SYS_ZIP_DICT', 'SYS_ZIP_D')
```

In MariaDB 10.2 and later, the

DYNAMIC

row format is the default row format. If your tables were originally created on one of these newer versions, then they may already be using this row format. In that case, you may need to try the next solution.

## Fitting More Columns on Overflow Pages

If the table is already using the

DYNAMIC

row format, then another potential solution to this problem is to change the table schema, so that the row format can store more columns on overflow pages.

In order for InnoDB to store some variable-length columns on overflow pages, the length of those columns may need to be increased.

Therefore, a counter-intuitive solution to the *Row size too large* error in a lot of cases is actually to **increase** the length of some variable-length columns, so that InnoDB's row format can store them on overflow pages.

Some possible ways to change the table schema are listed below.

### Converting Some Columns to

BLOB

or

TEXT

For

BLOB

and

TEXT

columns, the

DYNAMIC

row format can store these columns on overflow pages. See [InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format](#) for more information.

Therefore, a potential solution to the *Row size too large* error is to convert some columns to the

BLOB

or

TEXT

data types.

### Increasing the Length of VARBINARY Columns

For

VARBINARY

columns, the

DYNAMIC

row format can only store these columns on overflow pages if the maximum length of the column is 256 bytes or longer. See [InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format](#) for more information.

Therefore, a potential solution to the *Row size too large* error is to ensure that all

**VARBINARY**

columns are at least as long as  
varbinary(256)

## Increasing the Length of

**VARCHAR**  
Columns

For

**VARCHAR**

columns, the

**DYNAMIC**

row format can only store these columns on overflow pages if the maximum length of the column is 256 bytes or longer. See [InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format](#) for more information.

The original table schema shown earlier on this page causes the *Row size too large* error, because all of the table's

**VARCHAR**

columns are smaller than 256 bytes, which means that they have to be stored on the row's main data page.

Therefore, a potential solution to the *Row size too large* error is to ensure that all

**VARCHAR**

columns are at least as long as 256 bytes. The number of characters required to reach the 256 byte limit depends on the **character set** used by the column.

For example, when using InnoDB's

**DYNAMIC**

row format and a default character set of

**latin1**

(which requires up to 1 byte per character), the 256 byte limit means that a

**VARCHAR**

column will only be stored on overflow pages if it is at least as large as a

varchar(256)

:

```
SET GLOBAL innodb_default_row_format='dynamic';
SET SESSION innodb_strict_mode=ON;
CREATE OR REPLACE TABLE tab (
    col1 varchar(256) NOT NULL,
    col2 varchar(256) NOT NULL,
    col3 varchar(256) NOT NULL,
    col4 varchar(256) NOT NULL,
    col5 varchar(256) NOT NULL,
    col6 varchar(256) NOT NULL,
    col7 varchar(256) NOT NULL,
    col8 varchar(256) NOT NULL,
    col9 varchar(256) NOT NULL,
    col10 varchar(256) NOT NULL,
    col11 varchar(256) NOT NULL,
    col12 varchar(256) NOT NULL,
    col13 varchar(256) NOT NULL,
    col14 varchar(256) NOT NULL,
    col15 varchar(256) NOT NULL,
    col16 varchar(256) NOT NULL,
    col17 varchar(256) NOT NULL)
```

```
col11 varchar(256) NOT NULL,  
col18 varchar(256) NOT NULL,  
col19 varchar(256) NOT NULL,  
col20 varchar(256) NOT NULL,  
col21 varchar(256) NOT NULL,  
col22 varchar(256) NOT NULL,  
col23 varchar(256) NOT NULL,  
col24 varchar(256) NOT NULL,  
col25 varchar(256) NOT NULL,  
col26 varchar(256) NOT NULL,  
col27 varchar(256) NOT NULL,  
col28 varchar(256) NOT NULL,  
col29 varchar(256) NOT NULL,  
col30 varchar(256) NOT NULL,  
col31 varchar(256) NOT NULL,  
col32 varchar(256) NOT NULL,  
col33 varchar(256) NOT NULL,  
col34 varchar(256) NOT NULL,  
col35 varchar(256) NOT NULL,  
col36 varchar(256) NOT NULL,  
col37 varchar(256) NOT NULL,  
col38 varchar(256) NOT NULL,  
col39 varchar(256) NOT NULL,  
col40 varchar(256) NOT NULL,  
col41 varchar(256) NOT NULL,  
col42 varchar(256) NOT NULL,  
col43 varchar(256) NOT NULL,  
col44 varchar(256) NOT NULL,  
col45 varchar(256) NOT NULL,  
col46 varchar(256) NOT NULL,  
col47 varchar(256) NOT NULL,  
col48 varchar(256) NOT NULL,  
col49 varchar(256) NOT NULL,  
col50 varchar(256) NOT NULL,  
col51 varchar(256) NOT NULL,  
col52 varchar(256) NOT NULL,  
col53 varchar(256) NOT NULL,  
col54 varchar(256) NOT NULL,  
col55 varchar(256) NOT NULL,  
col56 varchar(256) NOT NULL,  
col57 varchar(256) NOT NULL,  
col58 varchar(256) NOT NULL,  
col59 varchar(256) NOT NULL,  
col60 varchar(256) NOT NULL,  
col61 varchar(256) NOT NULL,  
col62 varchar(256) NOT NULL,  
col63 varchar(256) NOT NULL,  
col64 varchar(256) NOT NULL,  
col65 varchar(256) NOT NULL,  
col66 varchar(256) NOT NULL,  
col67 varchar(256) NOT NULL,  
col68 varchar(256) NOT NULL,  
col69 varchar(256) NOT NULL,  
col70 varchar(256) NOT NULL,  
col71 varchar(256) NOT NULL,  
col72 varchar(256) NOT NULL,  
col73 varchar(256) NOT NULL,  
col74 varchar(256) NOT NULL,  
col75 varchar(256) NOT NULL,  
col76 varchar(256) NOT NULL,  
col77 varchar(256) NOT NULL,  
col78 varchar(256) NOT NULL,  
col79 varchar(256) NOT NULL,  
col80 varchar(256) NOT NULL,  
col81 varchar(256) NOT NULL,  
col82 varchar(256) NOT NULL,  
col83 varchar(256) NOT NULL,  
col84 varchar(256) NOT NULL,  
col85 varchar(256) NOT NULL,  
col86 varchar(256) NOT NULL,  
col87 varchar(256) NOT NULL,  
col88 varchar(256) NOT NULL,  
col89 varchar(256) NOT NULL,  
col90 varchar(256) NOT NULL,  
col91 varchar(256) NOT NULL,  
col92 varchar(256) NOT NULL,  
col93 varchar(256) NOT NULL,  
col94 varchar(256) NOT NULL,
```



```

col112 varchar(256) NOT NULL,
col113 varchar(256) NOT NULL,
col114 varchar(256) NOT NULL,
col115 varchar(256) NOT NULL,
col116 varchar(256) NOT NULL,
col117 varchar(256) NOT NULL,
col118 varchar(256) NOT NULL,
col119 varchar(256) NOT NULL,
col120 varchar(256) NOT NULL,
col121 varchar(256) NOT NULL,
col122 varchar(256) NOT NULL,
col123 varchar(256) NOT NULL,
col124 varchar(256) NOT NULL,
col125 varchar(256) NOT NULL,
col126 varchar(256) NOT NULL,
col127 varchar(256) NOT NULL,
PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

And when using InnoDB's

DYNAMIC

row format and a default character set of

utf8

(which requires up to 3 bytes per character), the 256 byte limit means that a

VARCHAR

column will only be stored on overflow pages if it is at least as large as a

varchar(86)

:

```

SET GLOBAL innodb_default_row_format='dynamic';
SET SESSION innodb_strict_mode=ON;
CREATE OR REPLACE TABLE tab (
    col1 varchar(86) NOT NULL,
    col2 varchar(86) NOT NULL,
    col3 varchar(86) NOT NULL,
    col4 varchar(86) NOT NULL,
    col5 varchar(86) NOT NULL,
    col6 varchar(86) NOT NULL,
    col7 varchar(86) NOT NULL,
    col8 varchar(86) NOT NULL,
    col9 varchar(86) NOT NULL,
    col10 varchar(86) NOT NULL,
    col11 varchar(86) NOT NULL,
    col12 varchar(86) NOT NULL,
    col13 varchar(86) NOT NULL,
    col14 varchar(86) NOT NULL,
    col15 varchar(86) NOT NULL,
    col16 varchar(86) NOT NULL,
    col17 varchar(86) NOT NULL,
    col18 varchar(86) NOT NULL,
    col19 varchar(86) NOT NULL,
    col20 varchar(86) NOT NULL,
    col21 varchar(86) NOT NULL,
    col22 varchar(86) NOT NULL,
    col23 varchar(86) NOT NULL,
    col24 varchar(86) NOT NULL,
    col25 varchar(86) NOT NULL,
    col26 varchar(86) NOT NULL,
    col27 varchar(86) NOT NULL,

```

```
col28 varchar(86) NOT NULL,
col29 varchar(86) NOT NULL,
col30 varchar(86) NOT NULL,
col31 varchar(86) NOT NULL,
col32 varchar(86) NOT NULL,
col33 varchar(86) NOT NULL,
col34 varchar(86) NOT NULL,
col35 varchar(86) NOT NULL,
col36 varchar(86) NOT NULL,
col37 varchar(86) NOT NULL,
col38 varchar(86) NOT NULL,
col39 varchar(86) NOT NULL,
col40 varchar(86) NOT NULL,
col41 varchar(86) NOT NULL,
col42 varchar(86) NOT NULL,
col43 varchar(86) NOT NULL,
col44 varchar(86) NOT NULL,
col45 varchar(86) NOT NULL,
col46 varchar(86) NOT NULL,
col47 varchar(86) NOT NULL,
col48 varchar(86) NOT NULL,
col49 varchar(86) NOT NULL,
col50 varchar(86) NOT NULL,
col51 varchar(86) NOT NULL,
col52 varchar(86) NOT NULL,
col53 varchar(86) NOT NULL,
col54 varchar(86) NOT NULL,
col55 varchar(86) NOT NULL,
col56 varchar(86) NOT NULL,
col57 varchar(86) NOT NULL,
col58 varchar(86) NOT NULL,
col59 varchar(86) NOT NULL,
col60 varchar(86) NOT NULL,
col61 varchar(86) NOT NULL,
col62 varchar(86) NOT NULL,
col63 varchar(86) NOT NULL,
col64 varchar(86) NOT NULL,
col65 varchar(86) NOT NULL,
col66 varchar(86) NOT NULL,
col67 varchar(86) NOT NULL,
col68 varchar(86) NOT NULL,
col69 varchar(86) NOT NULL,
col70 varchar(86) NOT NULL,
col71 varchar(86) NOT NULL,
col72 varchar(86) NOT NULL,
col73 varchar(86) NOT NULL,
col74 varchar(86) NOT NULL,
col75 varchar(86) NOT NULL,
col76 varchar(86) NOT NULL,
col77 varchar(86) NOT NULL,
col78 varchar(86) NOT NULL,
col79 varchar(86) NOT NULL,
col80 varchar(86) NOT NULL,
col81 varchar(86) NOT NULL,
col82 varchar(86) NOT NULL,
col83 varchar(86) NOT NULL,
col84 varchar(86) NOT NULL,
col85 varchar(86) NOT NULL,
col86 varchar(86) NOT NULL,
col87 varchar(86) NOT NULL,
col88 varchar(86) NOT NULL,
col89 varchar(86) NOT NULL,
col90 varchar(86) NOT NULL,
col91 varchar(86) NOT NULL,
col92 varchar(86) NOT NULL,
col93 varchar(86) NOT NULL,
col94 varchar(86) NOT NULL,
col95 varchar(86) NOT NULL,
col96 varchar(86) NOT NULL,
col97 varchar(86) NOT NULL,
col98 varchar(86) NOT NULL,
col99 varchar(86) NOT NULL,
col100 varchar(86) NOT NULL,
col101 varchar(86) NOT NULL,
col102 varchar(86) NOT NULL,
col103 varchar(86) NOT NULL,
col104 varchar(86) NOT NULL,
col105 varchar(86) NOT NULL
```

```
col105 varchar(86) NOT NULL,  
col106 varchar(86) NOT NULL,  
col107 varchar(86) NOT NULL,  
col108 varchar(86) NOT NULL,  
col109 varchar(86) NOT NULL,  
col110 varchar(86) NOT NULL,  
col111 varchar(86) NOT NULL,  
col112 varchar(86) NOT NULL,  
col113 varchar(86) NOT NULL,  
col114 varchar(86) NOT NULL,  
col115 varchar(86) NOT NULL,  
col116 varchar(86) NOT NULL,  
col117 varchar(86) NOT NULL,  
col118 varchar(86) NOT NULL,  
col119 varchar(86) NOT NULL,  
col120 varchar(86) NOT NULL,  
col121 varchar(86) NOT NULL,  
col122 varchar(86) NOT NULL,  
col123 varchar(86) NOT NULL,  
col124 varchar(86) NOT NULL,  
col125 varchar(86) NOT NULL,  
col126 varchar(86) NOT NULL,  
col127 varchar(86) NOT NULL,  
col128 varchar(86) NOT NULL,  
col129 varchar(86) NOT NULL,  
col130 varchar(86) NOT NULL,  
col131 varchar(86) NOT NULL,  
col132 varchar(86) NOT NULL,  
col133 varchar(86) NOT NULL,  
col134 varchar(86) NOT NULL,  
col135 varchar(86) NOT NULL,  
col136 varchar(86) NOT NULL,  
col137 varchar(86) NOT NULL,  
col138 varchar(86) NOT NULL,  
col139 varchar(86) NOT NULL,  
col140 varchar(86) NOT NULL,  
col141 varchar(86) NOT NULL,  
col142 varchar(86) NOT NULL,  
col143 varchar(86) NOT NULL,  
col144 varchar(86) NOT NULL,  
col145 varchar(86) NOT NULL,  
col146 varchar(86) NOT NULL,  
col147 varchar(86) NOT NULL,  
col148 varchar(86) NOT NULL,  
col149 varchar(86) NOT NULL,  
col150 varchar(86) NOT NULL,  
col151 varchar(86) NOT NULL,  
col152 varchar(86) NOT NULL,  
col153 varchar(86) NOT NULL,  
col154 varchar(86) NOT NULL,  
col155 varchar(86) NOT NULL,  
col156 varchar(86) NOT NULL,  
col157 varchar(86) NOT NULL,  
col158 varchar(86) NOT NULL,  
col159 varchar(86) NOT NULL,  
col160 varchar(86) NOT NULL,  
col161 varchar(86) NOT NULL,  
col162 varchar(86) NOT NULL,  
col163 varchar(86) NOT NULL,  
col164 varchar(86) NOT NULL,  
col165 varchar(86) NOT NULL,  
col166 varchar(86) NOT NULL,  
col167 varchar(86) NOT NULL,  
col168 varchar(86) NOT NULL,  
col169 varchar(86) NOT NULL,  
col170 varchar(86) NOT NULL,  
col171 varchar(86) NOT NULL,  
col172 varchar(86) NOT NULL,  
col173 varchar(86) NOT NULL,  
col174 varchar(86) NOT NULL,  
col175 varchar(86) NOT NULL,  
col176 varchar(86) NOT NULL,  
col177 varchar(86) NOT NULL,  
col178 varchar(86) NOT NULL,  
col179 varchar(86) NOT NULL,  
col180 varchar(86) NOT NULL,  
col181 varchar(86) NOT NULL,  
col182 varchar(86) NOT NULL,
```

```

col183 varchar(86) NOT NULL,
col184 varchar(86) NOT NULL,
col185 varchar(86) NOT NULL,
col186 varchar(86) NOT NULL,
col187 varchar(86) NOT NULL,
col188 varchar(86) NOT NULL,
col189 varchar(86) NOT NULL,
col190 varchar(86) NOT NULL,
col191 varchar(86) NOT NULL,
col192 varchar(86) NOT NULL,
col193 varchar(86) NOT NULL,
col194 varchar(86) NOT NULL,
col195 varchar(86) NOT NULL,
col196 varchar(86) NOT NULL,
col197 varchar(86) NOT NULL,
col198 varchar(86) NOT NULL,
PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

And when using InnoDB's

DYNAMIC

row format and a default character set of

utf8mb4

(which requires up to 4 bytes per character), the 256 byte limit means that a

VARCHAR

column will only be stored on overflow pages if it is at least as large as a

varchar(64)

:

```

SET GLOBAL innodb_default_row_format='dynamic';
SET SESSION innodb_strict_mode=ON;
CREATE OR REPLACE TABLE tab (
    col1 varchar(64) NOT NULL,
    col2 varchar(64) NOT NULL,
    col3 varchar(64) NOT NULL,
    col4 varchar(64) NOT NULL,
    col5 varchar(64) NOT NULL,
    col6 varchar(64) NOT NULL,
    col7 varchar(64) NOT NULL,
    col8 varchar(64) NOT NULL,
    col9 varchar(64) NOT NULL,
    col10 varchar(64) NOT NULL,
    col11 varchar(64) NOT NULL,
    col12 varchar(64) NOT NULL,
    col13 varchar(64) NOT NULL,
    col14 varchar(64) NOT NULL,
    col15 varchar(64) NOT NULL,
    col16 varchar(64) NOT NULL,
    col17 varchar(64) NOT NULL,
    col18 varchar(64) NOT NULL,
    col19 varchar(64) NOT NULL,
    col20 varchar(64) NOT NULL,
    col21 varchar(64) NOT NULL,
    col22 varchar(64) NOT NULL,
    col23 varchar(64) NOT NULL,
    col24 varchar(64) NOT NULL,
    col25 varchar(64) NOT NULL,
    col26 varchar(64) NOT NULL,
    col27 varchar(64) NOT NULL,
    col28 varchar(64) NOT NULL,
    col29 varchar(64) NOT NULL,
    col30 varchar(64) NOT NULL,
    col31 varchar(64) NOT NULL,
    col32 varchar(64) NOT NULL,
    col33 varchar(64) NOT NULL,
    col34 varchar(64) NOT NULL,
    col35 varchar(64) NOT NULL,
    col36 varchar(64) NOT NULL,
    col37 varchar(64) NOT NULL,
    col38 varchar(64) NOT NULL
);

```

```
col150 varchar(64) NOT NULL,  
col39 varchar(64) NOT NULL,  
col40 varchar(64) NOT NULL,  
col41 varchar(64) NOT NULL,  
col42 varchar(64) NOT NULL,  
col43 varchar(64) NOT NULL,  
col44 varchar(64) NOT NULL,  
col45 varchar(64) NOT NULL,  
col46 varchar(64) NOT NULL,  
col47 varchar(64) NOT NULL,  
col48 varchar(64) NOT NULL,  
col49 varchar(64) NOT NULL,  
col50 varchar(64) NOT NULL,  
col51 varchar(64) NOT NULL,  
col52 varchar(64) NOT NULL,  
col53 varchar(64) NOT NULL,  
col54 varchar(64) NOT NULL,  
col55 varchar(64) NOT NULL,  
col56 varchar(64) NOT NULL,  
col57 varchar(64) NOT NULL,  
col58 varchar(64) NOT NULL,  
col59 varchar(64) NOT NULL,  
col60 varchar(64) NOT NULL,  
col61 varchar(64) NOT NULL,  
col62 varchar(64) NOT NULL,  
col63 varchar(64) NOT NULL,  
col64 varchar(64) NOT NULL,  
col65 varchar(64) NOT NULL,  
col66 varchar(64) NOT NULL,  
col67 varchar(64) NOT NULL,  
col68 varchar(64) NOT NULL,  
col69 varchar(64) NOT NULL,  
col70 varchar(64) NOT NULL,  
col71 varchar(64) NOT NULL,  
col72 varchar(64) NOT NULL,  
col73 varchar(64) NOT NULL,  
col74 varchar(64) NOT NULL,  
col75 varchar(64) NOT NULL,  
col76 varchar(64) NOT NULL,  
col77 varchar(64) NOT NULL,  
col78 varchar(64) NOT NULL,  
col79 varchar(64) NOT NULL,  
col80 varchar(64) NOT NULL,  
col81 varchar(64) NOT NULL,  
col82 varchar(64) NOT NULL,  
col83 varchar(64) NOT NULL,  
col84 varchar(64) NOT NULL,  
col85 varchar(64) NOT NULL,  
col86 varchar(64) NOT NULL,  
col87 varchar(64) NOT NULL,  
col88 varchar(64) NOT NULL,  
col89 varchar(64) NOT NULL,  
col90 varchar(64) NOT NULL,  
col91 varchar(64) NOT NULL,  
col92 varchar(64) NOT NULL,  
col93 varchar(64) NOT NULL,  
col94 varchar(64) NOT NULL,  
col95 varchar(64) NOT NULL,  
col96 varchar(64) NOT NULL,  
col97 varchar(64) NOT NULL,  
col98 varchar(64) NOT NULL,  
col99 varchar(64) NOT NULL,  
col100 varchar(64) NOT NULL,  
col101 varchar(64) NOT NULL,  
col102 varchar(64) NOT NULL,  
col103 varchar(64) NOT NULL,  
col104 varchar(64) NOT NULL,  
col105 varchar(64) NOT NULL,  
col106 varchar(64) NOT NULL,  
col107 varchar(64) NOT NULL,  
col108 varchar(64) NOT NULL,  
col109 varchar(64) NOT NULL,  
col110 varchar(64) NOT NULL,  
col111 varchar(64) NOT NULL,  
col112 varchar(64) NOT NULL,  
col113 varchar(64) NOT NULL,  
col114 varchar(64) NOT NULL,  
col115 varchar(64) NOT NULL,
```

```
col116 varchar(64) NOT NULL,  
col117 varchar(64) NOT NULL,  
col118 varchar(64) NOT NULL,  
col119 varchar(64) NOT NULL,  
col120 varchar(64) NOT NULL,  
col121 varchar(64) NOT NULL,  
col122 varchar(64) NOT NULL,  
col123 varchar(64) NOT NULL,  
col124 varchar(64) NOT NULL,  
col125 varchar(64) NOT NULL,  
col126 varchar(64) NOT NULL,  
col127 varchar(64) NOT NULL,  
col128 varchar(64) NOT NULL,  
col129 varchar(64) NOT NULL,  
col130 varchar(64) NOT NULL,  
col131 varchar(64) NOT NULL,  
col132 varchar(64) NOT NULL,  
col133 varchar(64) NOT NULL,  
col134 varchar(64) NOT NULL,  
col135 varchar(64) NOT NULL,  
col136 varchar(64) NOT NULL,  
col137 varchar(64) NOT NULL,  
col138 varchar(64) NOT NULL,  
col139 varchar(64) NOT NULL,  
col140 varchar(64) NOT NULL,  
col141 varchar(64) NOT NULL,  
col142 varchar(64) NOT NULL,  
col143 varchar(64) NOT NULL,  
col144 varchar(64) NOT NULL,  
col145 varchar(64) NOT NULL,  
col146 varchar(64) NOT NULL,  
col147 varchar(64) NOT NULL,  
col148 varchar(64) NOT NULL,  
col149 varchar(64) NOT NULL,  
col150 varchar(64) NOT NULL,  
col151 varchar(64) NOT NULL,  
col152 varchar(64) NOT NULL,  
col153 varchar(64) NOT NULL,  
col154 varchar(64) NOT NULL,  
col155 varchar(64) NOT NULL,  
col156 varchar(64) NOT NULL,  
col157 varchar(64) NOT NULL,  
col158 varchar(64) NOT NULL,  
col159 varchar(64) NOT NULL,  
col160 varchar(64) NOT NULL,  
col161 varchar(64) NOT NULL,  
col162 varchar(64) NOT NULL,  
col163 varchar(64) NOT NULL,  
col164 varchar(64) NOT NULL,  
col165 varchar(64) NOT NULL,  
col166 varchar(64) NOT NULL,  
col167 varchar(64) NOT NULL,  
col168 varchar(64) NOT NULL,  
col169 varchar(64) NOT NULL,  
col170 varchar(64) NOT NULL,  
col171 varchar(64) NOT NULL,  
col172 varchar(64) NOT NULL,  
col173 varchar(64) NOT NULL,  
col174 varchar(64) NOT NULL,  
col175 varchar(64) NOT NULL,  
col176 varchar(64) NOT NULL,  
col177 varchar(64) NOT NULL,  
col178 varchar(64) NOT NULL,  
col179 varchar(64) NOT NULL,  
col180 varchar(64) NOT NULL,  
col181 varchar(64) NOT NULL,  
col182 varchar(64) NOT NULL,  
col183 varchar(64) NOT NULL,  
col184 varchar(64) NOT NULL,  
col185 varchar(64) NOT NULL,  
col186 varchar(64) NOT NULL,  
col187 varchar(64) NOT NULL,  
col188 varchar(64) NOT NULL,  
col189 varchar(64) NOT NULL,  
col190 varchar(64) NOT NULL,  
col191 varchar(64) NOT NULL,  
col192 varchar(64) NOT NULL,  
col193 varchar(64) NOT NULL.
```

```
-- col194 varchar(64) NOT NULL,  
col195 varchar(64) NOT NULL,  
col196 varchar(64) NOT NULL,  
col197 varchar(64) NOT NULL,  
col198 varchar(64) NOT NULL,  
PRIMARY KEY (col1)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

## Working Around the Problem

There are a few ways to work around this problem.

If you would like a solution for the problem instead of just working around it, then see the solutions mentioned in the previous section.

### Refactoring the Table into Multiple Tables

A *safe* workaround is to refactor the single wide table, so that its columns are spread among multiple tables.

This workaround can even work if your table is so wide that the previous solutions have failed to solve them problem for your table.

### Refactoring Some Columns into JSON

A *safe* workaround is to refactor some of the columns into a JSON document.

The JSON document can be queried and manipulated using MariaDB's [JSON functions](#).

The JSON document can be stored in a column that uses one of the following data types:

•

[TEXT](#)

: The maximum size of a

[TEXT](#)

column is 64 KB.

•

[MEDIUMTEXT](#)

: The maximum size of a

[MEDIUMTEXT](#)

column is 16 MB.

•

[LONGTEXT](#)

: The maximum size of a

[LONGTEXT](#)

column is 4 GB.

•

[JSON](#)

: This is just an alias for the

[LONGTEXT](#)

data type.

This workaround can even work if your table is so wide that the previous solutions have failed to solve them problem for your table.

### Disabling InnoDB Strict Mode

An *unsafe* workaround is to disable [InnoDB strict mode](#). [InnoDB strict mode](#) can be disabled by setting the

`innodb_strict_mode`

system variable to  
OFF

For example, even though the following table schema is too large for most InnoDB row formats to store, it can still be created when [InnoDB strict mode](#) is disabled:

```
SET GLOBAL innodb_default_row_format='dynamic';
SET SESSION innodb_strict_mode=OFF;
CREATE OR REPLACE TABLE tab (
    col1 varchar(40) NOT NULL,
    col2 varchar(40) NOT NULL,
    col3 varchar(40) NOT NULL,
    col4 varchar(40) NOT NULL,
    col5 varchar(40) NOT NULL,
    col6 varchar(40) NOT NULL,
    col7 varchar(40) NOT NULL,
    col8 varchar(40) NOT NULL,
    col9 varchar(40) NOT NULL,
    col10 varchar(40) NOT NULL,
    col11 varchar(40) NOT NULL,
    col12 varchar(40) NOT NULL,
    col13 varchar(40) NOT NULL,
    col14 varchar(40) NOT NULL,
    col15 varchar(40) NOT NULL,
    col16 varchar(40) NOT NULL,
    col17 varchar(40) NOT NULL,
    col18 varchar(40) NOT NULL,
    col19 varchar(40) NOT NULL,
    col20 varchar(40) NOT NULL,
    col21 varchar(40) NOT NULL,
    col22 varchar(40) NOT NULL,
    col23 varchar(40) NOT NULL,
    col24 varchar(40) NOT NULL,
    col25 varchar(40) NOT NULL,
    col26 varchar(40) NOT NULL,
    col27 varchar(40) NOT NULL,
    col28 varchar(40) NOT NULL,
    col29 varchar(40) NOT NULL,
    col30 varchar(40) NOT NULL,
    col31 varchar(40) NOT NULL,
    col32 varchar(40) NOT NULL,
    col33 varchar(40) NOT NULL,
    col34 varchar(40) NOT NULL,
    col35 varchar(40) NOT NULL,
    col36 varchar(40) NOT NULL,
    col37 varchar(40) NOT NULL,
    col38 varchar(40) NOT NULL,
    col39 varchar(40) NOT NULL,
    col40 varchar(40) NOT NULL,
    col41 varchar(40) NOT NULL,
    col42 varchar(40) NOT NULL,
    col43 varchar(40) NOT NULL,
    col44 varchar(40) NOT NULL,
    col45 varchar(40) NOT NULL,
    col46 varchar(40) NOT NULL,
    col47 varchar(40) NOT NULL,
    col48 varchar(40) NOT NULL,
    col49 varchar(40) NOT NULL,
    col50 varchar(40) NOT NULL,
    col51 varchar(40) NOT NULL,
    col52 varchar(40) NOT NULL,
    col53 varchar(40) NOT NULL,
    col54 varchar(40) NOT NULL,
    col55 varchar(40) NOT NULL,
    col56 varchar(40) NOT NULL,
    col57 varchar(40) NOT NULL,
    col58 varchar(40) NOT NULL,
    col59 varchar(40) NOT NULL,
    col60 varchar(40) NOT NULL,
    col61 varchar(40) NOT NULL,
    col62 varchar(40) NOT NULL,
    col63 varchar(40) NOT NULL,
    col64 varchar(40) NOT NULL,
    col65 varchar(40) NOT NULL,
    col66 varchar(40) NOT NULL,
    col67 varchar(40) NOT NULL,
    col68 varchar(40) NOT NULL)
```

```
col108 varchar(40) NOT NULL,  
col109 varchar(40) NOT NULL,  
col110 varchar(40) NOT NULL,  
col111 varchar(40) NOT NULL,  
col112 varchar(40) NOT NULL,  
col113 varchar(40) NOT NULL,  
col114 varchar(40) NOT NULL,  
col115 varchar(40) NOT NULL,  
col116 varchar(40) NOT NULL,  
col117 varchar(40) NOT NULL,  
col118 varchar(40) NOT NULL,  
col119 varchar(40) NOT NULL,  
col120 varchar(40) NOT NULL,  
col121 varchar(40) NOT NULL,  
col122 varchar(40) NOT NULL,  
col123 varchar(40) NOT NULL,  
col124 varchar(40) NOT NULL,  
col125 varchar(40) NOT NULL,  
col126 varchar(40) NOT NULL,  
col127 varchar(40) NOT NULL,  
col128 varchar(40) NOT NULL,  
col129 varchar(40) NOT NULL,  
col130 varchar(40) NOT NULL,  
col131 varchar(40) NOT NULL,  
col132 varchar(40) NOT NULL,  
col133 varchar(40) NOT NULL,  
col134 varchar(40) NOT NULL,  
col135 varchar(40) NOT NULL,  
col136 varchar(40) NOT NULL,  
col137 varchar(40) NOT NULL,  
col138 varchar(40) NOT NULL,  
col139 varchar(40) NOT NULL,  
col140 varchar(40) NOT NULL,  
col141 varchar(40) NOT NULL,  
col142 varchar(40) NOT NULL,  
col143 varchar(40) NOT NULL,  
col144 varchar(40) NOT NULL,  
col145 varchar(40) NOT NULL,
```

```

col146 varchar(40) NOT NULL,
col147 varchar(40) NOT NULL,
col148 varchar(40) NOT NULL,
col149 varchar(40) NOT NULL,
col150 varchar(40) NOT NULL,
col151 varchar(40) NOT NULL,
col152 varchar(40) NOT NULL,
col153 varchar(40) NOT NULL,
col154 varchar(40) NOT NULL,
col155 varchar(40) NOT NULL,
col156 varchar(40) NOT NULL,
col157 varchar(40) NOT NULL,
col158 varchar(40) NOT NULL,
col159 varchar(40) NOT NULL,
col160 varchar(40) NOT NULL,
col161 varchar(40) NOT NULL,
col162 varchar(40) NOT NULL,
col163 varchar(40) NOT NULL,
col164 varchar(40) NOT NULL,
col165 varchar(40) NOT NULL,
col166 varchar(40) NOT NULL,
col167 varchar(40) NOT NULL,
col168 varchar(40) NOT NULL,
col169 varchar(40) NOT NULL,
col170 varchar(40) NOT NULL,
col171 varchar(40) NOT NULL,
col172 varchar(40) NOT NULL,
col173 varchar(40) NOT NULL,
col174 varchar(40) NOT NULL,
col175 varchar(40) NOT NULL,
col176 varchar(40) NOT NULL,
col177 varchar(40) NOT NULL,
col178 varchar(40) NOT NULL,
col179 varchar(40) NOT NULL,
col180 varchar(40) NOT NULL,
col181 varchar(40) NOT NULL,
col182 varchar(40) NOT NULL,
col183 varchar(40) NOT NULL,
col184 varchar(40) NOT NULL,
col185 varchar(40) NOT NULL,
col186 varchar(40) NOT NULL,
col187 varchar(40) NOT NULL,
col188 varchar(40) NOT NULL,
col189 varchar(40) NOT NULL,
col190 varchar(40) NOT NULL,
col191 varchar(40) NOT NULL,
col192 varchar(40) NOT NULL,
col193 varchar(40) NOT NULL,
col194 varchar(40) NOT NULL,
col195 varchar(40) NOT NULL,
col196 varchar(40) NOT NULL,
col197 varchar(40) NOT NULL,
col198 varchar(40) NOT NULL,
PRIMARY KEY (col1)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

But as mentioned above, if [InnoDB strict mode](#) is **disabled** and if a [DDL](#) statement is executed, then InnoDB will still raise a **warning** with this message. The

`SHOW WARNINGS`

statement can be used to view the warning. For example:

```

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 139 | Row size too large (> 8126). Changing some columns to TEXT or BLOB may help. In current row format, BLOB prefix
+-----+-----+
1 row in set (0.000 sec)

```

As mentioned above, even though InnoDB is allowing the table to be created, there is still an opportunity for errors. Regardless of whether [InnoDB strict mode](#) is enabled, if a [DML](#) statement is executed that attempts to write a row that the table's InnoDB row format can't store, then InnoDB will raise an **error** with this message. This creates a somewhat *unsafe* situation, because it means that the application has the chance to encounter an additional error while executing [DML](#).

## 4.3.2.4 InnoDB System Variables

### Contents

1. [have\\_innodb](#)
2. [ignore\\_builtin\\_innodb](#)
3. [innodb\\_adaptive\\_checkpoint](#)
4. [innodb\\_adaptive\\_flushing](#)
5. [innodb\\_adaptive\\_flushing\\_lwm](#)
6. [innodb\\_adaptive\\_flushing\\_method](#)
7. [innodb\\_adaptive\\_hash\\_index](#)
8. [innodb\\_adaptive\\_hash\\_index\\_partitions](#)
9. [innodb\\_adaptive\\_hash\\_index\\_parts](#)
10. [innodb\\_adaptive\\_max\\_sleep\\_delay](#)
11. [innodb\\_additional\\_mem\\_pool\\_size](#)
12. [innodb\\_api\\_bk\\_commit\\_interval](#)
13. [innodb\\_api\\_disable\\_rowlock](#)
14. [innodb\\_api\\_enable\\_binlog](#)
15. [innodb\\_api\\_enable\\_mdl](#)
16. [innodb\\_api\\_trx\\_level](#)
17. [innodb\\_auto\\_lru\\_dump](#)
18. [innodb\\_autoextend\\_increment](#)
19. [innodb\\_autoinc\\_lock\\_mode](#)
20. [innodb\\_background\\_scrub\\_data\\_check\\_interval](#)
21. [innodb\\_background\\_scrub\\_data\\_compressed](#)
22. [innodb\\_background\\_scrub\\_data\\_interval](#)
23. [innodb\\_background\\_scrub\\_data\\_uncompressed](#)
24. [innodb\\_blocking\\_buffer\\_pool\\_restore](#)
25. [innodb\\_buf\\_dump\\_status\\_frequency](#)
26. [innodb\\_buffer\\_pool\\_chunk\\_size](#)
27. [innodb\\_buffer\\_pool\\_dump\\_at\\_shutdown](#)
28. [innodb\\_buffer\\_pool\\_dump\\_now](#)
29. [innodb\\_buffer\\_pool\\_dump\\_pct](#)
30. [innodb\\_buffer\\_pool\\_evict](#)
31. [innodb\\_buffer\\_pool\\_filename](#)
32. [innodb\\_buffer\\_pool\\_instances](#)
33. [innodb\\_buffer\\_pool\\_load\\_abort](#)
34. [innodb\\_buffer\\_pool\\_load\\_at\\_startup](#)
35. [innodb\\_buffer\\_pool\\_load\\_now](#)
36. [innodb\\_buffer\\_pool\\_load\\_pages\\_abort](#)
37. [innodb\\_buffer\\_pool\\_populate](#)
38. [innodb\\_buffer\\_pool\\_restore\\_at\\_startup](#)
39. [innodb\\_buffer\\_pool\\_shm\\_checksum](#)
40. [innodb\\_buffer\\_pool\\_shm\\_key](#)
41. [innodb\\_buffer\\_pool\\_size](#)
42. [innodb\\_change\\_buffer\\_dump](#)
43. [innodb\\_change\\_buffer\\_max\\_size](#)
44. [innodb\\_change\\_buffering](#)
45. [innodb\\_change\\_buffering\\_debug](#)
46. [innodb\\_checkpoint\\_age\\_target](#)
47. [innodb\\_checksum\\_algorithm](#)
48. [innodb\\_checksums](#)
49. [innodb\\_cleaner\\_lsn\\_age\\_factor](#)
50. [innodb\\_cmp\\_per\\_index\\_enabled](#)
51. [innodb\\_commit\\_concurrency](#)
52. [innodb\\_compression\\_algorithm](#)
53. [innodb\\_compression\\_default](#)
54. [innodb\\_compression\\_failure\\_threshold\\_pct](#)
55. [innodb\\_compression\\_level](#)
56. [innodb\\_compression\\_pad\\_pct\\_max](#)
57. [innodb\\_concurrency\\_tickets](#)
58. [innodb\\_corrupt\\_table\\_action](#)
59. [innodb\\_data\\_file\\_path](#)
60. [innodb\\_data\\_home\\_dir](#)
61. [innodb\\_deadlock\\_detect](#)
62. [innodb\\_deadlock\\_report](#)
63. [innodb\\_default\\_page\\_encryption\\_key](#)
64. [innodb\\_default\\_encryption\\_key\\_id](#)
65. [innodb\\_default\\_row\\_format](#)
66. [innodb\\_defragment](#)
67. [innodb\\_defragment\\_fill\\_factor](#)
68. [innodb\\_defragment\\_fill\\_factor\\_n\\_recs](#)
69. [innodb\\_defragment\\_frequency](#)

68. innodb\_low\_priority\_updates  
70. innodb\_defragment\_n\_pages  
71. innodb\_defragment\_stats\_accuracy  
72. innodb\_dict\_size\_limit  
73. innodb\_disable\_sort\_file\_cache  
74. innodb\_disallow\_writes  
75. innodb\_doublewrite  
76. innodb\_doublewrite\_file  
77. innodb\_empty\_free\_list\_algorithm  
78. innodb\_enable\_unsafe\_group\_commit  
79. innodb\_encrypt\_log  
80. innodb\_encrypt\_tables  
81. innodb\_encrypt\_temporary\_tables  
82. innodb\_encryption\_rotate\_key\_age  
83. innodb\_encryption\_rotation\_iops  
84. innodb\_encryption\_threads  
85. innodb\_extra\_rsegments  
86. innodb\_extra\_undoslots  
87. innodb\_fake\_changes  
88. innodb\_fast\_checksum  
89. innodb\_fast\_shutdown  
90. innodb\_fatal\_semaphore\_wait\_threshold  
91. innodb\_file\_format  
92. innodb\_file\_format\_check  
93. innodb\_file\_format\_max  
94. innodb\_file\_per\_table  
95. innodb\_fill\_factor  
96. innodb\_flush\_log\_at\_timeout  
97. innodb\_flush\_log\_at\_trx\_commit  
98. innodb\_flush\_method  
99. innodb\_flush\_neighbor\_pages  
00. innodb\_flush\_neighbors  
01. innodb\_flush\_sync  
02. innodb\_flushing\_avg\_loops  
03. innodb\_force\_load\_corrupted  
04. innodb\_force\_primary\_key  
05. innodb\_force\_recovery  
06. innodb\_foreground\_preflush  
07. innodb\_ft\_aux\_table  
08. innodb\_ft\_cache\_size  
09. innodb\_ft\_enable\_diag\_print  
10. innodb\_ft\_enable\_stopword  
11. innodb\_ft\_max\_token\_size  
12. innodb\_ft\_min\_token\_size  
13. innodb\_ft\_num\_word\_optimize  
14. innodb\_ft\_result\_cache\_limit  
15. innodb\_ft\_server\_stopword\_table  
16. innodb\_ft\_sort\_pll\_degree  
17. innodb\_ft\_total\_cache\_size  
18. innodb\_ft\_user\_stopword\_table  
19. innodb\_ibuf\_accel\_rate  
20. innodb\_ibuf\_active\_contract  
21. innodb\_ibuf\_max\_size  
22. innodb\_idle\_flush\_pct  
23. innodb\_immediate\_scrub\_data\_uncompressed  
24. innodb\_import\_table\_from\_xtrabackup  
25. innodb\_instant\_alter\_column\_allowed  
26. innodb\_instrument\_semaphores  
27. innodb\_io\_capacity  
28. innodb\_io\_capacity\_max  
29. innodb\_kill\_idle\_transaction  
30. innodb\_large\_prefix  
31. innodb\_lazy\_drop\_table  
32. innodb\_lock\_schedule\_algorithm  
33. innodb\_lock\_wait\_timeout  
34. innodb\_locking\_fake\_changes  
35. innodb\_locks\_unsafe\_for\_binlog  
36. innodb\_log\_arch\_dir  
37. innodb\_log\_arch\_expire\_sec  
38. innodb\_log\_archive  
39. innodb\_log\_block\_size  
40. innodb\_log\_buffer\_size  
41. innodb\_log\_checksum\_algorithm

42. innodb\_log\_checksums  
43. innodb\_log\_compressed\_pages  
44. innodb\_log\_file\_size  
45. innodb\_log\_files\_in\_group  
46. innodb\_log\_group\_home\_dir  
47. innodb\_log\_optimize\_ddl  
48. innodb\_log\_write\_ahead\_size  
49. innodb\_lru\_flush\_size  
50. innodb\_lru\_scan\_depth  
51. innodb\_max\_bitmap\_file\_size  
52. innodb\_max\_changed\_pages  
53. innodb\_max\_dirty\_pages\_pct  
54. innodb\_max\_dirty\_pages\_pct\_lwm  
55. innodb\_max\_purge\_lag  
56. innodb\_max\_purge\_lag\_delay  
57. innodb\_max\_purge\_lag\_wait  
58. innodb\_max\_undo\_log\_size  
59. innodb\_merge\_sort\_block\_size  
60. innodb\_mirrored\_log\_groups  
61. innodb\_mtflush\_threads  
62. innodb\_monitor\_disable  
63. innodb\_monitor\_enable  
64. innodb\_monitor\_reset  
65. innodb\_monitor\_reset\_all  
66. innodb\_numa\_interleave  
67. innodb\_old\_blocks\_pct  
68. innodb\_old\_blocks\_time  
69. innodb\_online\_alter\_log\_max\_size  
70. innodb\_open\_files  
71. innodb\_optimize\_fulltext\_only  
72. innodb\_page\_cleaners  
73. innodb\_page\_size  
74. innodb\_pass\_corrupt\_table  
75. innodb\_prefix\_index\_cluster\_optimization  
76. innodb\_print\_all\_deadlocks  
77. innodb\_purge\_batch\_size  
78. innodb\_purge\_rseg\_truncate\_frequency  
79. innodb\_purge\_threads  
80. innodb\_random\_read\_ahead  
81. innodb\_read\_ahead  
82. innodb\_read\_ahead\_threshold  
83. innodb\_read\_io\_threads  
84. innodb\_read\_only  
85. innodb\_read\_only\_compressed  
86. innodb\_recovery\_stats  
87. innodb\_recovery\_update\_relay\_log  
88. innodb\_replication\_delay  
89. innodb\_rollback\_on\_timeout  
90. innodb\_rollback\_segments  
91. innodb\_safe\_truncate  
92. innodb\_scrub\_log  
93. innodb\_scrub\_log\_interval  
94. innodb\_scrub\_log\_speed  
95. innodb\_sched\_priority\_cleaner  
96. innodb\_show\_locks\_held  
97. innodb\_show\_verbose\_locks  
98. innodb\_simulate\_comp\_failures  
99. innodb\_sort\_buffer\_size  
00. innodb\_spin\_wait\_delay  
01. innodb\_stats\_auto\_recalc  
02. innodb\_stats\_auto\_update  
03. innodb\_stats\_include\_delete\_marked  
04. innodb\_stats\_method  
05. innodb\_stats\_modified\_counter  
06. innodb\_stats\_on\_metadata  
07. innodb\_stats\_persistent  
08. innodb\_stats\_persistent\_sample\_pages  
09. innodb\_stats\_sample\_pages  
10. innodb\_stats\_traditional  
11. innodb\_stats\_transient\_sample\_pages  
12. innodb\_stats\_update\_need\_lock  
13. innodb\_status\_output  
14. innodb\_status\_output\_locks

```
15. innodb_strict_mode
16. innodb_support_xa
17. innodb_sync_array_size
18. innodb_sync_spin_loops
19. innodb_table_locks
20. innodb_thread_concurrency
21. innodb_thread_concurrency_timer_based
22. innodb_thread_sleep_delay
23. innodb_temp_data_file_path
24. innodb_tmpdir
25. innodb_track_changed_pages
26. innodb_track_redo_log_now
27. innodb_undo_directory
28. innodb_undo_log_truncate
29. innodb_undo_logs
30. innodb_undo tablespaces
31. innodb_use_atomic_writes
32. innodb_use_fallocate
33. innodb_use_global_flush_log_at_trx_comm
34. innodb_use_mtflush
35. innodb_use_native_aio
36. innodb_use_purge_thread
37. innodb_use_stacktrace
38. innodb_use_sys_malloc
39. innodb_use_sys_stats_table
40. innodb_use_trim
41. innodb_version
42. innodb_write_io_threads
```

This page documents system variables related to the [InnoDB storage engine](#). For options that are not system variables, see [InnoDB Options](#).

See [Server System Variables](#) for a complete list of system variables and instructions on setting them.

Also see the [Full list of MariaDB options, system and status variables](#).

## have\_innodb

- **Description:** If the server supports [InnoDB tables](#), will be set to

YES  
, otherwise will be set to  
NO

. Removed in [MariaDB 10.0](#), use the [Information Schema PLUGINS](#) table or [SHOW ENGINES](#) instead.

- **Scope:** Global
- **Dynamic:** No
- **Removed:** [MariaDB 10.0](#)

## ignore\_builtin\_innodb

- **Description:** Setting this to

1

results in the built-in InnoDB storage engine being ignored. In some versions of MariaDB, XtraDB is the default and is always present, so this variable is ignored and setting it results in a warning. From [MariaDB 10.0.1](#) to [MariaDB 10.0.8](#), when InnoDB was the default instead of XtraDB, this variable needed to be set. Usually used in conjunction with the [plugin-load=innodb=ha\\_innodb](#) option to use the InnoDB plugin.

- **Commandline:**

--ignore-built-in-innodb

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**

boolean

- **Default Value:**

OFF

## innodb\_adaptive\_checkpoint

- **Description:** Replaced with [innodb\\_adaptive\\_flushing\\_method](#). Controls adaptive checkpointing. InnoDB's fuzzy checkpointing can cause stalls, as many dirty blocks are flushed at once as the checkpoint age nears the maximum. Adaptive checkpointing aims for more consistent flushing, approximately

```
modified age / maximum checkpoint age  
. Can result in larger transaction log files
```

- - reflex  
Similar to [innodb\\_max\\_dirty\\_pages\\_pct](#) flushing but flushes blocks constantly and contiguously based on the oldest modified age. If the age exceeds 1/2 of the maximum age capacity, flushing will be weak contiguous. If the age exceeds 3/4, flushing will be strong. Strength can be adjusted by the variable [innodb\\_io\\_capacity](#).

- - estimate  
The default, and independent of [innodb\\_io\\_capacity](#). If the oldest modified age exceeds 1/2 of the maximum age capacity, blocks will be flushed every second at a rate determined by the number of modified blocks, LSN progress speed and the average age of all modified blocks.

- - keep\_average  
Attempts to keep the I/O rate constant by using a shorter loop cycle of one tenth of a second. Designed for SSD cards.

- **Commandline:**

```
--innodb-adaptive-checkpoint=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
string
```

- **Default Value:**

```
estimate
```

- **Valid Values:**

```
none  
or  
0  
,  
reflex  
or  
1  
,  
estimate  
or  
2  
,  
keep_average  
or  
3
```

- **Removed:** XtraDB 5.5 - replaced with [innodb\\_adaptive\\_flushing\\_method](#)

---

## innodb\_adaptive\_flushing

- **Description:** If set to

```
1
```

, the default, the server will dynamically adjust the flush rate of dirty pages in the InnoDB buffer pool. This assists to reduce brief bursts of I/O activity. If set to

```
0
```

, adaptive flushing will only take place when the limit specified by [innodb\\_adaptive\\_flushing\\_lwm](#) is reached.

- **Commandline:**

```
--innodb-adaptive-flushing={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
boolean
```

- **Default Value:**

```
ON
```

## `innodb_adaptive_flushing_lwm`

- **Description:** Adaptive flushing is enabled when this low water mark percentage of the [InnoDB redo log](#) capacity is reached. Takes effect even if `innodb_adaptive_flushing` is disabled.

- **Commandline:**

```
--innodb-adaptive-flushing-lwm=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

double

- **Default Value:**

10.000000

- **Range:**

0

to

70

## `innodb_adaptive_flushing_method`

- **Description:** Determines the method of flushing dirty blocks from the InnoDB [buffer pool](#). If set to

native

or

0

, the original InnoDB method is used. The maximum checkpoint age is determined by the total length of all transaction log files. When the checkpoint age reaches the maximum checkpoint age, blocks are flushed. This can cause lag if there are many updates per second and many blocks with an almost identical age need to be flushed. If set to

estimate

or

1

, the default, the oldest modified age will be compared with the maximum age capacity. If it's more than 1/4 of this age, blocks are flushed every second. The number of blocks flushed is determined by the number of modified blocks, the LSN progress speed and the average age of all modified blocks. It's therefore independent of the [innodb\\_io\\_capacity](#) for the 1-second loop, but not entirely so for the 10-second loop. If set to

keep\_average

or

2

, designed specifically for SSD cards, a shorter loop cycle is used in an attempt to keep the I/O rate constant. Removed in [MariaDB 10.0](#) /XtraDB 5.6 and replaced with InnoDB flushing method from MySQL 5.6.

- **Commandline:**

```
innodb-adaptive-flushing-method=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enumeration

- **Default Value:**

estimate

- **Valid Values:**

native

or

0

,

estimate

or

1

,

keep\_average

or

2

- **Removed:** MariaDB 10.0 - replaced with InnoDB flushing method from MySQL 5.6
- 

## innodb\_adaptive\_hash\_index

- **Description:** If set to

1

, the default until MariaDB 10.5 , the InnoDB hash index is enabled. Based on performance testing ( MDEV-17492 ), the InnoDB adaptive hash index helps performance in mostly read-only workloads, and could slow down performance in other environments, especially [DROP TABLE](#) , [TRUNCATE TABLE](#) , [ALTER TABLE](#) , or [DROP INDEX](#) operations.

- **Commandline:**

--innodb-adaptive-hash-index={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

(>= MariaDB 10.5 ),

ON

(<= MariaDB 10.4 )

## innodb\_adaptive\_hash\_index\_partitions

- **Description:** Specifies the number of partitions for use in adaptive searching. If set to

1

, no extra partitions are created. XtraDB-only. From MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB), this is an alias for [innodb\\_adaptive\\_hash\\_index\\_parts](#) to allow for easier upgrades.

- **Commandline:**

innodb-adaptive-hash-index-partitions=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

1

to

64

## innodb\_adaptive\_hash\_index\_parts

- **Description:** Specifies the number of partitions for use in adaptive searching. If set to

1

, no extra partitions are created.

- **Commandline:**

innodb-adaptive-hash-index-parts=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

8

- **Range:**

1  
to  
512

- **Introduced:** MariaDB 10.2.2
- 

### innodb\_adaptive\_max\_sleep\_delay

- **Description:** Maximum time in microseconds to automatically adjust the `innodb_thread_sleep_delay` value to, based on the workload. Useful in extremely busy systems with hundreds of thousands of simultaneous connections.

0  
disables any limit. Deprecated and ignored from MariaDB 10.5.5 .

- **Commandline:**

--innodb-adaptive-max-sleep-delay=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

- 0  
(>= MariaDB 10.5.5 )
- 150000  
(<= MariaDB 10.5.4 )

- **Range:**

0  
to  
1000000

- **Introduced:** MariaDB 10.0

- **Deprecated:** MariaDB 10.5.5

- **Removed:** MariaDB 10.6.0
- 

### innodb\_additional\_mem\_pool\_size

- **Description:** Size in bytes of the InnoDB memory pool used for storing information about internal data structures. Defaults to 8MB, if your application has many tables and a large structure, and this is exceeded, operating system memory will be allocated and warning messages written to the error log, in which case you should increase this value. Deprecated in MariaDB 10.0 and removed in MariaDB 10.2 along with InnoDB's internal memory allocator.

- **Commandline:**

--innodb-additional-mem-pool-size=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

8388608

- **Range:**

2097152  
to  
4294967295

- **Deprecated:** MariaDB 10.0

- **Removed:** MariaDB 10.2.2
-

## innodb\_api\_bk\_commit\_interval

- **Description:** Time in seconds between auto-commits for idle connections using the InnoDB memcached interface (not implemented in MariaDB).
  - **Commandline:**  
--innodb-api-bk-commit-interval=#
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
numeric
  - **Default Value:**  
5
  - **Range:**  
1  
to  
1073741824
  - **Introduced:** [MariaDB 10.0](#)
  - **Removed:** [MariaDB 10.2.4](#)
- 

## innodb\_api\_disable\_rowlock

- **Description:** For use with MySQL's memcached (not implemented in MariaDB)
  - **Commandline:**  
--innodb-api-disable-rowlock={0|1}
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
  - **Introduced:** [MariaDB 10.0](#)
  - **Removed:** [MariaDB 10.2.4](#)
- 

## innodb\_api\_enable\_binlog

- **Description:** For use with MySQL's memcached (not implemented in MariaDB)
  - **Commandline:**  
--innodb-api-enable-binlog={0|1}
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
  - **Introduced:** [MariaDB 10.0](#)
  - **Removed:** [MariaDB 10.2.4](#)
- 

## innodb\_api\_enable\_mdl

- **Description:** For use with MySQL's memcached (not implemented in MariaDB)
- **Commandline:**  
--innodb-api-enable-mdl={0|1}

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** MariaDB 10.0

- **Removed:** MariaDB 10.2.4
- 

## innodb\_api\_trx\_level

- **Description:** For use with MySQL's memcached (not implemented in MariaDB)

- **Commandline:**

--innodb-api-trx-level=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Introduced:** MariaDB 10.0

- **Removed:** MariaDB 10.2.4
- 

## innodb\_auto\_lru\_dump

- **Description:** Renamed `innodb_buffer_pool_restore_at_startup` since XtraDB 5.5.10-20.1, which was in turn replaced by `innodb_buffer_pool_load_at_startup` in MariaDB 10.0 .

- **Commandline:**

--innodb-auto-lru-dump=#

- **Removed:** XtraDB 5.5.10-20.1
- 

## innodb\_autoextend\_increment

- **Description:** Size in MB to increment an auto-extending shared tablespace file when it becomes full. If `innodb_file_per_table` was set to

1

, this setting does not apply to the resulting per-table tablespace files, which are automatically extended in their own way.

- **Commandline:**

--innodb-autoextend-increment=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

64

(from MariaDB 10.0 )

8

(before MariaDB 10.0 ),

- **Range:**

1

to

1000

## `innodb_autoinc_lock_mode`

- **Description:** The lock mode that is used when generating

`AUTO_INCREMENT`

values for InnoDB tables.

- Valid values are:

0

is the traditional lock mode.

1

is the consecutive lock mode.

2

is the interleaved lock mode.

- In order to use [Galera Cluster](#), the lock mode needs to be set to

2

- See [AUTO\\_INCREMENT Handling in InnoDB: AUTO\\_INCREMENT Lock Modes](#) for more information.

- **Commandline:**

`--innodb-autoinc-lock-mode=#`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0

to

2

## `innodb_background_scrub_data_check_interval`

- **Description:** Check if spaces needs scrubbing every `innodb_background_scrub_data_check_interval` seconds. See [Data Scrubbing](#). Deprecated and ignored from [MariaDB 10.5.2](#).

- **Commandline:**

`--innodb-background-scrub-data-check-interval=#`

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

3600

- **Range:**

1

to

4294967295

- **Introduced:** [MariaDB 10.1.3](#)

- **Deprecated:** [MariaDB 10.5.2](#)

- **Removed:** [MariaDB 10.6.0](#)

## `innodb_background_scrub_data_compressed`

- **Description:** Enable scrubbing of compressed data by background threads (same as encryption\_threads). See [Data Scrubbing](#). Deprecated and ignored from [MariaDB 10.5.2](#).
  - **Commandline:**

```
--innodb-background-scrub-data-compressed={0|1}
```
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value:**  
0
  - **Introduced:** [MariaDB 10.1.3](#)
  - **Deprecated:** [MariaDB 10.5.2](#)
  - **Removed:** [MariaDB 10.6.0](#)
- 

### innodb\_background\_scrub\_data\_interval

- **Description:** Scrub spaces that were last scrubbed longer than this number of seconds ago. See [Data Scrubbing](#). Deprecated and ignored from [MariaDB 10.5.2](#).
  - **Commandline:**

```
--innodb-background-scrub-data-interval=#
```
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
numeric
  - **Default Value:**  
604800
  - **Range:**  
1  
to  
4294967295
  - **Introduced:** [MariaDB 10.1.3](#)
  - **Deprecated:** [MariaDB 10.5.2](#)
  - **Removed:** [MariaDB 10.6.0](#)
- 

### innodb\_background\_scrub\_data\_uncompressed

- **Description:** Enable scrubbing of uncompressed data by background threads (same as encryption\_threads). See [Data Scrubbing](#). Deprecated and ignored from [MariaDB 10.5.2](#).
  - **Commandline:**

```
--innodb-background-scrub-data-uncompressed={0|1}
```
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value:**  
0
  - **Introduced:** [MariaDB 10.1.3](#)
  - **Deprecated:** [MariaDB 10.5.2](#)
  - **Removed:** [MariaDB 10.6.0](#)
- 

### innodb\_blocking\_buffer\_pool\_restore

- **Description:** If set to

1

(

0

is default), XtraDB will wait until the least-recently used (LRU) dump is completely restored upon restart before reporting back to the server that it has successfully started up. Available with XtraDB only, not InnoDB.

- **Commandline:**

`innodb-blocking-buffer-pool-restore={0|1}`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Removed:** MariaDB 10.0.0
- 

## innodb\_buf\_dump\_status\_frequency

- **Description:** Determines how often (as a percent) the buffer pool dump status should be printed in the logs. For example,

10

means that the buffer pool dump status is printed when every 10% of the number of buffer pool pages are dumped. The default is

0

(only start and end status is printed).

- **Commandline:**

`--innodb-buf-dump-status-frequency=#`

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

100

- **Introduced:** MariaDB 10.1.6
- 

## innodb\_buffer\_pool\_chunk\_size

- **Description:** Chunk size used for dynamically resizing the [buffer pool](#). Note that changing this setting can change the size of the buffer pool. When [large-pages](#) is used this value is effectively rounded up to the next multiple of [large-page-size](#). See [Setting Innodb Buffer Pool Size Dynamically](#). From [MariaDB 10.8.0](#), the variable is autosized based on the [buffer pool size](#).

- **Commandline:**

`--innodb-buffer-pool-chunk-size=#`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

◦

`autosize (0)`

, resulting in [innodb\\_buffer\\_pool\\_size](#) /64, if [large\\_pages](#) round down to multiple of largest page size, with 1MiB minimum (>= [MariaDB 10.8.1](#))

◦

`134217728`

(<= [MariaDB 10.8.0](#))

- **Range:**

- - 0 , as autosize, and then 1048576 to 18446744073709551615 (>= MariaDB 10.8 )
    - 1048576 to innodb\_buffer\_pool\_size / innodb\_buffer\_pool\_instances (<= MariaDB 10.7 )
  - **Introduced:** MariaDB 10.2.2
- 

## innodb\_buffer\_pool\_dump\_at\_shutdown

- **Description:** Whether to record pages cached in the [buffer pool](#) on server shutdown, which reduces the length of the warmup the next time the server starts. The related [innodb\\_buffer\\_pool\\_load\\_at\\_startup](#) specifies whether the buffer pool is automatically warmed up at startup.
  - **Commandline:**
    - innodb-buffer-pool-dump-at-shutdown={0|1}
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**
    - boolean
  - **Default Value:**
    - ON (>= MariaDB 10.2.2 )
    - OFF (<= MariaDB 10.2.1 )
  - **Introduced:** MariaDB 10.0
- 

## innodb\_buffer\_pool\_dump\_now

- **Description:** Immediately records pages stored in the [buffer pool](#). The related [innodb\\_buffer\\_pool\\_load\\_now](#) does the reverse, and will immediately warm up the buffer pool.
  - **Commandline:**
    - innodb-buffer-pool-dump-now={0|1}
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**
    - boolean
  - **Default Value:**
    - OFF
  - **Introduced:** MariaDB 10.0
- 

## innodb\_buffer\_pool\_dump\_pct

- **Description:** Dump only the hottest N% of each [buffer pool](#), defaults to 100 until MariaDB 10.2.1 . Since MariaDB 10.2.2 , defaults to 25% along with the changes to [innodb\\_buffer\\_pool\\_dump\\_at\\_shutdown](#) and [innodb\\_buffer\\_pool\\_load\\_at\\_startup](#).
- **Commandline:**
  - innodb-buffer-pool-dump-pct={0|1}
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**
  - boolean

- **Default Value:**

- - 25  
( $\geq$  MariaDB 10.2.2)
  - - 100  
( $\leq$  MariaDB 10.2.1)

- **Range:**

- 1  
to  
100

- **Introduced:** MariaDB 10.1.10

---

### innodb\_buffer\_pool\_evict

- **Description:** Evict pages from the buffer pool. If set to "uncompressed" then all uncompressed pages are evicted from the buffer pool. Variable to be used only for testing. Only exists in DEBUG builds.

- **Commandline:**

- innodb-buffer-pool-evict=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

- string

- **Default Value:**

- ""

- **Valid Values:** "" or "uncompressed"

---

### innodb\_buffer\_pool\_filename

- **Description:** The file that holds the **buffer pool** list of page numbers set by [innodb\\_buffer\\_pool\\_dump\\_at\\_shutdown](#) and [innodb\\_buffer\\_pool\\_dump\\_now](#).

- **Commandline:**

- innodb-buffer-pool-filename=file

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

- string

- **Default Value:**

- ib\_buffer\_pool

- **Introduced:** MariaDB 10.0

---

### innodb\_buffer\_pool\_instances

- **Description:** If **innodb\_buffer\_pool\_size** is set to more than 1GB, **innodb\_buffer\_pool\_instances** divides the **InnoDB** buffer pool into this many instances. The default was 1 in [MariaDB 5.5](#), but for large systems with buffer pools of many gigabytes, many instances can help reduce contention concurrency. The default is 8 in MariaDB 10 (except on Windows 32-bit, where it varies according to **innodb\_buffer\_pool\_size**, or from [MariaDB 10.2.2](#), where it is set to 1 if **innodb\_buffer\_pool\_size** < 1GB). Each instance manages its own data structures and takes an equal portion of the total buffer pool size, so for example if **innodb\_buffer\_pool\_size** is 4GB and **innodb\_buffer\_pool\_instances** is set to 4, each instance will be 1GB. Each instance should ideally be at least 1GB in size. Deprecated and ignored from [MariaDB 10.5.1](#), as the original reasons for splitting the buffer pool have mostly gone away.

- **Commandline:**

- innodb-buffer-pool-instances=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:** >= MariaDB 10.0.4 :

8

,

1

(>= MariaDB 10.2.2 if innodb\_buffer\_pool\_size < 1GB), or dependent on innodb\_buffer\_pool\_size (Windows 32-bit)

- **Deprecated:** MariaDB 10.5.1

- **Removed:** MariaDB 10.6.0

---

## innodb\_buffer\_pool\_load\_abort

- **Description:** Aborts the process of restoring buffer pool contents started by innodb\_buffer\_pool\_load\_at\_startup or innodb\_buffer\_pool\_load\_now .

- **Commandline:**

--innodb-buffer-pool-load-abort={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

## innodb\_buffer\_pool\_load\_at\_startup

- **Description:** Specifies whether the buffer pool is automatically warmed up when the server starts by loading the pages held earlier. The related innodb\_buffer\_pool\_dump\_at\_shutdown specifies whether pages are saved at shutdown. If the buffer pool is large and taking a long time to load, increasing innodb\_io\_capacity at startup may help.

- **Commandline:**

--innodb-buffer-pool-load-at-startup={0|1}

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

◦

ON

(>= MariaDB 10.2.2 )

◦

OFF

(<= MariaDB 10.2.1 )

---

## innodb\_buffer\_pool\_load\_now

- **Description:** Immediately warms up the buffer pool by loading the stored data pages. The related innodb\_buffer\_pool\_dump\_now does the reverse, and immediately records pages stored in the buffer pool.

- **Commandline:**

--innodb-buffer-pool-load-now={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** MariaDB 10.0
- 

### innodb\_buffer\_pool\_load\_pages\_abort

- **Description:** Number of pages during a buffer pool load to process before signaling `innodb_buffer_pool_load_abort=1`. Debug builds only.
- **Commandline:**

```
--innodb-buffer-pool-load-pages-abort=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

9223372036854775807

- **Range:**

1

to

9223372036854775807

- **Introduced:** MariaDB 10.3
- 

### innodb\_buffer\_pool\_populate

- **Description:** When set to

1

(

0

is default), XtraDB will preallocate pages in the buffer pool on starting up so that NUMA allocation decisions are made while the buffer cache is still clean. XtraDB only. This option was made ineffective in MariaDB 10.0.23 . Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

```
innodb-buffer-pool-populate={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Deprecated:** MariaDB 10.0.23

- **Removed:** MariaDB 10.3.0
- 

### innodb\_buffer\_pool\_restore\_at\_startup

- **Description:** Time in seconds between automatic buffer pool dumps. If set to a non-zero value, XtraDB will also perform an automatic restore of the `buffer pool` at startup. If set to

0

, automatic dumps are not performed, nor automatic restores on startup. Replaced by `innodb_buffer_pool_load_at_startup` in MariaDB 10.0 .

- **Commandline:**

```
innodb-buffer-pool-restore-at-startup
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range - 32 bit:**

0  
to  
4294967295

- **Range - 64 bit:**

0  
to  
18446744073709547520

- **Removed:** [MariaDB 10.0](#) - replaced by `innodb_buffer_pool_load_at_startup`
- 

### `innodb_buffer_pool_shm_checksum`

- **Description:** Used with Percona's SHM buffer pool patch in XtraDB 5.5. Was shortly deprecated and removed in XtraDB 5.6. XtraDB only.

- **Commandline:**

`innodb-buffer-pool-shm-checksum={0|1}`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

- **Removed:** [MariaDB 10.0](#)
- 

### `innodb_buffer_pool_shm_key`

- **Description:** Used with Percona's SHM buffer pool patch in XtraDB 5.5. Later deprecated in XtraDB 5.5, and removed in XtraDB 5.6.

- **Commandline:**

`innodb-buffer-pool-shm-key={0|1}`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

0

- **Removed:** [MariaDB 10.0](#)
- 

### `innodb_buffer_pool_size`

- **Description:** InnoDB buffer pool size in bytes. The primary value to adjust on a database server with entirely/primarily [InnoDB](#) tables, can be set up to 80% of the total memory in these environments. See the [InnoDB Buffer Pool](#) for more on setting this variable, and also [Setting InnoDB Buffer Pool Size Dynamically](#) if doing so dynamically.

- **Commandline:**

`--innodb-buffer-pool-size=#`

- **Scope:** Global

- **Dynamic:** Yes ( $\geq$  [MariaDB 10.2.2](#)), No ( $\leq$  [MariaDB 10.2.1](#))

- **Data Type:**

numeric

- **Default Value:**

134217728  
(128iMB)

- **Range:**
  - Minimum:
    - 5242880  
(5MiB ) for [InnoDB Page Size](#) <= 16k otherwise
    - 25165824  
(24MiB) for [InnoDB Page Size](#) > 16k (for versions less than next line)
  - Minimum:
    - 2MiB  
[InnoDB Page Size](#) = 4k,
    - 3MiB  
[InnoDB Page Size](#) = 8k,
    - 5MiB  
[InnoDB Page Size](#) = 16k,
    - 10MiB  
[InnoDB Page Size](#) = 32k,
    - 20MiB  
[InnoDB Page Size](#) = 64k, (>= [MariaDB 10.2.42](#) , >= [MariaDB 10.3.33](#) , >= [MariaDB 10.4.23](#) , >= [MariaDB 10.5.14](#) , >= [MariaDB 10.6.6](#) , >= [MariaDB 10.7.2](#) )
  - Minimum: 1GiB for [innodb\\_buffer\\_pool\\_instances](#) > 1 (<= [MariaDB 10.7](#) )
  - Maximum:
    - 9223372036854775807  
(8192PB) (all versions)

## innodb\_change\_buffer\_dump

- **Description:** If set, causes the contents of the InnoDB change buffer to be dumped to the server error log at startup. Only available in debug builds.
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**
  - boolean
- **Default Value:**
  - OFF
- **Introduced:** [MariaDB 10.2.28](#) , [MariaDB 10.3.19](#) , [MariaDB 10.4.9](#)

## innodb\_change\_buffer\_max\_size

- **Description:** Maximum size of the [InnoDB Change Buffer](#) as a percentage of the total buffer pool. The default is 25%, and this can be increased up to 50% for servers with high write activity, and lowered down to 0 for servers used exclusively for reporting.
- **Commandline:**
  - innodb-change-buffer-max-size=#
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**
  - numeric
- **Default Value:**
  - 25
- **Range:**
  - 0
  - to
  - 50
- **Introduced:** [MariaDB 10.0](#)

## innodb\_change\_buffering

- **Description:** Sets how [InnoDB](#) change buffering is performed. See [InnoDB Change Buffering](#) for details on the settings. Deprecated and ignored

from [MariaDB 10.9.0](#) .

- **Commandline:**  
--innodb-change-buffering=#
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
enumeration  
(>= [MariaDB 10.3.7](#) ),  
string  
(<= [MariaDB 10.3.6](#) )
- **Default Value:**
  - >= [MariaDB 10.5.15](#) , [MariaDB 10.6.7](#) , [MariaDB 10.7.3](#) , [MariaDB 10.8.2](#) :  
none
  - <= [MariaDB 10.5.14](#) , [MariaDB 10.6.6](#) , [MariaDB 10.7.2](#) , [MariaDB 10.8.1](#) :  
all
- **Valid Values:**  
inserts  
,  
none  
,  
deletes  
,  
purges  
,  
changes  
,  
all
- **Deprecated:** [MariaDB 10.9.0](#)

---

## innodb\_change\_buffering\_debug

- **Description:** If set to
  - 1  
, an [InnoDB Change Buffering](#) debug flag is set.
  - 1  
forces all changes to the change buffer, while
  - 2  
causes a crash at merge.
  - 0  
, the default, indicates no flag is set. Only available in debug builds.
- **Commandline:**  
--innodb-change-buffering-debug=#
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric
- **Default Value:**  
0
- **Range:**  
0  
to  
2

---

## innodb\_checkpoint\_age\_target

- **Description:** The maximum value of the checkpoint age. If set to
  - 0  
, has no effect. Removed in [MariaDB 10.0](#) /XtraDB 5.6 and replaced with InnoDB flushing method from MySQL 5.6.

- **Commandline:**

```
innodb_checkpoint_age_target=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    0
```

- **Range:**

```
    0
```

```
    upwards
```

- **Removed:** MariaDB 10.0 - replaced with InnoDB flushing method from MySQL 5.6.

---

## innodb\_checksum\_algorithm

- **Description:** Specifies how the InnoDB tablespace checksum is generated and verified.

```
◦
```

```
    innodb
```

```
        : Backwards compatible with earlier versions (<= MariaDB 5.5 ). Deprecated in MariaDB 10.3.29 , MariaDB 10.4.19 , MariaDB 10.5.10 and removed in MariaDB 10.6 . If really needed, data files can still be converted with innoread .
```

```
◦
```

```
    crc32
```

```
        : A newer, faster algorithm, but incompatible with earlier versions. Tablespace blocks will be converted to the new format over time, meaning that a mix of checksums may be present.
```

```
◦
```

```
    full_crc32
```

```
    and
```

```
    strict_full_crc32
```

```
        : From MariaDB 10.4.3 . Permits encryption to be supported over a SPATIAL INDEX , which
```

```
    crc32
```

```
        does not support. Newly-created data files will carry a flag that indicates that all pages of the file will use a full CRC-32C checksum over the entire page contents (excluding the bytes where the checksum is stored, at the very end of the page). Such files will always use that checksum, no matter what parameter
```

```
    innodb_checksum_algorithm
```

```
    is assigned to. Even if
```

```
    innodb_checksum_algorithm
```

```
        is modified later, the same checksum will continue to be used. A special flag will be set in the FSP_SPACE_FLAGS in the first data page to indicate the new format of checksum and encryption/page_compressed. ROW_FORMAT=COMPRESSED tables will only use the old format. These tables do not support new features, such as larger innodb_page_size or instant ADD/DROP COLUMN. Also cleans up the MariaDB tablespace flags - flags are reserved to store the page_compressed compression algorithm, and to store the compressed payload length, so that checksum can be computed over the compressed (and possibly encrypted) stream and can be validated without decrypting or decompressing the page. In the full_crc32 format, there no longer are separate before-encryption and after-encryption checksums for pages. The single checksum is computed on the page contents that is written to the file. See MDEV-12026 for details.
```

```
◦
```

```
    none
```

```
        : Writes a constant rather than calculate a checksum. Deprecated in MariaDB 10.3.29 , MariaDB 10.4.19 , MariaDB 10.5.10 and removed in MariaDB 10.6 as was mostly used to disable the original, slow, page checksum for benchmarking purposes.
```

```
◦
```

```
    strict_crc32
```

```
,
```

```
    strict_innodb
```

```
    and
```

```
    strict_none
```

```
        : The options are the same as the regular options, but InnoDB will halt if it comes across a mix of checksum values. These are faster, as both new and old checksum values are not required, but can only be used when setting up tablespaces for the first time.
```

- **Commandline:**

```
--innodb_checksum_algorithm=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    enumeration
```

- **Default Value:**

```
◦
```

```
    full_crc32
```

- - (>= MariaDB 10.5.0 )
- - crc32
    - (>= MariaDB 10.2.2 )
- - innodb
    - (<= MariaDB 10.2.1 )

• **Valid Values:**

- >= MariaDB 10.6.0 :
  - crc32
  - ,
  - full\_crc32
  - ,
  - strict\_crc32
  - ,
  - strict\_full\_crc32
- MariaDB 10.5 , >= MariaDB 10.4.3 :
  - innodb
  - ,
  - crc32
  - ,
  - full\_crc32
  - ,
  - none
  - ,
  - strict\_innodb
  - ,
  - strict\_crc32
  - ,
  - strict\_none
  - ,
  - strict\_full\_crc32
- <= MariaDB 10.4.2 :
  - innodb
  - ,
  - crc32
  - ,
  - none
  - ,
  - strict\_innodb
  - ,
  - strict\_crc32
  - ,
  - strict\_none

- **Introduced:** MariaDB 10.0
- 

### `innodb_checksums`

- **Description:** By default, InnoDB performs checksum validation on all pages read from disk, which provides extra fault tolerance. You would usually want this set to

```

1
in production environments, although setting it to
0
can provide marginal performance improvements. Deprecated and functionality replaced by innodb_checksum_algorithm in MariaDB
10.0 , and should be removed to avoid conflicts.
ON
is equivalent to
--innodb_checksum_algorithm=innodb
and
OFF
to
--innodb_checksum_algorithm=none
.
```

- **Commandline:**

```
--innodb-checksums
,
```

```
--skip-innodb-checksums
```

- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
  - **Default Value:**  
ON
  - **Deprecated:** MariaDB 10.0
  - **Removed:** MariaDB 10.5.0
- 

## innodb\_cleaner\_lsn\_age\_factor

- **Description:** XtraDB has enhanced page cleaner heuristics, and with these in place, the default InnoDB adaptive flushing may be too aggressive. As a result, a new LSN age factor formula has been introduced, controlled by this variable. The default setting,

high\_checkpoint  
, uses the new formula, while the alternative,  
legacy  
, uses the original algorithm. XtraDB only. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**  
`--innodb-cleaner-lsn-age-factor=value`

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
enum

- **Default Value:**

- deprecated  
(>= MariaDB 10.2.6 )
- high\_checkpoint  
(<= MariaDB 10.1 )

- **Valid Values:**

- high\_checkpoint  
,  
legacy  
(<= MariaDB 10.1 )
- deprecated  
,  
high\_checkpoint  
,  
legacy  
(>= MariaDB 10.2.6 )

- **Introduced:** MariaDB 10.0.9
  - **Deprecated:** MariaDB 10.2.6
  - **Removed:** MariaDB 10.3.0
- 

## innodb\_cmp\_per\_index\_enabled

- **Description:** If set to

ON  
(  
OFF

is default), per-index compression statistics are stored in the INFORMATION\_SCHEMA.INNODB\_CMP\_PER\_INDEX table. These are expensive to record, so this setting should only be changed with care, such as for performance tuning on development or replica servers.

- **Commandline:**

`--innodb-cmp-per-index-enabled={0|1}`

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** MariaDB 10.0
- 

## innodb\_commit\_concurrency

- **Description:** Limit to the number of transaction threads that can commit simultaneously. 0, the default, imposes no limit. While you can change from one positive limit to another at runtime, you cannot set this variable to 0, or change it from 0, while the server is running. Deprecated and ignored from MariaDB 10.5.5.

- **Commandline:**

--innodb-commit-concurrency=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

1000

- **Deprecated:** MariaDB 10.5.5

- **Removed:** MariaDB 10.6.0
- 

## innodb\_compression\_algorithm

- **Description:** Compression algorithm used for InnoDB page compression . The supported values are:

◦

none

: Pages are not compressed.

◦

zlib

: Pages are compressed using the bundled

[zlib](#)

compression algorithm.

◦

1z4

: Pages are compressed using the

[1z4](#)

compression algorithm.

◦

1zo

: Pages are compressed using the

[1zo](#)

compression algorithm.

◦

lzma

: Pages are compressed using the

[lzma](#)

- compression algorithm.
    - bzip2
      - : Pages are compressed using the
    - bzip2
      - bzip2
      - compression algorithm.
    - snappy
      - : Pages are compressed using the
    - snappy
      - snappy
      - algorithm.
  - On many distributions, MariaDB may not support all page compression algorithms by default. From [MariaDB 10.7](#), libraries can be installed as a plugin. See [Compression Plugins](#).
  - See [InnoDB Page Compression: Configuring the InnoDB Page Compression Algorithm](#) for more information.
- **Commandline:**
    - innodb-compression-algorithm=value
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**
    - enum
  - **Default Value:**
    - zlib
      - ([=> MariaDB 10.2.4](#) , [MariaDB 10.1.22](#) ),
      - none
        - (=< MariaDB 10.2.3 , [MariaDB 10.1.21](#) )
  - **Valid Values:**
    - none
    - ,
    - zlib
    - ,
    - lz4
    - ,
    - lzo
    - ,
    - lzma
    - ,
    - bzip2
    - or
    - snappy
    - ( [MariaDB 10.1.3](#) )
  - **Introduced:** [MariaDB 10.1.0](#)
- 

## innodb\_compression\_default

- **Description:** Whether or not [InnoDB page compression](#) is enabled by default for new tables.
    - The default value is
      - OFF
      - , which means new tables are not compressed.
    - See [InnoDB Page Compression: Enabling InnoDB Page Compression by Default](#) for more information.
    - **Commandline:**
      - innodb-compression-default={0|1}
    - **Scope:** Global, Session
    - **Dynamic:** Yes
    - **Data Type:**
      - boolean
    - **Default Value:**
      - OFF
    - **Introduced:** [MariaDB 10.2.3](#)
-

## `innodb_compression_failure_threshold_pct`

- **Description:** Specifies the percentage cutoff for expensive compression failures during updates to a table that uses [InnoDB page compression](#), after which free space is added to each new compressed page, dynamically adjusted up to the level set by `innodb_compression_pad_pct_max`. Zero disables checking of compression efficiency and adjusting padding.
    - See [InnoDB Page Compression: Configuring the Failure Threshold and Padding](#) for more information.
  - **Commandline:**

```
--innodb-compression-failure-threshold-pct=#
```
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**

numeric
  - **Default Value:**

5
  - **Range:**

0  
to  
100
  - **Introduced:** [MariaDB 10.0](#)
- 

## `innodb_compression_level`

- **Description:** Specifies the default level of compression for tables that use [InnoDB page compression](#).
    - Only a subset of InnoDB page compression algorithms support compression levels. If an InnoDB page compression algorithm does not support compression levels, then the compression level value is ignored.
    - The compression level can be set to any value between
      - 1
      - and
      - 9
      - . The default compression level is
      - 6
      - . The range goes from the fastest to the most compact, which means that
      - 1
      - is the fastest and
      - 9
      - is the most compact.
    - See [InnoDB Page Compression: Configuring the Default Compression Level](#) for more information.
  - **Commandline:**

```
--innodb-compression-level=#
```
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**

numeric
  - **Default Value:**

6
  - **Range:**

1  
to  
9
  - **Introduced:** [MariaDB 10.0](#)
- 

## `innodb_compression_pad_pct_max`

- **Description:** The maximum percentage of reserved free space within each compressed page for tables that use [InnoDB page compression](#). Reserved free space is used when the page's data is reorganized and might be recompressed. Only used when `innodb_compression_failure_threshold_pct` is not zero, and the rate of compression failures exceeds its setting.

- See [InnoDB Page Compression: Configuring the Failure Threshold and Padding](#) for more information.

- **Commandline:**

```
--innodb-compression-pad-pct-max=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    50
```

- **Range:**

```
    0
```

```
    to
```

```
    75
```

- **Introduced:** [MariaDB 10.0](#)

---

## innodb\_concurrency\_tickets

- **Description:** Number of times a newly-entered thread can enter and leave InnoDB until it is again subject to the limitations of [innodb\\_thread\\_concurrency](#) and may possibly be queued. Deprecated and ignored from [MariaDB 10.5.5](#).

- **Commandline:**

```
--innodb-concurrency-tickets=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    ◦
```

```
        0  
        (>= MariaDB 10.5.5 )
```

```
    ◦
```

```
        5000  
        (<= MariaDB 10.5.4 )
```

- **Range:**

```
    1
```

```
    to
```

```
    18446744073709551615
```

- **Deprecated:** [MariaDB 10.5.5](#)

- **Removed:** [MariaDB 10.6.0](#)

---

## innodb\_corrupt\_table\_action

- **Description:** What action to perform when a corrupt table is found. XtraDB only.

- When set to

```
    assert  
    , the default, XtraDB will intentionally crash the server when it detects corrupted data in a single-table tablespace, with an assertion failure.
```

- When set to

```
    warn  
    , it will pass corruption as corrupt table instead of crashing, and disable all further I/O (except for deletion) on the table file.
```

- If set to

```
    salvage  
    , read access is permitted, but corrupted pages are ignored. innodb\_file\_per\_table must be enabled for this option. Previously named
```

```
    innodb_pass_corrupt_table
```

◦ Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

```
    innodb-corrupt-table-action=value
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
enumeration

- **Default Value:**

- assert  
(<= MariaDB 10.1 )
- deprecated  
(<= MariaDB 10.2.6 )

- **Valid Values:**

- deprecated
- ,
- assert
- ,
- warn
- ,
- salvage  
(>= MariaDB 10.2.6 )
- assert
- ,
- warn
- ,
- salvage  
(<= MariaDB 10.1 )

- **Deprecated:** MariaDB 10.2.6

- **Removed:** MariaDB 10.3.0

---

## innodb\_data\_file\_path

- **Description:** Individual InnoDB data files, paths and sizes. The value of `innodb_data_home_dir` is joined to each path specified by `innodb_data_file_path` to get the full directory path. If `innodb_data_home_dir` is an empty string, absolute paths can be specified here. A file size is specified with K for kilobytes, M for megabytes and G for gigabytes, and whether or not to autoextend the data file is also specified.

- **Commandline:**

```
--innodb-data-file-path=name
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**

numeric

- **Default Value:**

```
ibdata1:12M:autoextend  
(from MariaDB 10.0 ),  
ibdata1:10M:autoextend  
(before MariaDB 10.0 )
```

---

## innodb\_data\_home\_dir

- **Description:** Directory path for all InnoDB data files in the shared tablespace (assuming `innodb_file_per_table` is not enabled). File-specific information can be added in `innodb_data_file_path`, as well as absolute paths if `innodb_data_home_dir` is set to an empty string.

- **Commandline:**

```
--innodb-data-home-dir=path
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
directory name

- **Default Value:**

The MariaDB data directory

## `innodb_deadlock_detect`

- **Description:** By default, the InnoDB deadlock detector is enabled. If set to off, deadlock detection is disabled and MariaDB will rely on `innodb_lock_wait_timeout` instead. This may be more efficient in systems with high concurrency as deadlock detection can cause a bottleneck when a number of threads have to wait for the same lock.

- **Commandline:**

```
--innodb-deadlock-detect
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

1

- **Introduced:** [MariaDB 10.2.6](#)

## `innodb_deadlock_report`

- **Description:** How to report deadlocks (if `innodb_deadlock_detect=ON` ).

◦

off

: Do not report any details of deadlocks.

◦

basic

: Report transactions and waiting locks.

◦

full

: Default. Report transactions, waiting locks and blocking locks.

- **Commandline:**

```
--innodb-deadlock-report=val
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

full

- **Valid Values:**

off

,

basic

,

full

- **Introduced:** [MariaDB 10.6.0](#)

## `innodb_default_page_encryption_key`

- **Description:** Encryption key used for page encryption.

◦ See [Data-at-Rest Encryption](#) and [InnoDB Encryption Keys](#) for more information.

- **Commandline:**

```
--innodb-default-page-encryption-key=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

1  
to  
255

- **Introduced:** MariaDB 10.1.3

- **Removed:** MariaDB 10.1.4

---

## innodb\_default\_encryption\_key\_id

- **Description:** ID of encryption key used by default to encrypt InnoDB tablespaces.

- See [Data-at-Rest Encryption](#) and [InnoDB Encryption Keys](#) for more information.

- **Commandline:**

--innodb-default-encryption-key-id=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

1  
to  
4294967295

- **Introduced:** MariaDB 10.1.4

---

## innodb\_default\_row\_format

- **Description:** Specifies the default [row format](#) to be used for InnoDB tables. The compressed row format cannot be set as the default.

- See [InnoDB Row Formats Overview: Default Row Format](#) for more information.

- **Commandline:**

--innodb-default-row-format=value

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

dynamic  
(>= MariaDB 10.2.2 ),  
compact  
(>= MariaDB 10.1.32 )

- **Valid Values:**

redundant  
,  
compact  
or  
dynamic

- **Introduced:** MariaDB 10.2.2 , Mariadb 10.1.32

---

## innodb\_defragment

- **Description:** When set to

```
1  
(the default is  
0
```

), InnoDB defragmentation is enabled. When set to FALSE, all existing defragmentation will be paused and new defragmentation commands will fail. Paused defragmentation commands will resume when this variable is set to true again. See [Defragmenting InnoDB Tablespaces](#).

- **Commandline:**

```
--innodb-defragment={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
boolean
```

- **Default Value:**

```
OFF
```

- **Introduced:** [MariaDB 10.1.1](#)
- 

### `innodb_defragment_fill_factor`

- **Description:** . Indicates how full defragmentation should fill a page. Together with `innodb_defragment_fill_factor_n_recs` ensures defragmentation won't pack the page too full and cause page split on the next insert on every page. The variable indicating more defragmentation gain is the one effective. See [Defragmenting InnoDB Tablespaces](#).

- **Commandline:**

```
--innodb-defragment-fill-factor=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
double
```

- **Default Value:**

```
0.9
```

- **Range:**

```
0.7
```

```
to
```

```
1
```

- **Introduced:** [MariaDB 10.1.1](#)
- 

### `innodb_defragment_fill_factor_n_recs`

- **Description:** Number of records of space that defragmentation should leave on the page. This variable, together with `innodb_defragment_fill_factor`, is introduced so defragmentation won't pack the page too full and cause page split on the next insert on every page. The variable indicating more defragmentation gain is the one effective. See [Defragmenting InnoDB Tablespaces](#).

- **Commandline:**

```
--innodb-defragment-fill-factor-n-recs=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
numeric
```

- **Default Value:**

```
20
```

- **Range:**

```
1
```

```
to
```

```
100
```

- **Introduced:** [MariaDB 10.1.1](#)

## innodb\_defragment\_frequency

- **Description:** Maximum times per second for defragmenting a single index. This controls the number of times the defragmentation thread can request X\_LOCK on an index. The defragmentation thread will check whether 1/defragment\_frequency (s) has passed since it last worked on this index, and put the index back in the queue if not enough time has passed. The actual frequency can only be lower than this given number. See [Defragmenting InnoDB Tablespaces](#).

- **Commandline:**

```
--innodb-defragment-frequency=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

integer

- **Default Value:**

40

- **Range:**

1  
to  
1000

- **Introduced:** [MariaDB 10.1.1](#)

## innodb\_defragment\_n\_pages

- **Description:** Number of pages considered at once when merging multiple pages to defragment. See [Defragmenting InnoDB Tablespaces](#).

- **Commandline:**

```
--innodb-defragment-n-pages=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

7

- **Range:**

2  
to  
32

- **Introduced:** [MariaDB 10.1.1](#)

## innodb\_defragment\_stats\_accuracy

- **Description:** Number of defragment stats changes there are before the stats are written to persistent storage. Defaults to zero, meaning disable defragment stats tracking. See [Defragmenting InnoDB Tablespaces](#).

- **Commandline:**

```
--innodb-defragment-stats-accuracy=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to  
4294967295

- **Introduced:** MariaDB 10.1.1
- 

### innodb\_dict\_size\_limit

- **Description:** Size in bytes of a soft limit the memory used by tables in the data dictionary. Once this limit is reached, XtraDB will attempt to remove unused entries. If set to

0

, the default and standard InnoDB behavior, there is no limit to memory usage. Removed in MariaDB 10.0 /XtraDB 5.6 and replaced by MySQL 5.6's new [table\\_definition\\_cache](#) implementation.

- **Commandline:**

innodb-dict-size-limit=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Default Value - 32 bit:**

2147483648

- **Default Value - 64 bit:**

9223372036854775807

- **Removed:** MariaDB 10.0 - replaced by MySQL 5.6's [table\\_definition\\_cache](#) implementation.
- 

### innodb\_disable\_sort\_file\_cache

- **Description:** If set to

1

(

0

is default), the operating system file system cache for merge-sort temporary files is disabled.

- **Commandline:**

--innodb-disable-sort-file-cache={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### innodb\_disallow\_writes

- **Description:** Tell InnoDB to stop any writes to disk.

- **Commandline:** None

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** MariaDB 10.1.3
- 

## innodb\_doublewrite

- **Description:** If set to

1

, the default, to improve fault tolerance InnoDB first stores data to a [doublewrite buffer](#) before writing it to data file. Disabling will provide a marginal performance improvement.

- **Commandline:**

```
--innodb-doublewrite  
,  
--skip-innodb-doublewrite
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

## innodb\_doublewrite\_file

- **Description:** The absolute or relative path and filename to a dedicated tablespace for the [doublewrite buffer](#). In heavy workloads, the doublewrite buffer can impact heavily on the server, and moving it to a different drive will reduce contention on random reads. Since the doublewrite buffer is mostly sequential writes, a traditional HDD is a better choice than SSD. This Percona XtraDB variable has not been ported to XtraDB 5.6.

- **Commandline:**

```
innodb-doublewrite-file=filename
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

filename

- **Default Value:**

NULL

- **Removed:** MariaDB 10.0
- 

## innodb\_empty\_free\_list\_algorithm

- **Description:** XtraDB 5.6.13-61 introduced an algorithm to assist with reducing mutex contention when the buffer pool free list is empty, controlled by this variable. If set to

backoff

, the default until [MariaDB 10.1.24](#), the new algorithm will be used. If set to

legacy

, the original InnoDB algorithm will be used. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades. See [#1651657](#) for the reasons this was changed back to

legacy

in XtraDB 5.6.36-82.0. When upgrading from 10.0 to 10.1 ( $\geq 10.1.24$ ), for large buffer pools the default will remain

backoff

, while for small ones it will be changed to

legacy

- **Commandline:**

```
innodb-empty-free-list-algorithm=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

- - deprecated  
( $\geq$  MariaDB 10.2.6)
  - legacy  
( $\geq$  MariaDB 10.1.24)
  - backoff  
( $\leq$  MariaDB 10.1.23)

- **Valid Values:**

- - deprecated
  - ,
  - backoff
  - ,
  - legacy  
( $\geq$  MariaDB 10.2.6)
- - backoff
  - ,
  - legacy  
( $\leq$  MariaDB 10.1)

- **Introduced:** MariaDB 10.0.9

- **Deprecated:** MariaDB 10.2.6

- **Removed:** MariaDB 10.3.0

---

## innodb\_enable\_unsafe\_group\_commit

- **Description:** Unneeded after XtraDB 1.0.5. If set to

- 0
  - , the default, InnoDB will keep transactions between the transaction log and [binary log](#)s in the same order. Safer, but slower. If set to 1
  - , transactions can be group-committed, but there is no guarantee of the order being kept, and a small risk of the two logs getting out of sync. In write-intensive environments, can lead to a significant improvement in performance.

- **Commandline:**

```
--innodb-enable-unsafe-group-commit
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

1

- **Removed:** Not needed after XtraDB 1.0.5

---

## innodb\_encrypt\_log

- **Description:** Enables encryption of the [InnoDB redo log](#). This also enables encryption of some temporary files created internally by InnoDB, such as those used for merge sorts and row logs.

- See [Data-at-Rest Encryption](#) and [InnoDB / XtraDB Enabling Encryption: Enabling Encryption for Redo Log](#) for more information.

- **Commandline:**

```
--innodb-encrypt-log
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** MariaDB 10.1.3
- 

## innodb\_encrypt\_tables

- **Description:** Enables automatic encryption of all InnoDB tablespaces.

◦

OFF

- Disables table encryption for all new and existing tables that have the

ENCRYPTED

table option set to

DEFAULT

◦

ON

- Enables table encryption for all new and existing tables that have the

ENCRYPTED

table option set to

DEFAULT

, but allows unencrypted tables to be created.

◦

FORCE

- Enables table encryption for all new and existing tables that have the

ENCRYPTED

table option set to

DEFAULT

, and doesn't allow unencrypted tables to be created (CREATE TABLE ... ENCRYPTED=NO will fail).

- See [Data-at-Rest Encryption](#) and [InnoDB / XtraDB Enabling Encryption: Enabling Encryption for Automatically Encrypted Tablespaces](#) for more information.

- **Commandline:**

--innodb-encrypt-tables={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Valid Values:**

ON

,

OFF

,

FORCE

(from MariaDB 10.1.4 )

- **Introduced:** MariaDB 10.1.3
- 

## innodb\_encrypt\_temporary\_tables

- **Description:** Enables automatic encryption of the InnoDB temporary tablespace .

- See [Data-at-Rest Encryption](#) and [InnoDB / XtraDB Enabling Encryption: Enabling Encryption for Temporary Tablespaces](#) for more information.

- **Commandline:**

--innodb-encrypt-temporary-tables={0|1}

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**

OFF

- **Valid Values:**

ON  
,  
OFF

- **Introduced:** MariaDB 10.2.26 , MariaDB 10.3.17 , MariaDB 10.4.7
- 

### innodb\_encryption\_rotate\_key\_age

- **Description:** Re-encrypt in background any page having a key older than this number of key versions. When setting up encryption, this variable must be set to a non-zero value. Otherwise, when you enable encryption through

#### [innodb\\_encrypt\\_tables](#)

MariaDB won't be able to automatically encrypt any unencrypted tables.

- See [Data-at-Rest Encryption](#) and [InnoDB / XtraDB Encryption Keys: Key Rotation](#) for more information.

- **Commandline:**

--innodb-encryption-rotate-key-age=#

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0  
to  
4294967295

- **Introduced:** MariaDB 10.1.3
- 

### innodb\_encryption\_rotation\_iops

- **Description:** Use this many iops for background key rotation operations performed by the background encryption threads.

- See [Data-at-Rest Encryption](#) and [InnoDB / XtraDB Encryption Keys: Key Rotation](#) for more information.

- **Commandline:**

--innodb-encryption-rotation\_iops=#

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**

numeric

- **Default Value:**

100

- **Range:**

0  
to  
4294967295

- **Introduced:** MariaDB 10.1.3
-

## innodb\_encryption\_threads

- **Description:** Number of background encryption threads performing background key rotation and [scrubbing](#). When setting up encryption, this variable must be set to a non-zero value. Otherwise, when you enable encryption through [innodb\\_encrypt\\_tables](#) MariaDB won't be able to automatically encrypt any unencrypted tables. Recommended never be set higher than 255.
  - See [Data-at-Rest Encryption](#) and [InnoDB Background Encryption Threads](#) for more information.
- **Commandline:**

```
--innodb-encryption-threads=#
```
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**

```
numeric
```
- **Default Value:**

```
0
```
- **Range:**
  - ```
0
to
4294967295
(<= MariaDB 10.1.45 , MariaDB 10.2.32 , MariaDB 10.3.23 , MariaDB 10.4.13 , MariaDB 10.5.3 )
```
  - ```
0
to
255
(>= MariaDB 10.1.46 , MariaDB 10.2.33 , MariaDB 10.3.24 , MariaDB 10.4.14 , MariaDB 10.5.4 )
```
- **Introduced:** [MariaDB 10.1.3](#)

---

## innodb\_extra\_rsegments

- **Description:** Removed in XtraDB 5.5 and replaced by [innodb\\_rollback\\_segments](#). Usually there is one rollback segment protected by single mutex, a source of contention in high write environments. This option specifies a number of extra user rollback segments. Changing the default will make the data readable by XtraDB only, and is incompatible with InnoDB. After modifying, the server must be slow-shutdown. If there is existing data, it must be dumped before changing, and re-imported after the change has taken effect.
- **Commandline:**

```
--innodb-extra-rsegments=#
```
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**

```
numeric
```
- **Default Value:**

```
0
```
- **Range:**

```
0
to
126
```
- **Removed:** XtraDB 5.5 - replaced by [innodb\\_rollback\\_segments](#)

---

## innodb\_extra\_undoslots

- **Description:** Usually, InnoDB has 1024 undo slots in its rollback segment, so 1024 transactions can run in parallel. New transactions will fail if all slots are used. Setting this variable to
  - 1expands the available undo slots to 4072. Not recommended unless you get the
  - Warning: cannot find a free slot for an undo log
  - error in the error log, as it makes data files unusable for ibbackup, or MariaDB servers not run with this option. See also [undo log](#).
- **Commandline:**

```
--innodb-extra-undoslots={0|1}
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
OFF

- **Removed:** XtraDB 5.5
- 

### innodb\_fake\_changes

- **Description:** From MariaDB 5.5 until MariaDB 10.1, XtraDB-only option that enables the fake changes feature. In replication, setting up or restarting a replica can cause a replication reads to perform more slowly, as MariaDB is single-threaded and needs to read the data before it can execute the queries. This can be speeded up by prefetching threads to warm the server, replaying the statements and then rolling back at commit. This however has an overhead from locking rows only then to undo changes at rollback. Fake changes attempts to reduce this overhead by reading the rows for INSERT, UPDATE and DELETE statements but not updating them. The rollback is then very fast with little or nothing to do. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades. Not present in MariaDB 10.3 and beyond.

- **Commandline:**

```
--innodb-fake-changes={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Deprecated:** MariaDB 10.2.6

- **Removed:** MariaDB 10.3.0
- 

### innodb\_fast\_checksum

- **Description:** Implements a more CPU efficient XtraDB checksum algorithm, useful for write-heavy loads with high I/O. If set to

1

on a server with tables that have been created with it set to

0

, reads will be slower, so tables should be recreated (dumped and reloaded). XtraDB will fail to start if set to

0

and there are tables created while set to

1

. Replaced with innodb\_checksum\_algorithm in MariaDB 10.0 /XtraDB 5.6.

- **Commandline:**

```
--innodb-fast-checksum={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Removed:** MariaDB 10.0 /XtraDB 5.6 - replaced with innodb\_checksum\_algorithm
- 

### innodb\_fast\_shutdown

- **Description:** The shutdown mode.

◦

0  
- InnoDB performs a slow shutdown, including full purge (before [MariaDB 10.3.6](#), not always, due to [MDEV-13603](#)) and change buffer merge. Can be very slow, even taking hours in extreme cases.

o 1  
- the default, InnoDB performs a fast shutdown, not performing a full purge or an insert buffer merge.

o 2  
, the [InnoDB redo log](#) is flushed and a cold shutdown takes place, similar to a crash. The resulting startup then performs crash recovery. Extremely fast, in cases of emergency, but risks corruption. Not suitable for upgrades between major versions!

o 3  
(from [MariaDB 10.3.6](#)) - active transactions will not be rolled back, but all changed pages will be written to data files. The active transactions will be rolled back by a background thread on a subsequent startup. The fastest option that will not involve [InnoDB redo log](#) apply on subsequent startup. See [MDEV-15832](#).

- **Commandline:**  
`--innodb-fast-shutdown[=#]`
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric
- **Default Value:**  
1
- **Range:**  
0  
to  
3  
( $\geq$  [MariaDB 10.3.6](#)),  
0  
to  
2  
( $\leq$  [MariaDB 10.3.5](#))

---

## innodb\_fatal\_semaphore\_wait\_threshold

- **Description:** In MariaDB, the fatal semaphore timeout is configurable. This variable sets the maximum number of seconds for semaphores to time out in InnoDB.
- **Commandline:**  
`--innodb-fatal-semaphore-wait-threshold=#`
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric
- **Default Value:**  
600
- **Range:**  
1  
to  
4294967295
- **Introduced:** [MariaDB 10.1.2](#)

---

## innodb\_file\_format

- **Description:** File format for new InnoDB tables. Can either be  
Antelope  
, the default and the original format, or  
Barracuda  
, which supports [compression](#). Note that this value is also used when a table is re-created with an [ALTER TABLE](#) which requires a table copy. See [XtraDB/InnoDB File Format](#) for more on the file formats. Removed in 10.3.1 and restored as a deprecated and unused variable in 10.4.3

for compatibility purposes.

- **Commandline:**

```
--innodb-file-format=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:**

◦

Barracuda  
( $\geq$  MariaDB 10.2.2 )  
◦  
Antelope  
( $\leq$  MariaDB 10.2.1 )

- **Valid Values:**

Antelope

,

Barracuda

- **Deprecated:** MariaDB 10.2

- **Removed:** MariaDB 10.3.1

- **Re-introduced:** MariaDB 10.4.3 (for compatibility purposes)

- **Removed:** MariaDB 10.6.0

## innodb\_file\_format\_check

- **Description:** If set to

1

, the default, InnoDB checks the shared tablespace file format tag. If this is higher than the current version supported by XtraDB/InnoDB (for example Barracuda when only Antelope is supported), XtraDB/InnoDB will not start. If it the value is not higher, XtraDB/InnoDB starts correctly and the innodb\_file\_format\_max value is set to this value. If innodb\_file\_format\_check is set to

0

, no checking is performed. See [XtraDB/InnoDB File Format](#) for more on the file formats.

- **Commandline:**

```
--innodb-file-format-check={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

- **Deprecated:** MariaDB 10.2

- **Removed:** MariaDB 10.3.1

## innodb\_file\_format\_max

- **Description:** The highest XtraDB/InnoDB file format. This is set to the value of the file format tag in the shared tablespace on startup (see [innodb\\_file\\_format\\_check](#)). If the server later creates a higher table format, innodb\_file\_format\_max is set to that value. See [XtraDB/InnoDB File Format](#) for more on the file formats.

- **Commandline:**

```
--innodb-file-format-max=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:**

Antelope

- **Valid Values:**

Antelope  
,  
Barracuda

- **Deprecated:** [MariaDB 10.2](#)

- **Removed:** [MariaDB 10.3.1](#)

---

## innodb\_file\_per\_table

- **Description:** If set to

ON  
, then new [InnoDB](#) tables are created with their own [InnoDB file-per-table tablespaces](#). If set to  
OFF  
, then new tables are created in the [InnoDB system tablespace](#) instead. [Page compression](#) is only available with file-per-table  
tablespaces. Note that this value is also used when a table is re-created with an [ALTER TABLE](#) which requires a table copy.

- **Commandline:**

--innodb-file-per-table

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

---

## innodb\_fill\_factor

- **Description:** Percentage of B-tree page filled during bulk insert (sorted index build). Used as a hint rather than an absolute value. Setting to

70  
, for example, reserves 30% of the space on each B-tree page for the index to grow in future.

- **Commandline:**

--innodb-fill-factor=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value :**

100

- **Range:**

10

to

100

- **Introduced:** [MariaDB 10.2.2](#)

---

## innodb\_flush\_log\_at\_timeout

- **Description:** Interval in seconds to write and flush the [InnoDB redo log](#). Before MariaDB 10, this was fixed at one second, which is still the default, but this can now be changed. It's usually increased to reduce flushing and avoid impacting performance of binary log group commit.

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0  
to  
2700

---

## innodb\_flush\_log\_at\_trx\_commit

- **Description:** Set to

1

, along with [sync\\_binlog=1](#) for the greatest level of fault tolerance. The value of [innodb\\_use\\_global\\_flush\\_log\\_at\\_trx\\_commit](#) determines whether this variable can be reset with a SET statement or not.

◦

1

The default, the log buffer is written to the [InnoDB redo log](#) file and a flush to disk performed after each transaction. This is required for full ACID compliance.

◦

0

Nothing is done on commit; rather the log buffer is written and flushed to the [InnoDB redo log](#) once a second. This gives better performance, but a server crash can erase the last second of transactions.

◦

2

The log buffer is written to the [InnoDB redo log](#) after each commit, but flushing takes place once a second. Performance is slightly better, but a OS or power outage can cause the last second's transactions to be lost.

◦

3

Emulates [MariaDB 5.5 group commit](#) (3 syncs per group commit). See [Binlog group commit and innodb\\_flush\\_log\\_at\\_trx\\_commit](#). This option has not been working correctly since 10.2 and may be removed in future, see <https://github.com/MariaDB/server/pull/1873>

- **Commandline:**

--innodb-flush-log-at-trx-commit[=#]

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enumeration

- **Default Value:**

1

- **Valid Values:**

0

,

1

,

2

or

3

---

## innodb\_flush\_method

- **Description:** [InnoDB](#) flushing method. Windows always uses `async_unbuffered` and this variable then has no effect. On Unix, before [MariaDB 10.6.0](#) , by default `fsync()` is used to flush data and logs. Adjusting this variable can give performance improvements, but behavior differs widely on different filesystems, and changing from the default has caused problems in some situations, so test and benchmark carefully before adjusting. In MariaDB, Windows recognises and correctly handles the Unix methods, but if none are specified it uses own default - unbuffered write (analog of `O_DIRECT`) + syncs (e.g `FileFlushBuffers()`) for all files.

◦

`O_DSYNC`

- `O_DSYNC` is used to open and flush logs, and `fsync()` to flush the data files.

◦

`O_DIRECT`

- `O_DIRECT` or `directio()`, is used to open data files, and `fsync()` to flush data and logs. Default on Unix from [MariaDB 10.6.0](#) .

◦

`fsync`

- Default on Unix until [MariaDB 10.5](#). Can be specified directly, but if the variable is unset on Unix, fsync() will be used by default.
- o
  - O\_DIRECT\_NO\_FSYNC
    - introduced in [MariaDB 10.0](#). Uses O\_DIRECT during flushing I/O, but skips fsync() afterwards. Not suitable for XFS filesystems. Generally not recommended over O\_DIRECT, as does not get the benefit of [innodb\\_use\\_native\\_aio=ON](#).
  - ALL\_O\_DIRECT
    - introduced in [MariaDB 5.5](#) and available with XtraDB only. Uses O\_DIRECT for opening both data and logs and fsync() to flush data but not logs. Use with large InnoDB files only, otherwise may cause a performance degradation. Set [innodb\\_log\\_block\\_size](#) to 4096 on ext4 filesystems. This is the default log block size on ext4 and will avoid unaligned AIO/DIO warnings.
  - unbuffered
    - Windows-only default
  - async\_unbuffered
    - Windows-only, alias for unbuffered
  - normal
    - Windows-only, alias for fsync

- **Commandline:**

```
--innodb-flush-method=name
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
enumeration
(=> MariaDB 10.3.7 ),
string
(<= MariaDB 10.3.6 )
```

- **Default Value:**

- o
  - O\_DIRECT
    - (Unix, >= [MariaDB 10.6.0](#) )
  - fsync
    - (Unix, >= [MariaDB 10.3.7](#) , <= [MariaDB 10.5](#) )
- Not set (<= [MariaDB 10.3.6](#) )

- **Valid Values:**

- Unix:
  - fsync
  - ,
  - O\_DSYNC
  - ,
  - O\_DIRECT
  - ,
  - O\_DIRECT\_NO\_FSYNC
  - ,
  - ALL\_O\_DIRECT
    - (>= [MariaDB 5.5](#) to <= [MariaDB 10.1](#) , XtraDB only)
- Windows:
  - unbuffered
  - ,
  - async\_unbuffered
  - ,
  - normal

## [innodb\\_flush\\_neighbor\\_pages](#)

- **Description:** Determines whether, when dirty pages are flushed to the data file, neighboring pages in the data file are flushed at the same time. If set to

```
none
, the feature is disabled. If set to
area
```

, the default, the standard InnoDB behavior is used. For each page to be flushed, dirty neighboring pages are flushed too. If there's little head seek delay, such as SSD or large enough write buffer, one of the other two options may be more efficient. If set to

cont

, for each page to be flushed, neighboring contiguous blocks are flushed at the same time. Being contiguous, a sequential I/O is used, unlike the random I/O used in

area

. Replaced by [innodb\\_flush\\_neighbors](#) in MariaDB 10.0 /XtraDB 5.6.

- **Commandline:**

```
innodb-flush-neighbor-pages=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enumeration

- **Default Value:**

area

- **Valid Values:**

none

or

0

,

area

or

1

,

cont

or

2

- **Removed:** MariaDB 10.0 /XtraDB 5.6 - replaced by [innodb\\_flush\\_neighbors](#)

---

## innodb\_flush\_neighbors

- **Description:** Determines whether flushing a page from the [buffer pool](#) will flush other dirty pages in the same group of pages (extent). In high write environments, if flushing is not aggressive enough, it can fall behind resulting in higher memory usage, or if flushing is too aggressive, cause excess I/O activity. SSD devices, with low seek times, would be less likely to require dirty neighbor flushing to be set.

◦

1

: The default, flushes contiguous dirty pages in the same extent from the buffer pool.

◦

0

: No other dirty pages are flushed.

◦

2

: Flushes dirty pages in the same extent from the buffer pool.

- **Commandline:**

```
--innodb-flush-neighbors=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enumeration

- **Default Value:**

1

- **Valid Values:**

0

,

1

,

2

## innodb\_flush\_sync

- **Description:** If set to  
ON  
, the default, the [innodb\\_io\\_capacity](#) setting is ignored for I/O bursts occurring at checkpoints.
  - **Commandline:**  
--innodb-flush-sync={0|1}
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value :**  
ON
  - **Introduced:** [MariaDB 10.2.2](#)
- 

## innodb\_flushing\_avg\_loops

- **Description:** Determines how quickly adaptive flushing will respond to changing workloads. The value is the number of iterations that a previously calculated flushing state snapshot is kept. Increasing the value smooths and slows the rate that the flushing operations change, while decreasing it causes flushing activity to spike quickly in response to workload changes.
  - **Commandline:**  
--innodb-flushing-avg-loops=#
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
numeric
  - **Default Value:**  
30
  - **Range:**  
1  
to  
1000
- 

## innodb\_force\_load\_corrupted

- **Description:** Set to  
0  
by default, if set to  
1  
, InnoDB will be permitted to load tables marked as corrupt. Only use this to recover data you can't recover any other way, or in troubleshooting. Always restore to  
0  
when returning to regular use. Given that [MDEV-11412](#) in [MariaDB 10.5.4](#) aims to allow any metadata for a missing or corrupted table to be dropped, and given that [MDEV-17567](#) and [MDEV-25506](#) and related tasks made DDL operations crash-safe, the parameter no longer serves any purpose and was removed in [MariaDB 10.6.6](#).
- **Commandline:**  
--innodb-force-load-corrupted
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean
- **Default Value:**  
OFF
- **Removed:** [MariaDB 10.6.6](#)

## innodb\_force\_primary\_key

- **Description:** If set to

```
1  
(  
0  
is default) CREATE TABLEs without a primary or unique key where all keyparts are NOT NULL will not be accepted, and will return an error.
```

- **Commandline:**

```
--innodb-force-primary-key
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
boolean
```

- **Default Value:**

```
OFF
```

- **Introduced:** MariaDB 10.1.0
- 

## innodb\_force\_recovery

- **Description:** InnoDB crash recovery mode.

```
0  
is the default. The other modes are for recovery purposes only, and no data can be changed while another mode is active. Some queries relying on indexes are also blocked. See InnoDB Recovery Modes for more on mode specifics.
```

- **Commandline:**

```
--innodb-force-recovery=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
enumeration
```

- **Default Value:**

```
0
```

- **Range:**

```
0
```

```
to
```

```
6
```

---

## innodb\_foreground\_preflush

- **Description:** Before XtraDB 5.6.13-61.0, if the checkpoint age is in the sync preflush zone while a thread is writing to the [XtraDB redo log](#), it will try to advance the checkpoint by issuing a flush list flush batch if this is not already being done. XtraDB has enhanced page cleaner tuning, and may already be performing furious flushing, resulting in the flush simply adding unneeded mutex pressure. Instead, the thread now waits for the flushes to finish, and then has two options, controlled by this variable. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- - **exponential\_backoff**
    - thread sleeps while it waits for the flush list flush to occur. The sleep time randomly progressively increases, periodically reset to avoid runaway sleeps.
  - **sync\_preflush**
    - thread issues a flush list batch, and waits for it to complete. This is the same as is used when the page cleaner thread is not running.

- **Commandline:**

```
innodb-foreground-preflush=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

- - deprecated  
(>= MariaDB 10.2.6 )
  - exponential\_backoff  
(<= MariaDB 10.1 )

- **Valid Values:**

- - deprecated
  - ,
  - exponential\_backoff
  - ,
  - sync\_preflush  
(>= MariaDB 10.2.6 )
- - exponential\_backoff
  - ,
  - sync\_preflush  
(<= MariaDB 10.1 )

- **Introduced:** MariaDB 10.0.9

- **Deprecated:** MariaDB 10.2.6

- **Removed:** MariaDB 10.3.0

---

### innodb\_ft\_aux\_table

- **Description:** Diagnostic variable intended only to be set at runtime. It specifies the qualified name (for example

test/ft\_innodb  
) of an InnoDB table that has a FULLTEXT index , and after being set the INFORMATION\_SCHEMA tables INNODB\_FT\_INDEX\_TABLE , INNODB\_FT\_INDEX\_CACHE , INNODB\_FT\_CONFIG, INNODB\_FT\_DELETED , and INNODB\_FT\_BEING\_DELETED will contain search index information for the specified table.

- **Commandline:**

--innodb-ft-aux-table=value

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

---

### innodb\_ft\_cache\_size

- **Description:** Cache size available for a parsed document while creating an InnoDB FULLTEXT index .

- **Commandline:**

--innodb-ft-cache-size=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

8000000

---

### innodb\_ft\_enable\_diag\_print

- **Description:** If set to

1

, additional [full-text](#) search diagnostic output is enabled.

- **Commandline:**

```
--innodb_ft-enable-diag-print={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

## innodb\_ft\_enable\_stopword

- **Description:** If set to

1

, the default, a set of [stopwords](#) is associated with an InnoDB [FULLTEXT index](#) when it is created. The stopword list comes from the table set by the session variable [innodb\\_ft\\_user\\_stopword\\_table](#), if set, otherwise the global variable [innodb\\_ft\\_server\\_stopword\\_table](#), if that is set, or the [built-in list](#) if neither variable is set.

- **Commandline:**

```
--innodb_ft-enable-stopword={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

---

## innodb\_ft\_max\_token\_size

- **Description:** Maximum length of words stored in an InnoDB [FULLTEXT index](#). A larger limit will increase the size of the index, slowing down queries, but permit longer words to be searched for. In most normal situations, longer words are unlikely search terms.

- **Commandline:**

```
--innodb_ft-max-token-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

84

- **Range:**

10

to

252

---

## innodb\_ft\_min\_token\_size

- **Description:** Minimum length of words stored in an InnoDB [FULLTEXT index](#). A smaller limit will increase the size of the index, slowing down queries, but permit shorter words to be searched for. For data stored in a Chinese, Japanese or Korean [character set](#), a value of 1 should be specified to preserve functionality.

- **Commandline:**

```
--innodb_ft-min-token-size=#
```

- **Scope:** Global

- **Dynamic:** No
- **Data Type:** numeric

- **Default Value:**

3

- **Range:**

0  
to  
16

---

### innodb\_ft\_num\_word\_optimize

- **Description:** Number of words processed during each [OPTIMIZE TABLE](#) on an InnoDB [FULLTEXT index](#). To ensure all changes are incorporated, multiple OPTIMIZE TABLE statements could be run in case of a substantial change to the index.

- **Commandline:**

--innodb-ft-num-word-optimize=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

2000

---

### innodb\_ft\_result\_cache\_limit

- **Description:** Limit in bytes of the InnoDB [FULLTEXT index](#) query result cache per fulltext query. The latter stages of the full-text search are handled in memory, and limiting this prevents excess memory usage. If the limit is exceeded, the query returns an error.

- **Commandline:**

--innodb-ft-result-cache-limit=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

2000000000

- **Range:**

1000000  
to  
4294967295  
(<= MariaDB 10.2.18 , MariaDB 10.1.36 , MariaDB 10.0.36 )

- **Range:**

1000000  
to  
18446744073709551615  
(64-bit, >= MariaDB 10.2.19 , MariaDB 10.1.37 , MariaDB 10.0.37 )

---

### innodb\_ft\_server\_stopword\_table

- **Description:** Table name containing a list of stopwords to ignore when creating an InnoDB [FULLTEXT index](#), in the format db\_name/table\_name. The specified table must exist before this option is set, and must be an InnoDB table with a single column, a [VARCHAR](#) named VALUE. See also [innodb\\_ft\\_enable\\_stopword](#).

- **Commandline:**

--innodb-ft-server-stopword-table=db\_name/table\_name

- **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
string
  - **Default Value:** Empty
- 

### innodb\_ft\_sort\_pll\_degree

- **Description:** Number of parallel threads used when building an InnoDB [FULLTEXT index](#). See also [innodb\\_sort\\_buffer\\_size](#).
- **Commandline:**

```
--innodb-ft-sort-pll-degree=#
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**

2

- **Range:**

1  
to  
32

---

### innodb\_ft\_total\_cache\_size

- **Description:** Total memory allocated for the cache for all InnoDB [FULLTEXT index](#) tables. A force sync is triggered if this limit is exceeded.
- **Commandline:**

```
--innodb-ft-total-cache-size=#
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**

640000000

- **Range:**

32000000  
to  
1600000000

- **Introduced:** [MariaDB 10.0.9](#)
- 

### innodb\_ft\_user\_stopword\_table

- **Description:** Table name containing a list of stopwords to ignore when creating an InnoDB [FULLTEXT index](#), in the format db\_name/table\_name. The specified table must exist before this option is set, and must be an InnoDB table with a single column, a [VARCHAR](#) named VALUE. See also [innodb\\_ft\\_enable\\_stopword](#).

- **Commandline:**

```
--innodb-ft-user-stopword-table=db_name/table_name
```

- **Scope:** Session
- **Dynamic:** Yes
- **Data Type:**  
string

- **Default Value:** Empty
- 

### innodb\_ibuf\_accel\_rate

- **Description:** Allows the insert buffer activity to be adjusted. The following formula is used: [real activity] = [default activity] \* (innodb\_io\_capacity/100) \* (innodb\_ibuf\_accel\_rate/100). As innodb\_ibuf\_accel\_rate is increased from its default value of 100, the lowest setting, insert buffer activity is increased. See also [innodb\\_io\\_capacity](#). This Percona XtraDB variable has not been ported to XtraDB 5.6.

- **Commandline:**

```
innodb_ibuf_accel_rate=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

100

- **Range:**

100  
to  
999999999

- **Removed:** MariaDB 10.0
- 

### innodb\_ibuf\_active\_contract

- **Description:** Specifies whether the insert buffer can be processed before it's full. If set to

0

, the standard InnoDB method is used, and the buffer is not processed until it's full. If set to

1

, the default, the insert buffer can be processed before it is full. This Percona XtraDB variable has not been ported to XtraDB 5.6.

- **Commandline:**

```
innodb_ibuf_active_contract=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0

to

1

- **Removed:** MariaDB 10.0
- 

### innodb\_ibuf\_max\_size

- **Description:** Maximum size in bytes of the insert buffer. Defaults to half the size of the [buffer pool](#) so you may want to reduce if you have a very large buffer pool. If set to

0

, the insert buffer is disabled, which will cause all secondary index updates to be performed synchronously, usually at a cost to performance. This Percona XtraDB variable has not been ported to XtraDB 5.6.

- **Commandline:**

```
innodb_ibuf_max_size=#
```

- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
    numeric
  - **Default Value:** 1/2 the size of the InnoDB buffer pool
  - **Range:**  
    0  
    to 1/2 the size of the InnoDB buffer pool
  - **Removed:** [MariaDB 10.0](#)
- 

### innodb\_idle\_flush\_pct

- **Description:** Up to what percentage of dirty pages should be flushed when innodb finds it has spare resources to do so. Has had no effect since merging InnoDB 5.7 from mysql-5.7.9 ([MariaDB 10.2.2](#)). Deprecated in [MariaDB 10.2.37](#), [MariaDB 10.3.28](#), [MariaDB 10.4.18](#) and removed in [MariaDB 10.5.9](#).
  - **Commandline:**  
    --innodb-idle-flush-pct=#
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    numeric
  - **Default Value:**  
    100
  - **Range:**  
    0  
    to  
    100
  - **Introduced:** [MariaDB 10.1.2](#)
  - **Deprecated:** [MariaDB 10.2.37](#), [MariaDB 10.3.28](#), [MariaDB 10.4.18](#)
  - **Removed:** [MariaDB 10.5.9](#)
- 

### innodb\_immediate\_scrub\_data\_uncompressed

- **Description:** Enable scrubbing of data. See [Data Scrubbing](#).
  - **Commandline:**  
    --innodb-immediate-scrub-data-uncompressed={0|1}
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    boolean
  - **Default Value:**  
    OFF
  - **Introduced:** [MariaDB 10.1.3](#)
- 

### innodb\_import\_table\_from\_xtrabackup

- **Description:** If set to  
    1  
    , permits importing of .ibd files exported with the [XtraBackup](#) --export option. Previously named  
    innodb\_expand\_import  
    . Removed in [MariaDB 10.0](#) /XtraDB 5.6 and replaced with MySQL 5.6's transportable tablespaces.
- **Commandline:**

```
innodb-import-table-from-xtrabackup=#
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
    numeric

- **Default Value:**

```
0
```

- **Range:**

```
0
```

```
to
```

```
1
```

- **Removed:** MariaDB 10.0

## innodb\_instant\_alter\_column\_allowed

- **Description:**

- If a table is altered using ALGORITHM=INSTANT, it can force the table to use a non-canonical format: A hidden metadata record at the start of the clustered index is used to store each column's DEFAULT value. This makes it possible to add new columns that have default values without rebuilding the table. Starting with MariaDB 10.4, a BLOB in the hidden metadata record is used to store column mappings. This makes it possible to drop or reorder columns without rebuilding the table. This also makes it possible to add columns to any position or drop columns from any position in the table without rebuilding the table. If a column is dropped without rebuilding the table, old records will contain garbage in that column's former position, and new records will be written with NULL values, empty strings, or dummy values.
- This is generally not a problem. However, there may be cases where you want to avoid putting a table into this format. For example, to ensure that future UPDATE operations after an ADD COLUMN will be performed in-place, to reduce write amplification. (Instantly added columns are essentially always variable-length.) Also avoid bugs similar to MDEV-19916, or to be able to export tables to older versions of the server.
- This variable has been introduced as a result, with the following values:
  - never  
(0): Do not allow instant add/drop/reorder, to maintain format compatibility with MariaDB 10.x and MySQL 5.x. If the table (or partition) is not in the canonical format, then any ALTER TABLE (even one that does not involve instant column operations) will force a table rebuild.
  - add\_last  
(1, default in 10.3): Store a hidden metadata record that allows columns to be appended to the table instantly ( MDEV-11369 ). In 10.4 or later, if the table (or partition) is not in this format, then any ALTER TABLE (even one that does not involve column changes) will force a table rebuild.
  - add\_drop\_reorder  
(2, default): From MariaDB 10.4 only. Like 'add\_last', but allow the metadata record to store a column map, to support instant add/drop/reorder of columns.

- **Commandline:**

```
--innodb-instant-alter-column-allowed=value
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**

```
enum
```

- **Valid Values:**

- <= MariaDB 10.3 :

```
never  
,  
add_last
```

- >= MariaDB 10.4 :

```
never  
,  
add_last  
,  
add_drop_reorder
```

- **Default Value:**

- <= MariaDB 10.3 :

```
add_last
```

◦ >= MariaDB 10.4 :  
add\_drop\_reorder

- **Introduced:** MariaDB 10.3.23 , MariaDB 10.4.13 , MariaDB 10.5.3
- 

### innodb\_instrument\_semaphores

- **Description:** Enable semaphore request instrumentation. This could have some effect on performance but allows better information on long semaphore wait problems.

- **Commandline:**

--innodb-instrument-semaphores={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** MariaDB 10.1.3

- **Deprecated:** MariaDB 10.2.5 (treated as if

OFF

)

- **Removed:** MariaDB 10.3.0
- 

### innodb\_io\_capacity

- **Description:** Limit on I/O activity for InnoDB background tasks, including merging data from the insert buffer and flushing pages. Should be set to around the number of I/O operations per second that system can handle, based on the type of drive/s being used. You can also set it higher when the server starts to help with the extra workload at that time, and then reduce for normal use. Ideally, opt for a lower setting, as at higher value data is removed from the buffers too quickly, reducing the effectiveness of caching. See also [innodb\\_flush\\_sync](#) .

◦ See [InnoDB Page Flushing: Configuring the InnoDB I/O Capacity](#) for more information.

- **Commandline:**

--innodb-io-capacity=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

200

- **Range:**

100

to

18446744073709551615

( $2^{64}-1$ )

---

### innodb\_io\_capacity\_max

- **Description:** Upper limit to which InnoDB can extend [innodb\\_io\\_capacity](#) in case of emergency. See [InnoDB Page Flushing: Configuring the InnoDB I/O Capacity](#) for more information.

- **Commandline:**

--innodb-io-capacity-max=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

2000  
or twice `innodb_io_capacity`, whichever is higher.

- **Range :**

100  
to  
18446744073709551615  
( $2^{64}-1$ )

---

## `innodb_kill_idle_transaction`

- **Description:** Time in seconds before killing an idle XtraDB transaction. If set to

0

(the default), the feature is disabled. Used to prevent accidental user locks. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0  
to  
9223372036854775807

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

---

## `innodb_large_prefix`

- **Description:** If set to

1

, tables that use specific `row formats` are permitted to have index key prefixes up to 3072 bytes (for 16k pages, `smaller otherwise`). If not set, the limit is 767 bytes.

◦ This applies to the

`DYNAMIC`

and

`COMPRESSED`

`row formats`.

◦ Removed in 10.3.1 and restored as a deprecated and unused variable in 10.4.3 for compatibility purposes.

- **Commandline:**

`--innodb-large-prefix`

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

◦

`ON`  
( $\geq$  [MariaDB 10.2.2](#))

◦

`OFF`  
( $\leq$  [MariaDB 10.2.1](#))

- **Deprecated:** [MariaDB 10.2](#)

- **Removed:** [MariaDB 10.3.1](#)

- **Re-introduced:** [MariaDB 10.4.3](#) (for compatibility purposes)

- **Removed:** MariaDB 10.6.0
- 

## innodb\_lazy\_drop\_table

- **Description:** Deprecated and removed in XtraDB 5.6. `DROP TABLE` processing can take a long time when `innodb_file_per_table` is set to 1 and there's a large `buffer pool`. If

```
innodb_lazy_drop_table  
is set to  
1  
(  
0
```

is default), XtraDB attempts to optimize `DROP TABLE` processing by deferring the dropping of related pages from the `buffer pool` until there is time, only initially marking them.

- **Commandline:**

```
innodb-lazy-drop-table={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

0

- **Deprecated:** XtraDB 5.5.30-30.2

- **Removed:** MariaDB 10.0.0
- 

## innodb\_lock\_schedule\_algorithm

- **Description:** Specifies the algorithm that InnoDB/XtraDB uses to decide which of the waiting transactions should be granted the lock once it has been released. The possible values are:

FCFS  
(First-Come-First-Served) where locks are granted in the order they appear in the lock queue and  
VATS  
(Variance-Aware-Transaction-Scheduling) where locks are granted based on the Eldest-Transaction-First heuristic. Note that  
VATS  
should not be used with `Galera`. From [MariaDB 10.1.30](#), InnoDB will refuse to start if  
VATS  
is used with Galera. From [MariaDB 10.2](#),  
VATS  
is default, but from [MariaDB 10.2.12](#), the value will be changed to  
FCFS  
and a warning produced when using Galera.

- **Commandline:**

```
--innodb-lock-schedule-algorithm=#
```

- **Scope:** Global

- **Dynamic:** No (>= [MariaDB 10.2.12](#), [MariaDB 10.1.30](#)), Yes (<= [MariaDB 10.2.11](#), [MariaDB 10.1.29](#))

- **Data Type:**

enum

- **Valid Values:**

FCFS  
,  
VATS

- **Default Value:**

FCFS  
( [MariaDB 10.3.9](#), [MariaDB 10.2.17](#) ),  
VATS  
( [MariaDB 10.2.3](#) ),  
FCFS  
( [MariaDB 10.1](#) )

- **Introduced:** [MariaDB 10.2.3](#), [MariaDB 10.1.19](#)

- **Deprecated:** [MariaDB 10.5.7](#), [MariaDB 10.4.16](#), [MariaDB 10.3.26](#), [MariaDB 10.2.35](#)

- **Removed:** [MariaDB 10.6.0](#)

## innodb\_lock\_wait\_timeout

- **Description:** Time in seconds that an InnoDB transaction waits for an InnoDB record lock (or table lock) before giving up with the error  
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction  
. When this occurs, the statement (not transaction) is rolled back. The whole transaction can be rolled back if the  
**innodb\_rollback\_on\_timeout** option is used. Increase this for data warehousing applications or where other long-running operations are common, or decrease for OLTP and other highly interactive applications. This setting does not apply to deadlocks, which InnoDB detects immediately, rolling back a deadlocked transaction.

0  
(from MariaDB 10.3.0 ) means no wait. See [WAIT](#) and [NOWAIT](#) . Setting to  
100000000  
or more (from MariaDB 10.6.3 ,  
100000000  
is the maximum) means the timeout is infinite.

- **Commandline:**

--innodb-lock-wait-timeout=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

INT UNSIGNED  
(>= MariaDB 10.6.3 ),  
BIGINT UNSIGNED  
(<= MariaDB 10.6.2 )

- **Default Value:**

50

- **Range:**

- 0  
to  
100000000  
(>= MariaDB 10.6.3 )
- 0  
to  
1073741824  
(>= MariaDB 10.3 to <= MariaDB 10.6.2 )
- 1  
to  
1073741824  
(<= MariaDB 10.2 )

## innodb\_locking\_fake\_changes

- **Description:** From MariaDB 5.5 to MariaDB 10.1 , XtraDB-only option that if set to

OFF

, fake transactions (see [innodb\\_fake\\_changes](#) ) don't take row locks. This is an experimental feature to attempt to deal with drawbacks in fake changes blocking real locks. It is not safe for use in all environments. Added as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

--innodb-locking-fake-changes

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

- **Deprecated:** MariaDB 10.2.6

- **Removed:** MariaDB 10.3.0

## `innodb_locks_unsafe_for_binlog`

- **Description:** Set to

0

by default, in which case XtraDB/InnoDB uses [gap locking](#). If set to

1

, gap locking is disabled for searches and index scans. Deprecated in [MariaDB 10.0](#), and removed in [MariaDB 10.5](#), use [READ COMMITTED transaction isolation level](#) instead.

- **Commandline:**

`--innodb-locks-unsafe-for-binlog`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Deprecated:** [MariaDB 10.0](#)

- **Removed:** [MariaDB 10.5.0](#)

## `innodb_log_arch_dir`

- **Description:** The directory for [XtraDB redo log](#) archiving. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

`--innodb-log-arch-dir=name`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

- **Default Value:**

`./`

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

## `innodb_log_arch_expire_sec`

- **Description:** Time in seconds since the last change after which the archived [XtraDB redo log](#) should be deleted. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

`--innodb-log-arch-expire-sec=#`

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

## `innodb_log_archive`

- **Description:** Whether or not [XtraDB redo log](#) archiving is enabled. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

```
--innodb-log-archive={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

---

## innodb\_log\_block\_size

- **Description:** Size in bytes of the [XtraDB redo log](#) records. Generally

512

, the default, or

4096

, are the only two useful values. If the server is restarted and this value is changed, all old log files need to be removed. Should be set to  
4096

for SSD cards or if [innodb\\_flush\\_method](#) is set to

ALL\_O\_DIRECT

on ext4 filesystems. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

```
innodb-log-block-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

512

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

---

## innodb\_log\_buffer\_size

- **Description:** Size in bytes of the buffer for writing [InnoDB redo log](#) files to disk. Increasing this means larger transactions can run without needing to perform disk I/O before committing.

- **Commandline:**

```
--innodb-log-buffer-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

16777216

(16MB) >= [MariaDB 10.1.9](#),

8388608

(8MB) <= [MariaDB 10.1.8](#)

- **Range:**

262144

to

4294967295

(256KB to 4096MB)

## innodb\_log\_checksum\_algorithm

- **Description:** Experimental feature (as of [MariaDB 10.0.9](#) ), this variable specifies how to generate and verify [XtraDB redo log](#) checksums. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- - none
    - No checksum. A constant value is instead written to logs, and no checksum validation is performed.
  - innodb
    - The default, and the original InnoDB algorithm. This is inefficient, but compatible with all MySQL, MariaDB and Percona versions that don't support other checksum algorithms.
  - crc32
    - CRC32 © is used for log block checksums, which also permits recent CPUs to use hardware acceleration (on SSE4.2 x86 machines and Power8 or later) for the checksums.
  - strict\_\*
- Whether or not to accept checksums from other algorithms. If strict mode is used, checksums blocks will be considered corrupt if they don't match the specified algorithm. Normally they are considered corrupt only if no other algorithm matches.

- **Commandline:**

```
innodb-log-checksum-algorithm=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

- - deprecated
    - (>= [MariaDB 10.2.6](#) )
  - innodb
    - (<= [MariaDB 10.1](#) )

- **Valid Values:**

- - deprecated
  - ,
  - innodb
  - ,
  - none
  - ,
  - crc32
  - ,
  - strict\_none
  - ,
  - strict\_innodb
  - ,
  - strict\_crc32
  - (>= [MariaDB 10.2.6](#) )
- - innodb
  - ,
  - none
  - ,
  - crc32
  - ,
  - strict\_none
  - ,
  - strict\_innodb
  - ,
  - strict\_crc32
  - (<= [MariaDB 10.1](#) )

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

---

## innodb\_log\_checksums

- **Description:** If set to

1  
, the CRC32C for InnoDB or  
`innodb_log_checksum_algorithm`  
for XtraDB algorithm is used for [InnoDB redo log](#) pages. If disabled, the checksum field contents are ignored. From [MariaDB 10.5.0](#), the variable is deprecated, and checksums are always calculated, as previously, the InnoDB redo log used the slow innodb algorithm, but with hardware or SIMD assisted CRC-32C computation being available, there is no reason to allow checksums to be disabled on the redo log.

- **Commandline:**

```
innodb-log-checksums={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

- **Introduced:** [MariaDB 10.2.2](#)

- **Deprecated:** [MariaDB 10.5.0](#)

- **Removed:** [MariaDB 10.6.0](#)
- 

## innodb\_log\_compressed\_pages

- **Description:** Whether or not images of recompressed pages are stored in the [InnoDB redo log](#). Deprecated and ignored from [MariaDB 10.5.3](#).

- **Commandline:**

```
--innodb-log-compressed-pages={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

◦

ON

(>= [MariaDB 10.2.4](#) , >= [MariaDB 10.1.26](#) , <= [MariaDB 10.1.1](#) )

◦

OFF

( [MariaDB 10.2.0](#) - [MariaDB 10.2.3](#) , [MariaDB 10.1.2](#) - [MariaDB 10.1.25](#) )

- **Deprecated:** [MariaDB 10.5.3](#)

- **Removed:** [MariaDB 10.6.0](#)
- 

## innodb\_log\_file\_size

- **Description:** Size in bytes of each [InnoDB redo log](#) file in the log group. The combined size can be no more than 512GB. Larger values mean less disk I/O due to less flushing checkpoint activity, but also slower recovery from a crash. In [MariaDB 10.5](#), crash recovery has been improved and shouldn't run out of memory, so the default has been increased. It can safely be set higher to reduce checkpoint flushing, even larger than [innodb\\_buffer\\_pool\\_size](#).

- **Commandline:**

```
--innodb-log-file-size=
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

100663296

(96MB) (>= [MariaDB 10.5](#) ),

50331648

(48MB) (<= [MariaDB 10.4](#) )

- **Range:**

◦ >= [MariaDB 10.8.3](#) :

4194304

```
to  
512GB  
(4MB to 512GB)  
◦ <= MariaDB 10.8.2 :  
    1048576  
    to  
    512GB  
(1MB to 512GB)
```

---

## innodb\_log\_files\_in\_group

- **Description:** Number of physical files in the [InnoDB redo log](#). Deprecated and ignored from [MariaDB 10.5.2](#).
- **Commandline:**

```
--innodb-log-files-in-group=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    1  
(>= MariaDB 10.5 ),  
    2  

```

- **Range:**

```
    1  
    to  
    100  
(>= MariaDB 10.2.4 ),  
    2  

```

- **Deprecated:** [MariaDB 10.5.2](#)

- **Removed:** [MariaDB 10.6.0](#)
- 

## innodb\_log\_group\_home\_dir

- **Description:** Path to the [InnoDB redo log](#) files. If none is specified, [innodb\\_log\\_files\\_in\\_group](#) files named ib\_logfile0 and so on, with a size of [innodb\\_log\\_file\\_size](#) are created in the data directory.

- **Commandline:**

```
--innodb-log-group-home-dir=path
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
    directory name
```

---

## innodb\_log\_optimize\_ddl

- **Description:** Whether [InnoDB redo log](#) activity should be reduced when natively creating indexes or rebuilding tables. Reduced logging requires additional page flushing and interferes with [Mariabackup](#). Enabling this may slow down backup and cause delay due to page flushing. Deprecated and ignored from [MariaDB 10.5.1](#). Deprecated (but not ignored) from [MariaDB 10.4.16](#), [MariaDB 10.3.26](#) and [MariaDB 10.2.35](#).

- **Commandline:**

```
--innodb-log-optimize-ddl={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    boolean
```

- **Default Value:**

- OFF  
( $\geq$  MariaDB 10.5.1 , MariaDB 10.4.16 , MariaDB 10.3.26 , MariaDB 10.2.35 )
- ON  
( $\leq$  MariaDB 10.5.0 , MariaDB 10.4.15 , MariaDB 10.3.25 , MariaDB 10.2.34 )

- **Introduced:** MariaDB 10.2.17 , MariaDB 10.3.9

- **Deprecated:** MariaDB 10.5.1 , MariaDB 10.4.16 , MariaDB 10.3.26 , MariaDB 10.2.35

- **Removed:** MariaDB 10.6.0

---

### innodb\_log\_write\_ahead\_size

- **Description:** InnoDB redo log write ahead unit size to avoid read-on-write. Should match the OS cache block IO size.

- **Commandline:**

```
--innodb-log-write-ahead-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

8192

- **Range:**

512  
to innodb\_page\_size

- **Introduced:** MariaDB 10.2.2

---

### innodb\_lru\_flush\_size

- **Description:** Number of pages to flush on LRU eviction.

- **Commandline:**

```
--innodb-lru-flush-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

32

- **Range:**

1  
to  
18446744073709551615

- **Introduced:** MariaDB 10.5.7

---

### innodb\_lru\_scan\_depth

- **Description:** Specifies how far down the buffer pool least-recently used (LRU) list the cleaning thread should look for dirty pages to flush. This process is performed once a second. In an I/O intensive-workload, can be increased if there is spare I/O capacity, or decreased if in a write-intensive workload with little spare I/O capacity.

- See [InnoDB Page Flushing](#) for more information.

- **Commandline:**

```
--innodb-lru-scan-depth=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

◦

1536  
( $\geq$  MariaDB 10.5.7)

◦

1024  
( $\leq$  MariaDB 10.5.6)

- **Range - 32bit:**

100

to

2

32

-1

- **Range - 64bit:**

100

to

2

64

-1

## innodb\_max\_bitmap\_file\_size

- **Description:** Limit in bytes of the changed page bitmap files. For faster incremental backup with [Xtrabackup](#), XtraDB tracks pages with changes written to them according to the [XtraDB redo log](#) and writes the information to special changed page bitmap files. These files are rotated when the server restarts or when this limit is reached. XtraDB only. See also [innodb\\_track\\_changed\\_pages](#) and [innodb\\_max\\_changed\\_pages](#).

◦ Deprecated and ignored in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

innodb-max-bitmap-file-size=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

4096  
(4KB)

- **Range:**

4096  
(4KB) to  
18446744073709551615  
(16EB)

- **Deprecated:** MariaDB 10.2.6

## innodb\_max\_changed\_pages

- **Description:** Limit to the number of changed page bitmap files (stored in the [Information Schema INNODB\\_CHANGED\\_PAGES table](#)). Zero is unlimited. See [innodb\\_max\\_bitmap\\_file\\_size](#) and [innodb\\_track\\_changed\\_pages](#). Previously named

innodb\_changed\_pages\_limit  
. XtraDB only.

◦ Deprecated and ignored in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

innodb-max-changed-pages=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**  
numeric

- **Default Value:**  
1000000

- **Range:**  
0  
to  
18446744073709551615

- **Deprecated:** MariaDB 10.2.6
- 

### innodb\_max\_dirty\_pages\_pct

- **Description:** Maximum percentage of unwritten (dirty) pages in the buffer pool.
  - See [InnoDB Page Flushing](#) for more information.

- **Commandline:**  
--innodb-max-dirty-pages-pct=#

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric

- **Default Value:**
  - 90.000000  
(>= MariaDB 10.5.7 )
  - 75.000000  
(<= MariaDB 10.5.6 )

- **Range:**  
0  
to  
99.999
- 

### innodb\_max\_dirty\_pages\_pct\_lwm

- **Description:** Low water mark percentage of dirty pages that will enable preflushing to lower the dirty page ratio. The value 0 (default) means 'refer to [innodb\\_max\\_dirty\\_pages\\_pct](#)'.
  - See [InnoDB Page Flushing](#) for more information.

- **Commandline:**  
--innodb-max-dirty-pages-pct-lwm=#

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric

- **Default Value:**  
0  
(>= MariaDB 10.2.2 ),  
0.001000  
(<= MariaDB 10.2.1 )

- **Range:**  
0  
to  
99.999
- 

### innodb\_max\_purge\_lag

- **Description:** When purge operations are lagging on a busy server, setting innodb\_max\_purge\_lag can help. By default set to 0, no lag, the figure is used to calculate a time lag for each INSERT, UPDATE, and DELETE when the system is lagging. XtraDB/InnoDB keeps a list of transactions with delete-marked index records due to UPDATE and DELETE statements. The length of this list is purge\_lag, and the calculation, performed every ten seconds, is as follows: ((purge\_lag/innodb\_max\_purge\_lag)×10)–5 milliseconds.

- **Commandline:**

```
--innodb-max-purge-lag=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

4294967295

## innodb\_max\_purge\_lag\_delay

- **Description:** Maximum delay in milliseconds imposed by the [innodb\\_max\\_purge\\_lag](#) setting. If set to 0, the default, there is no maximum.

- **Commandline:**

```
--innodb-max-purge-lag-delay=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

## innodb\_max\_purge\_lag\_wait

- **Description:** Wait until History list length is below the specified limit.

- **Commandline:**

```
--innodb-max-purge-wait=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

4294967295

- **Range:**

0

to

4294967295

- **Introduced:** [MariaDB 10.5.7](#) , [MariaDB 10.4.16](#) , [MariaDB 10.3.26](#) , [MariaDB 10.2.35](#)

## `innodb_max_undo_log_size`

- **Description:** If an undo tablespace is larger than this, it will be marked for truncation if `innodb_undo_log_truncate` is set.

- **Commandline:**

```
--innodb-max-undo-log-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

◦

10485760  
( $\geq$  MariaDB 10.2.6 )

◦

1073741824  
( $\leq$  MariaDB 10.2.5 )

- **Range:**

10485760  
to  
18446744073709551615

- **Introduced:** MariaDB 10.2.2
- 

## `innodb_merge_sort_block_size`

- **Description:** Size in bytes of the block used for merge sorting in fast index creation. Replaced in MariaDB 10.0 /XtraDB 5.6 by `innodb_sort_buffer_size`.

- **Commandline:**

```
innodb-merge-sort-block-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1048576  
(1M)

- **Range:**

1048576  
(1M) to  
1073741824  
(1G)

- **Removed:** MariaDB 10.0 - replaced by `innodb_sort_buffer_size`
- 

## `innodb_mirrored_log_groups`

- **Description:** Unused. Restored as a deprecated and ignored option in MariaDB 10.2.6 (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Deprecated:** MariaDB 10.0

- **Removed:** MariaDB 10.2.2 - MariaDB 10.2.5
- 

## `innodb_mtflush_threads`

- **Description:** Sets the number of threads to use in Multi-Threaded Flush operations. For more information, see [Fusion-io Multi-threaded Flush](#).

◦ InnoDB's multi-thread flush feature was deprecated in MariaDB 10.2.9 and removed from MariaDB 10.3.2. In later versions of MariaDB, us

## `innodb_page_cleaners`

system variable instead.

◦ See [InnoDB Page Flushing: Page Flushing with Multi-threaded Flush Threads](#) for more information.

- **Commandline:**

```
--innodb-mtflush-threads=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    8
```

- **Range:**

```
    1
```

```
    to
```

```
    64
```

- **Introduced:** MariaDB 10.1.0

- **Deprecated:** MariaDB 10.2.9

- **Removed:** MariaDB 10.3.2

---

### innodb\_monitor\_disable

- **Description:** Disables the specified counters in the [INFORMATION\\_SCHEMA.INNODB\\_METRICS](#) table.

- **Commandline:**

```
--innodb-monitor-disable=string
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    string
```

---

### innodb\_monitor\_enable

- **Description:** Enables the specified counters in the [INFORMATION\\_SCHEMA.INNODB\\_METRICS](#) table.

- **Commandline:**

```
--innodb-monitor-enable=string
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    string
```

---

### innodb\_monitor\_reset

- **Description:** Resets the count value of the specified counters in the [INFORMATION\\_SCHEMA.INNODB\\_METRICS](#) table to zero.

- **Commandline:**

```
--innodb-monitor-reset=string
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    string
```

---

### innodb\_monitor\_reset\_all

- **Description:** Resets all values for the specified counters in the [INFORMATION\\_SCHEMA.INNODB\\_METRICS](#) table.

- **Commandline:**

```
--innodb-monitor-reset-all=string
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

---

## innodb\_numa\_interleave

- **Description:** Whether or not to use the NUMA interleave memory policy to allocate the [InnoDB buffer pool](#). Before [MariaDB 10.2.4](#), required that MariaDB be compiled on a NUMA-enabled Linux system.

- **Commandline:**

```
innodb-numa-interleave={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Removed:** [MariaDB 10.2.23](#), [MariaDB 10.3.14](#), [MariaDB 10.4.4](#)
- 

## innodb\_old\_blocks\_pct

- **Description:** Percentage of the [buffer pool](#) to use for the old block sublist.

- **Commandline:**

```
--innodb-old-blocks-pct=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

37

- **Range:**

5

to

95

---

## innodb\_old\_blocks\_time

- **Description:** Time in milliseconds an inserted block must stay in the old sublist after its first access before it can be moved to the new sublist. '0' means "no delay". Setting a non-zero value can help prevent full table scans clogging the [buffer pool](#). See also [innodb\\_old\\_blocks\\_pct](#).

- **Commandline:**

```
--innodb-old-blocks-time=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1000

- **Range:**

0  
to  
2

32

-1

---

## innodb\_online\_alter\_log\_max\_size

- **Description:** The maximum size for temporary log files during online DDL (data and index structure changes). The temporary log file is used for each table being altered, or index being created, to store data changes to the table while the process is underway. The table is extended by [innodb\\_sort\\_buffer\\_size](#) up to the limit set by this variable. If this limit is exceeded, the online DDL operation fails and all uncommitted changes are rolled back. A lower value reduces the time a table could lock at the end of the operation to apply all the log's changes, but also increases the chance of the online DDL changes failing.

- **Commandline:**

--innodb-online-alter-log-max-size=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

134217728

- **Range:**

65536  
to  
2

64

-1

---

## innodb\_open\_files

- **Description:** Maximum .ibd files MariaDB can have open at the same time. Only applies to systems with multiple XtraDB/InnoDB tablespaces, and is separate to the table cache and [open\\_files\\_limit](#). The default, if [innodb\\_file\\_per\\_table](#) is disabled, is 300 or the value of [table\\_open\\_cache](#), whichever is higher. It will also auto-size up to the default value if it is set to a value less than

10

- **Commandline:**

--innodb-open-files=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

autosized

- **Range:**

10  
to  
4294967295

## innodb\_optimize\_fulltext\_only

- **Description:** When set to

```
1  
(  
0  
is default), OPTIMIZE TABLE will only process InnoDB FULLTEXT index data. Only intended for use during fulltext index maintenance.
```

- **Commandline:**

```
--innodb-optimize-fulltext-only={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

## innodb\_page\_cleaners

- **Description:** Number of page cleaner threads. The default is

```
4  
, but the value will be set to the number of innodb\_buffer\_pool\_instances if this is lower. If set to  
1  
, only a single cleaner thread is used, as was the case until MariaDB 10.2.1. Cleaner threads flush dirty pages from the buffer pool,  
performing flush list and least-recently used (LRU) flushing. Deprecated and ignored from MariaDB 10.5.1, as the original reasons for splitting  
the buffer pool have mostly gone away.
```

◦ See [InnoDB Page Flushing: Page Flushing with Multiple InnoDB Page Cleaner Threads](#) for more information.

- **Commandline:**

```
--innodb-page-cleaners=#
```

- **Scope:** Global

- **Dynamic:** Yes (>= [MariaDB 10.3.3](#)), No (<= [MariaDB 10.3.2](#))

- **Data Type:**

numeric

- **Default Value:**

```
4  
(or set to innodb\_buffer\_pool\_instances if lower)
```

- **Range:**

```
1  
to  
64
```

- **Introduced:** [MariaDB 10.2.2](#)

- **Deprecated:** [MariaDB 10.5.1](#)

- **Removed:** [MariaDB 10.6.0](#)

---

## innodb\_page\_size

- **Description:** Specifies the page size in bytes for all InnoDB tablespaces. The default,

```
16k  
, is suitable for most uses.  
◦ A smaller InnoDB page size might work more effectively in a situation with many small writes (OLTP), or with SSD storage, which usually  
has smaller block sizes.  
◦ A larger InnoDB page size can provide a larger maximum row size.  
◦ InnoDB's page size can be as large as
```

```
64k  
for tables using the following row formats : DYNAMIC , COMPACT , and REDUNDANT.
```

◦ InnoDB's page size must still be

```
16k  
or less for tables using the COMPRESSED row format.
```

◦ This system variable's value cannot be changed after the

```
datadir  
has been initialized. InnoDB's page size is set when a MariaDB instance starts, and it remains constant afterwards.
```

- **Commandline:**

```
--innodb-page-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
enumeration
```

- **Default Value:**

```
16384
```

- **Valid Values:**

```
4k  
or  
4096  
,  
8k  
or  
8192  
,  
16k  
or  
16384  
,  
32k  
and  
64k
```

---

## innodb\_pass\_corrupt\_table

- **Removed:** XtraDB 5.5 - renamed [innodb\\_corrupt\\_table\\_action](#).

---

## innodb\_prefix\_index\_cluster\_optimization

- **Description:** Enable prefix optimization to sometimes avoid cluster index lookups.

- **Commandline:**

```
--innodb-prefix-index-cluster-optimization={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
boolean
```

- **Default Value:**

```
OFF
```

- **Introduced:** [MariaDB 10.1.2](#)

---

## innodb\_print\_all\_deadlocks

- **Description:** If set to

```
1
```

```
(
```

```
0
```

```
is default), all XtraDB/InnoDB transaction deadlock information is written to the error log.
```

- **Commandline:**

```
--innodb-print-all-deadlocks={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

### innodb\_purge\_batch\_size

- **Description:** Units of [InnoDB redo log](#) records that will trigger a purge operation. Together with [innodb\\_purge\\_threads](#) has a small effect on tuning.
- **Commandline:**

```
--innodb-purge-batch-size=#
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**  
20

- **Range:**  
1  
to  
5000
- 

### innodb\_purge\_rseg\_truncate\_frequency

- **Description:** Frequency with which undo records are purged. Set by default to every 128 times, reducing this increases the frequency at which rollback segments are freed. See also [innodb\\_undo\\_log\\_truncate](#).
- **Commandline:**

```
-- innodb-purge-rseg-truncate-frequency=#
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric

- **Default Value:**  
128

- **Range:**  
1  
to  
128

- **Introduced:** [MariaDB 10.2.2](#)
- 

### innodb\_purge\_threads

- **Description:** Number of background threads dedicated to InnoDB purge operations. The range is

```
1
```

```
to
```

```
32
```

. At least one background thread is always used from [MariaDB 10.0](#). The default has been increased from

```
1
```

```
to
```

```
4
```

in [MariaDB 10.2.2](#). Setting to a value greater than 1 creates many separate purge threads. This can improve efficiency in some cases, such as when performing DML operations on many tables. In [MariaDB 5.5](#), the options are

0  
and  
1  
. If set to  
0  
, the default, purging is done with the primary thread. If set to  
1  
, purging is done on a separate thread, which could reduce contention. See also [innodb\\_purge\\_batch\\_size](#).

- **Commandline:**

```
--innodb-purge-threads=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

- - 4  
(>= [MariaDB 10.2.2](#))
  - 1  
(>= [MariaDB 10.0](#) to <= [MariaDB 10.2.1](#))

- **Range:**

1  
to  
32

---

## innodb\_random\_read\_ahead

- **Description:** Originally, random read-ahead was always set as an optimization technique, but was removed in [MariaDB 5.5](#).

innodb\_random\_read\_ahead  
permits it to be re-instated if set to  
1  
(  
0  
) is default.

- **Commandline:**

```
--innodb-random-read-ahead={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

## innodb\_read\_ahead

- **Description:** If set to

linear  
, the default, XtraDB/InnoDB will automatically fetch remaining pages if there are enough within the same extent that can be accessed sequentially. If set to

none  
, read-ahead is disabled.

random  
has been removed and is now ignored, while

both

sets to both

linear

and

random

. Also see [innodb\\_read\\_ahead\\_threshold](#) for more control on read-aheads. Removed in [MariaDB 10.0](#) /XtraDB 5.6 and replaced by

MySQL 5.6's [innodb\\_random\\_read\\_ahead](#).

- **Commandline:**

```
innodb-read-ahead=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
enumeration
```

- **Default Value:**

```
linear
```

- **Valid Values:**

```
none
```

```
,
```

```
random
```

```
,
```

```
linear
```

```
,
```

```
both
```

- **Removed:** [MariaDB 10.0](#) /XtraDB 5.6 - replaced by MySQL 5.6's [innodb\\_random\\_read\\_ahead](#)

---

## innodb\_read\_ahead\_threshold

• **Description:** Minimum number of pages XtraDB/InnoDB must read from an extent of 64 before initiating an asynchronous read for the following extent.

- **Commandline:**

```
--innodb-read-ahead-threshold=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
numeric
```

- **Default Value:**

```
56
```

- **Range:**

```
0
```

```
to
```

```
64
```

## innodb\_read\_io\_threads

• **Description:** Number of I/O threads for XtraDB/InnoDB reads. You may on rare occasions need to reduce this default on Linux systems running multiple MariaDB servers to avoid exceeding system limits.

- **Commandline:**

```
--innodb-read-io-threads=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
numeric
```

- **Default Value:**

```
4
```

- **Range:**

```
1
```

```
to
```

```
64
```

## innodb\_read\_only

- **Description:** If set to

1

(

0

is default), the server will be read-only. For use in distributed applications, data warehouses or read-only media.

- **Commandline:**

--innodb-read-only={0|1}

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

## innodb\_read\_only\_compressed

- **Description:** If set (the default before [MariaDB 10.6.6](#)), **ROW\_FORMAT=COMPRESSED** tables will be read-only. This was intended to be the first step towards removing write support and deprecating the feature, but this plan has been abandoned.

- **Commandline:**

--innodb-read-only-compressed  
,  
--skip-innodb-read-only-compressed

- **Scope:**

- **Dynamic:**

- **Data Type:**

boolean

- **Default Value:**

OFF  
(>= [MariaDB 10.6.6](#) ),  
ON  
(<= [MariaDB 10.6.5](#) )

- **Introduced:** [MariaDB 10.6.0](#)
- ## innodb\_recovery\_stats
- **Description:** If set to
- 1
- (
- 0
- is default) and recovery is necessary on startup, the server will write detailed recovery statistics to the error log at the end of the recovery process. This Percona XtraDB variable has not been ported to XtraDB 5.6.
- **Commandline:** No
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**
- boolean
- **Default Value:**
- OFF
- **Removed:** [MariaDB 10.0](#)

## innodb\_recovery\_update\_relay\_log

---

1800/2552

- **Description:** If set to

```
1
(
0
```

is default), the relay log info file will be overwritten on crash recovery if the information differs from the InnoDB record. Should not be used if multiple storage engine types are being replicated. Previously named

```
innodb_overwrite_relay_log_info
. Removed in MariaDB 10.0 /XtraDB 5.6 and replaced by MySQL 5.6's
relay-log-recovery
```

- **Commandline:**

```
innodb-recovery-update-relay-log={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
boolean
```

- **Default Value:**

```
OFF
```

- **Removed:** MariaDB 10.0 - replaced by MySQL 5.6's

```
relay-log-recovery
```

---

## innodb\_replication\_delay

- **Description:** Time in milliseconds for the replica server to delay the replication thread if [innodb\\_thread\\_concurrency](#) is reached. Deprecated and ignored from [MariaDB 10.5.5](#).

- **Commandline:**

```
--innodb-replication-delay=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
numeric
```

- **Default Value:**

```
0
```

- **Range:**

```
0
to
4294967295
```

- **Deprecated:** MariaDB 10.5.5

- **Removed:** MariaDB 10.6.0
- 

## innodb\_rollback\_on\_timeout

- **Description:** InnoDB usually rolls back the last statement of a transaction that's been timed out (see [innodb\\_lock\\_wait\\_timeout](#)). If [innodb\\_rollback\\_on\\_timeout](#) is set to 1 (0 is default), InnoDB will roll back the entire transaction. Before [MariaDB 5.5](#), rolling back the entire transaction was the default behavior.

- **Commandline:**

```
--innodb-rollback-on-timeout
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
boolean
```

- **Default Value:**

```
0
```

---

## innodb\_rollback\_segments

- **Description:** Specifies the number of rollback segments that XtraDB/InnoDB will use within a transaction (see [undo log](#) ). Deprecated and replaced by [innodb\\_undo\\_logs](#) in [MariaDB 10.0](#) .
  - **Commandline:**

```
--innodb-rollback-segments=#
```
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
numeric
  - **Default Value:**  
128
  - **Range:**  
1  
to  
128
  - **Deprecated:** [MariaDB 10.0](#)
  - **Removed:** [MariaDB 10.5.0](#)
- 

## innodb\_safe\_truncate

- **Description:** Use a backup-safe [TRUNCATE TABLE](#) implementation and crash-safe rename operations inside InnoDB. This is not compatible with hot backup tools other than [Mariabackup](#) . Users who need to use such tools may set this to  
OFF
  - **Commandline:**

```
--innodb-safe-truncate={0|1}
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
  - **Default Value:**  
ON
  - **Introduced:** [MariaDB 10.2.19](#)
  - **Removed:** [MariaDB 10.3.0](#)
- 

## innodb\_scrub\_log

- **Description:** Enable [InnoDB redo log](#) scrubbing. See [Data Scrubbing](#) . Deprecated and ignored from [MariaDB 10.5.2](#) , as never really worked ([MDEV-13019](#) and [MDEV-18370](#) ). If old log contents should be kept secret, then enabling [innodb\\_encrypt\\_log](#) or setting a smaller [innodb\\_log\\_file\\_size](#) could help.
  - **Commandline:**

```
--innodb-scrub-log
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
  - **Introduced:** [MariaDB 10.1.3](#)
  - **Deprecated:** [MariaDB 10.5.2](#)
  - **Removed:** [MariaDB 10.6.0](#)
-

## `innodb_scrub_log_interval`

- **Description:** Used with [Data Scrubbing](#) in 10.1.3 only - replaced in 10.1.4 by [innodb\\_scrub\\_log\\_speed](#). InnoDB redo log scrubbing interval in milliseconds.
  - **Commandline:**  
    `--innodb-scrub-log-interval=#`
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    numeric
  - **Default Value:**  
    56
  - **Range:**  
    0  
    to  
    50000
  - **Introduced:** [MariaDB 10.1.3](#)
  - **Removed:** [MariaDB 10.1.4](#)
- 

## `innodb_scrub_log_speed`

- **Description:** InnoDB redo log scrubbing speed in bytes/sec. See [Data Scrubbing](#). Deprecated and ignored from [MariaDB 10.5.2](#).
  - **Commandline:**  
    `--innodb-scrub-log-speed=#`
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    numeric
  - **Default Value:**  
    256
  - **Range:**  
    1  
    to  
    50000
  - **Introduced:** [MariaDB 10.1.4](#)
  - **Deprecated:** [MariaDB 10.5.2](#)
  - **Removed:** [MariaDB 10.6.0](#)
- 

## `innodb_sched_priority_cleaner`

- **Description:** Set a thread scheduling priority for cleaner and least-recently used (LRU) manager threads. The range from 0 to 39 corresponds in reverse order to Linux nice values of -20 to 19. So 0 is the lowest priority (Linux nice value 19) and 39 is the highest priority (Linux nice value -20)

). XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

```
innodb-sched-priority-cleaner=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

19

- **Range:**

0

to

39

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

---

## innodb\_show\_locks\_held

- **Description:** Specifies the number of locks held for each InnoDB transaction to be displayed in [SHOW ENGINE INNODB STATUS](#) output. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

```
innodb-show-locks-held=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

10

- **Range:**

0

to

1000

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

---

## innodb\_show\_verbose\_locks

- **Description:** If set to

1

, and [innodb\\_status\\_output\\_locks](#) is also ON, the traditional InnoDB behavior is followed and locked records will be shown in [SHOW ENGINE INNODB STATUS](#) output. If set to

0

, the default, only high-level information about the lock is shown. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

```
innodb-show-verbose-locks=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0  
to  
1

- **Deprecated:** MariaDB 10.2.6
  - **Removed:** MariaDB 10.3.0
- 

### innodb\_simulate\_comp\_failures

- **Description:** Simulate compression failures. Used for testing robustness against random compression failures. XtraDB only.

- **Commandline:** None

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

99

### innodb\_sort\_buffer\_size

- **Description:** Size of the sort buffers used for sorting data when an InnoDB index is created, as well as the amount by which the temporary log file is extended during online DDL operations to record concurrent writes. The larger the setting, the fewer merge phases are required between buffers while sorting. When a [CREATE TABLE](#) or [ALTER TABLE](#) creates a new index, three buffers of this size are allocated, as well as pointers for the rows in the buffer.

- **Commandline:**

--innodb-sort-buffer-size=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

1048576  
(1M)

- **Range:**

65536

to

67108864

### innodb\_spin\_wait\_delay

- **Description:** Maximum delay (not strictly corresponding to a time unit) between spin lock polls. Default changed from

6  
to  
4

in [MariaDB 10.3.5](#), as this was verified to give the best throughput by OLTP update index and read-write benchmarks on Intel Broadwell (2/20/40) and ARM (1/46/46).

- **Commandline:**

--innodb-spin-wait-delay=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

4  
(>= MariaDB 10.3.5 ),  
6  
(<= MariaDB 10.3.4 )

- **Range:**

0  
to  
4294967295

---

## innodb\_stats\_auto\_recalc

- **Description:** If set to

1

(the default), persistent statistics are automatically recalculated when the table changes significantly (more than 10% of the rows). Affects tables created or altered with STATS\_PERSISTENT=1 (see [CREATE TABLE](#)), or when [innodb\\_stats\\_persistent](#) is enabled.

[innodb\\_stats\\_persistent\\_sample\\_pages](#) determines how much data to sample when recalculating. See [InnoDB Persistent Statistics](#).

- **Commandline:**

--innodb-stats-auto-recalc={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

---

## innodb\_stats\_auto\_update

- **Description:** If set to

0

(

1

is default), index statistics will not be automatically calculated except when an [ANALYZE TABLE](#) is run, or the table is first opened.

Replaced by [innodb\\_stats\\_auto\\_recalc](#) in MariaDB 10.0 /XtraDB 5.6.

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

1

- **Removed:** MariaDB 10.0 - replaced by [innodb\\_stats\\_auto\\_recalc](#).

---

## innodb\_stats\_include\_delete\_marked

- **Description:** Include delete marked records when calculating persistent statistics.

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** MariaDB 10.2.6

## innodb\_stats\_method

- **Description:** Determines how NULLs are treated for InnoDB index statistics purposes.

- - **nulls\_equal**  
◦ The default, all NULL index values are treated as a single group. This is usually fine, but if you have large numbers of NULLs the average group size is slanted higher, and the optimizer may miss using the index for ref accesses when it would be useful.
  - **nulls\_unequal**  
◦ The opposite approach to **nulls\_equal** is taken, with each NULL forming its own group of one. Conversely, the average group size is slanted lower, and the optimizer may use the index for ref accesses when not suitable.
  - **nulls\_ignored**  
◦ Ignore NULLs altogether from index group calculations.

◦ See also [Index Statistics](#) , [aria\\_stats\\_method](#) and [myisam\\_stats\\_method](#) .

- **Commandline:**

◦ --innodb-stats-method=name

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

◦ enumeration

- **Default Value:**

◦ nulls\_equal

- **Valid Values:**

◦ nulls\_equal  
◦ ,  
◦ nulls\_unequal  
◦ ,  
◦ nulls\_ignored

## innodb\_stats\_modified\_counter

- **Description:** The number of rows modified before we calculate new statistics. If set to

◦ 0  
◦ , the default, current limits are used.

- **Commandline:**

◦ --innodb-stats-modified-counter=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

◦ numeric

- **Default Value:**

◦ 0

- **Range:**

◦ 0  
◦ to  
◦ 18446744073709551615

## innodb\_stats\_on\_metadata

- **Description:** If set to

◦ 1

, the default, XtraDB/InnoDB updates statistics when accessing the INFORMATION\_SCHEMA.TABLES or INFORMATION\_SCHEMA.STATISTICS tables, and when running metadata statements such as [SHOW INDEX](#) or [SHOW TABLE STATUS](#). If set to

0

, statistics are not updated at those times, which can reduce the access time for large schemas, as well as make execution plans more stable.

- **Commandline:**

```
--innodb-stats-on-metadata
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

## innodb\_stats\_persistent

- **Description:** [ANALYZE TABLE](#) produces index statistics, and this setting determines whether they will be stored on disk, or be required to be recalculated more frequently, such as when the server restarts. This information is stored for each table, and can be set with the STATS\_PERSISTENT clause when creating or altering tables (see [CREATE TABLE](#)). See [InnoDB Persistent Statistics](#).

- **Commandline:**

```
--innodb-stats-persistent={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

---

## innodb\_stats\_persistent\_sample\_pages

- **Description:** Number of index pages sampled when estimating cardinality and statistics for indexed columns. Increasing this value will increase index statistics accuracy, but use more I/O resources when running [ANALYZE TABLE](#). See [InnoDB Persistent Statistics](#).

- **Commandline:**

```
--innodb-stats-persistent-sample-pages=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

20

---

## innodb\_stats\_sample\_pages

- **Description:** Gives control over the index distribution statistics by determining the number of index pages to sample. Higher values produce more disk I/O, but, especially for large tables, produce more accurate statistics and therefore make more effective use of the query optimizer. Lower values than the default are not recommended, as the statistics can be quite inaccurate.

- If [innodb\\_stats\\_traditional](#) is enabled, then the exact number of pages configured by this system variable will be sampled for statistics.
- If [innodb\\_stats\\_traditional](#) is disabled, then the number of pages to sample for statistics is calculated using a logarithmic algorithm, so the exact number can change depending on the size of the table. This means that more samples may be used for larger tables.
- If [persistent statistics](#) are enabled, then the [innodb\\_stats\\_persistent\\_sample\\_pages](#) system variable applies instead. [persistent statistics](#) are enabled with the [innodb\\_stats\\_persistent](#) system variable.
- This system variable has been [deprecated](#). The [innodb\\_stats\\_transient\\_sample\\_pages](#) system variable should be used instead.

- **Commandline:**

```
--innodb-stats-sample-pages#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

8

- **Range:**

1

to

2

64

-1

- **Deprecated:** MariaDB 10.0

- **Removed:** MariaDB 10.5.0

## innodb\_stats\_traditional

- **Description:** This system variable affects how the number of pages to sample for transient statistics is determined, in particular how [innodb\\_stats\\_transient\\_sample\\_pages](#) #is used.

- If [innodb\\_stats\\_traditional](#) is enabled, then the exact number of pages configured by the system variable will be sampled for statistics.
- If [innodb\\_stats\\_traditional](#) is disabled, then the number of pages to sample for statistics is calculated using a logarithmic algorithm, so the exact number can change depending on the size of the table. This means that more samples may be used for larger tables.
- This system variable does not affect the calculation of [persistent statistics](#).

- **Commandline:**

```
--innodb-stats-traditional={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

## innodb\_stats\_transient\_sample\_pages

- **Description:** Gives control over the index distribution statistics by determining the number of index pages to sample. Higher values produce more disk I/O, but, especially for large tables, produce more accurate statistics and therefore make more effective use of the query optimizer. Lower values than the default are not recommended, as the statistics can be quite inaccurate.

- If

[innodb\\_stats\\_traditional](#)

is enabled, then the exact number of pages configured by this system variable will be sampled for statistics.

- If

[innodb\\_stats\\_traditional](#)

is disabled, then the number of pages to sample for statistics is calculated using a logarithmic algorithm, so the exact number can change depending on the size of the table. This means that more samples may be used for larger tables.

- If [persistent statistics](#) are enabled, then the

[innodb\\_stats\\_persistent\\_sample\\_pages](#)

system variable applies instead. [persistent statistics](#) are enabled with the

[innodb\\_stats\\_persistent](#)

system variable.

- **Commandline:**

```
--innodb-stats-transient-sample-pages=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

8

- **Range:**

1

to

2

64

-1

## innodb\_stats\_update\_need\_lock

- **Description:** Setting to

0

(

1

is default) may help reduce contention of the

&dict\_operation\_lock

, but also disables the *Data\_free* option in [SHOW TABLE STATUS](#). This Percona XtraDB variable has not been ported to XtraDB 5.6.

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

1

- **Removed:** [MariaDB 10.0](#) /XtraDB 5.6

## innodb\_status\_output

- **Description:** Enable [InnoDB monitor](#) output to the [error log](#).

- **Commandline:**

```
--innodb-status-output={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

## innodb\_status\_output\_locks

- **Description:** Enable [InnoDB lock monitor](#) output to the [error log](#) and [SHOW ENGINE INNODB STATUS](#). Also requires [innodb\\_status\\_output=ON](#) to enable output to the error log.

- **Commandline:**

```
--innodb-status-output-locks={0|1}
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**

OFF

---

## innodb\_strict\_mode

- **Description:** If set to

1

(

0

is the default before [MariaDB 10.2.2](#) ), XtraDB/InnoDB will return errors instead of warnings in certain cases, similar to strict SQL mode.

- **Commandline:**

```
--innodb-strict-mode={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

◦

ON

(>= [MariaDB 10.2.2](#) )

◦

OFF

(<= [MariaDB 10.2.1](#) )

---

## innodb\_support\_xa

- **Description:** If set to

1

, the default, [XA transactions](#) are supported. XA support ensures data is written to the [binary log](#) in the same order to the actual database, which is critical for [replication](#) and disaster recovery, but comes at a small performance cost. If your database is set up to only permit one thread to change data (for example, on a replication replica with only the replication thread writing), it is safe to turn this option off. Removed in [MariaDB 10.3](#) , XA transactions are always supported.

- **Commandline:**

```
--innodb-support-xa
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

- **Deprecated:** [MariaDB 10.2](#)

- **Removed:** [MariaDB 10.3.0](#)
- 

## innodb\_sync\_array\_size

- **Description:** By default

1

, can be increased to split internal thread co-ordinating, giving higher concurrency when there are many waiting threads.

- **Commandline:**

```
--innodb-sync-array-size=#
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    numeric

- **Default Value:**

```
1
```

- **Range:**

```
1  
to  
1024
```

- **Removed:** MariaDB 10.6.0
- 

## innodb\_sync\_spin\_loops

- **Description:** The number of times a thread waits for an XtraDB/InnoDB mutex to be freed before the thread is suspended.
- **Commandline:**

```
--innodb-sync-spin-loops=#
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
    numeric

- **Default Value:**

```
30
```

- **Range:**

```
0  
to  
4294967295
```

---

## innodb\_table\_locks

- **Description:** If `autocommit` is set to to

```
0  
(  
1  
is default), setting innodb_table_locks to  
1  
, the default, will cause XtraDB/InnoDB to lock a table internally upon a LOCK TABLE .
```

- **Commandline:**

```
--innodb-table-locks
```

- **Scope:** Global, Session
- **Dynamic:** Yes
- **Data Type:**  
    boolean

- **Default Value:**

```
ON
```

---

## innodb\_thread\_concurrency

- **Description:** Once this number of threads is reached (excluding threads waiting for locks), XtraDB/InnoDB will place new threads in a wait state in a first-in, first-out queue for execution, in order to limit the number of threads running concurrently. A setting of

0

, the default, permits as many threads as necessary. A suggested setting is twice the number of CPU's plus the number of disks. Deprecated and ignored from [MariaDB 10.5.5](#).

- **Commandline:**

```
--innodb-thread-concurrency=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

1000

- **Deprecated:** [MariaDB 10.5.5](#)

- **Removed:** [MariaDB 10.6.0](#)

## innodb\_thread\_concurrency\_timer\_based

- **Description:** If set to

1

, thread concurrency will be handled in a lock-free timer-based manner rather than the default mutex-based method. Depends on atomic op builtins being available. This Percona XtraDB variable has not been ported to XtraDB 5.6.

- **Commandline:**

```
innodb-thread-concurrency-timer-based={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Removed:** [MariaDB 10.0](#) /XtraDB 5.6

## innodb\_thread\_sleep\_delay

- **Description:** Time in microseconds that InnoDB threads sleep before joining the queue. Setting to

0

disables sleep. Deprecated and ignored from [MariaDB 10.5.5](#).

- **Commandline:**

```
--innodb-thread-sleep-delay=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

◦

0

(>= [MariaDB 10.5.5](#) .)

◦

10000

(<= [MariaDB 10.5.4](#) )

- **Range:**

0

to

1000000

- **Deprecated:** MariaDB 10.5.5
  - **Removed:** MariaDB 10.6.0
- 

## innodb\_temp\_data\_file\_path

- **Description:**
  - **Commandline:**  
--innodb-temp-data-file-path=path
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
string
  - **Default Value:**  
ibtmp1:12M:autoextend
  - **Introduced:** MariaDB 10.2.2
- 

## innodb\_tmpdir

- **Description:** Allows an alternate location to be set for temporary non-tablespace files. If not set (the default), files will be created in the usual tmpdir location.
  - **Commandline:**  
--innodb-tmpdir=path
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
string
  - **Default Value:** Empty
  - **Introduced:** MariaDB 10.1.14 , MariaDB 10.2.1
- 

## innodb\_track\_changed\_pages

- **Description:** For faster incremental backup with [Xtrabackup](#) , XtraDB tracks pages with changes written to them according to the [XtraDB redo log](#) and writes the information to special changed page bitmap files. This read-only variable is used for controlling this feature. See also [innodb\\_max\\_changed\\_pages](#) and [innodb\\_max\\_bitmap\\_file\\_size](#) . XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
  - **Commandline:**  
innodb-track-changed-pages={0|1}
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
  - **Deprecated:** MariaDB 10.2.6
- 

## innodb\_track\_redo\_log\_now

- **Description:** Available on debug builds only. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.
- **Commandline:**

innodb-track-redo-log-now={0|1}

- **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
  - **Deprecated:** MariaDB 10.2.6
- 

## innodb\_undo\_directory

- **Description:** Path to the directory (relative or absolute) that InnoDB uses to create separate tablespaces for the [undo logs](#).  
(the default value before 10.2.2) leaves the undo logs in the same directory as the other log files. From MariaDB 10.2.2, the default value is NULL, and if no path is specified, undo tablespaces will be created in the directory defined by [datadir](#). Use together with [innodb\\_undo\\_logs](#) and [innodb\\_undo\\_tablespaces](#). Undo logs are most usefully placed on a separate storage device.
  - **Commandline:**  
--innodb-undo-directory=name
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
string
  - **Default Value:** NULL (>= MariaDB 10.2.2),  
(<= MariaDB 10.2.1)
- 

## innodb\_undo\_log\_truncate

- **Description:** When enabled, undo tablespaces that are larger than [innodb\\_max\\_undo\\_log\\_size](#) are marked for truncation. See also [innodb\\_purge\\_rseg\\_truncate\\_frequency](#).
  - **Commandline:**  
--innodb-undo-log-truncate[={0|1}]
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
  - **Introduced:** MariaDB 10.2.2
- 

## innodb\_undo\_logs

- **Description:** Specifies the number of rollback segments that XtraDB/InnoDB will use within a transaction (or the number of active [undo logs](#)). By default set to the maximum,  
128  
, it can be reduced to avoid allocating unneeded rollback segments. See the [Innodb\\_available\\_undo\\_logs](#) status variable for the number of undo logs available. See also [innodb\\_undo\\_directory](#) and [innodb\\_undo\\_tablespaces](#). Replaced [innodb\\_rollback\\_segments](#) in MariaDB 10.0. The [Information Schema XTRADB\\_RSEG Table](#) contains information about the XtraDB rollback segments. Deprecated and ignored in MariaDB 10.5.0, as it always makes sense to use the maximum number of rollback segments.
- **Commandline:**  
--innodb-undo-logs=#
- **Scope:** Global
- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

128

- **Range:**

0  
to  
128

- **Deprecated:** MariaDB 10.5.0

- **Removed:** MariaDB 10.6.0

---

## innodb\_undo\_tablespaces

- **Description:** Number of tablespaces files used for dividing up the [undo logs](#). By default, undo logs are all part of the system tablespace, which contains one undo tablespace more than the

innodb\_undo\_tablespaces

setting. When the undo logs can grow large, splitting them over multiple tablespaces will reduce the size of any single tablespace. Must be set before InnoDB is initialized, or else MariaDB will fail to start, with an error saying that

InnoDB did not find the expected number of undo tablespaces

. The files are created in the directory specified by [innodb\\_undo\\_directory](#), and are named

undo

, N being an integer. The default size of an undo tablespace is 10MB. [innodb\\_undo\\_logs](#) must have a non-zero setting for

innodb\_undo\_tablespaces

to take effect.

- **Commandline:**

--innodb-undo-tablespaces=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0  
to  
95  
(>= MariaDB 10.2.2 ),  
0  
to  
126  
(<= MariaDB 10.2.1 )

## innodb\_use\_atomic\_writes

- **Description:** Implement atomic writes on supported SSD devices. See [atomic write support](#) for other variables affected when this is set.

- **Commandline:**

innodb-use-atomic-writes={0|1}

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON  
(>= MariaDB 10.2.4 ),  
OFF  
(<= MariaDB 10.2.3 )

## `innodb_use_fallocate`

- **Description:** Preallocate files fast, using operating system functionality. On POSIX systems, `posix_fallocate` system call is used.

- Automatically set to

- 1  
when `innodb_use_atomic_writes` is set - see [FusionIO DirectFS atomic write support](#).

- See [InnoDB Page Compression: Saving Storage Space with Sparse Files](#) for more information.

- **Commandline:**

- `--innodb-use-fallocate={0|1}`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

- `boolean`

- **Default Value:**

- `OFF`

- **Deprecated:** [MariaDB 10.2.5](#) (treated as if

- `ON`

- `)`

- **Removed:** [MariaDB 10.3.0](#)
- 

## `innodb_use_global_flush_log_at_trx_commit`

- **Description:** Determines whether a user can set the variable `innodb_flush_log_at_trx_commit`. If set to

- `1`  
, a user cannot reset the value with a `SET` command, while if set to

- `1`  
, a user can reset the value of  
`innodb_flush_log_at_trx_commit`

- `XtraDB` only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

- `--innodb-use-global-flush-log-at-trx_commit={0|1}`

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

- `boolean`

- **Default Value:**

- `ON`

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)
- 

## `innodb_use_mtflush`

- **Description:** Whether to enable Multi-Threaded Flush operations. For more information, see [Fusion](#).

- InnoDB's multi-thread flush feature was deprecated in [MariaDB 10.2.9](#) and removed from [MariaDB 10.3.2](#). In later versions of MariaDB, us

### `innodb_page_cleaners`

system variable instead.

- See [InnoDB Page Flushing: Page Flushing with Multi-threaded Flush Threads](#) for more information.

- **Commandline:**

- `--innodb-use-mtflush={0|1}`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

- `boolean`

- **Default Value:**

OFF

- **Introduced:** MariaDB 10.1.0
  - **Deprecated:** MariaDB 10.2.9
  - **Removed:** MariaDB 10.3.2
- 

### innodb\_use\_native\_aio

- **Description:** For Linux systems only, specified whether to use Linux's asynchronous I/O subsystem. Set to

ON

by default, it may be changed to

0

at startup if InnoDB detects a problem, or from [MariaDB 10.6.5 / MariaDB 10.7.1](#), if a 5.11 - 5.15 Linux kernel is detected, to avoid an interesting bug/incompatibility ([MDEV-26674](#)). MariaDB-10.6/MariaDB-10.7.2 and later also consider 5.15.3+ as a fixed kernel and default to

ON

. To really benefit from the setting, the files should be opened in O\_DIRECT mode (`innodb_flush_method=O_DIRECT`, default from [MariaDB 10.6](#)), to bypass the file system cache. In this way, the reads and writes can be submitted with DMA, using the InnoDB buffer pool directly, and no processor cycles need to be used for copying data.

- **Commandline:**

`--innodb-use-native-aio={0|1}`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

---

### innodb\_use\_purge\_thread

- **Description:** Usually with InnoDB, data changed by a transaction is written to an undo space to permit read consistency, and freed when the transaction is complete. Many, or large, transactions, can cause the main tablespace to grow dramatically, reducing performance. This option, introduced in XtraDB 5.1 and removed for 5.5, allows multiple threads to perform the purging, resulting in slower, but much more stable performance.

- **Commandline:**

`--innodb-use-purge-thread=#`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0

to

32

- **Removed:** XtraDB 5.5
- 

### innodb\_use\_stacktrace

- **Description:** If set to

ON

(

OFF

is default), a signal handler for SIGUSR2 is installed when the InnoDB server starts. When a long semaphore wait is detected at sync/sync0array.c, a SIGUSR2 signal is sent to the waiting thread and thread that has acquired the RW-latch. For both threads a full stacktrace is produced as well as if possible. XtraDB only. Added as a deprecated and ignored option in [MariaDB 10.2.6](#) (which uses InnoDB as default instead of XtraDB) to allow for easier upgrades.

- **Commandline:**

```
--innodb-use-stacktrace={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Deprecated:** [MariaDB 10.2.6](#)

- **Removed:** [MariaDB 10.3.0](#)

---

## innodb\_use\_sys\_malloc

- **Description:** If set the

1

, the default, XtraDB/InnoDB will use the operating system's memory allocator. If set to

0

it will use its own. Deprecated in [MariaDB 10.0](#) and removed in [MariaDB 10.2](#) along with InnoDB's internal memory allocator.

- **Commandline:**

```
--innodb-use-sys-malloc={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

- **Deprecated:** [MariaDB 10.0](#)

- **Removed :** [MariaDB 10.2.2](#)

---

## innodb\_use\_sys\_stats\_table

- **Description:** If set to

1

(

0

is default), XtraDB will use the SYS\_STATS system table for extra table index statistics. When a table is opened for the first time, statistics will then be loaded from SYS\_STATS instead of sampling the index pages. Statistics are designed to be maintained only by running an [ANALYZE TABLE](#) . Replaced by MySQL 5.6's Persistent Optimizer Statistics.

- **Commandline:**

```
innodb-use-sys-stats-table={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

0

- **Removed:** [MariaDB 10.0](#) /XtraDB 5.6

---

## innodb\_use\_trim

- **Description:** Use trim to free up space of compressed blocks.
    - See [InnoDB Page Compression: Saving Storage Space with Sparse Files](#) for more information.
  - **Commandline:**

```
--innodb-use-trim={0|1}
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**

```
boolean
```
  - **Default Value:**

```
ON  
(>= MariaDB 10.2.4 ),  
OFF  
(<= MariaDB 10.2.3 )
```
  - **Introduced:** MariaDB 10.1.0
  - **Deprecated:** MariaDB 10.2.4
  - **Removed:** MariaDB 10.3.0
- 

## innodb\_version

- **Description:** InnoDB version number. From [MariaDB 10.3.7](#), as the InnoDB implementation in MariaDB has diverged from MySQL, the MariaDB version is instead reported. For example, the InnoDB version reported in [MariaDB 10.1](#) (which is based on MySQL 5.6) included encryption and variable-size page compression before MySQL 5.7 introduced them. [MariaDB 10.2](#) (based on MySQL 5.7) introduced persistent AUTO\_INCREMENT ([MDEV-6076](#)) in a GA release before MySQL 8.0. [MariaDB 10.3](#) (based on MySQL 5.7) introduced instant ADD COLUMN ([MDEV-11369](#)) before MySQL.
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**

```
string
```
- 

## innodb\_write\_io\_threads

- **Description:** Number of I/O threads for XtraDB/InnoDB writes. You may on rare occasions need to reduce this default on Linux systems running multiple MariaDB servers to avoid exceeding system limits.
  - **Commandline:**

```
--innodb-write-io-threads=#
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**

```
numeric
```
  - **Default Value:**

```
4
```
  - **Range:**

```
1  
to  
64
```
- 

## 4.3.2.5 InnoDB Server Status Variables

### Contents

1. [Innodb\\_adaptive\\_hash\\_cells](#)
2. [Innodb\\_adaptive\\_hash\\_hash\\_searches](#)
3. [Innodb\\_adaptive\\_hash\\_heap\\_buffers](#)
4. [Innodb\\_adaptive\\_hash\\_non\\_hash\\_searches](#)
5. [Innodb\\_available\\_undo\\_logs](#)
6. [Innodb\\_background\\_log\\_sync](#)
7. [Innodb\\_buffer\\_pool\\_bytes\\_data](#)
8. [Innodb\\_buffer\\_pool\\_bytes\\_dirty](#)

9. Innodb\_buffer\_pool\_dump\_status
10. Innodb\_buffer\_pool\_load\_incomplete
11. Innodb\_buffer\_pool\_load\_status
12. Innodb\_buffer\_pool\_pages\_data
13. Innodb\_buffer\_pool\_pages\_dirty
14. Innodb\_buffer\_pool\_pages\_flushed
15. Innodb\_buffer\_pool\_pages\_LRU\_flushed
16. Innodb\_buffer\_pool\_pages\_LRU\_freed
17. Innodb\_buffer\_pool\_pages\_free
18. Innodb\_buffer\_pool\_pages\_made\_not\_your
19. Innodb\_buffer\_pool\_pages\_made\_young
20. Innodb\_buffer\_pool\_pages\_misc
21. Innodb\_buffer\_pool\_pages\_old
22. Innodb\_buffer\_pool\_pages\_total
23. Innodb\_buffer\_pool\_read\_ahead
24. Innodb\_buffer\_pool\_read\_ahead\_evicted
25. Innodb\_buffer\_pool\_read\_ahead\_rnd
26. Innodb\_buffer\_pool\_read\_requests
27. Innodb\_buffer\_pool\_reads
28. Innodb\_buffer\_pool\_resize\_status
29. Innodb\_buffer\_pool\_wait\_free
30. Innodb\_buffer\_pool\_write\_requests
31. Innodb\_buffered\_aio\_submitted
32. Innodb\_checkpoint\_age
33. Innodb\_checkpoint\_max\_age
34. Innodb\_checkpoint\_target\_age
35. Innodb\_current\_row\_locks
36. Innodb\_data\_fsyncs
37. Innodb\_data\_pending\_fsyncs
38. Innodb\_data\_pending\_reads
39. Innodb\_data\_pending\_writes
40. Innodb\_data\_read
41. Innodb\_data\_reads
42. Innodb\_data\_writes
43. Innodb\_data\_written
44. Innodb\_dblwr\_pages\_written
45. Innodb\_dblwr\_writes
46. Innodb\_deadlocks
47. Innodb\_defragment\_compression\_failures
48. Innodb\_defragment\_count
49. Innodb\_defragment\_failures
50. Innodb\_dict\_tables
51. Innodb\_encryption\_n\_merge\_blocks\_decrypt
52. Innodb\_encryption\_n\_merge\_blocks\_encrypt
53. Innodb\_encryption\_n\_rowlog\_blocks\_decrypt
54. Innodb\_encryption\_n\_rowlog\_blocks\_encrypt
55. Innodb\_encryption\_n\_temp\_blocks\_decrypt
56. Innodb\_encryption\_n\_temp\_blocks\_encrypt
57. Innodb\_encryption\_num\_key\_requests
58. Innodb\_encryption\_rotation\_estimated\_iops
59. Innodb\_encryption\_rotation\_pages\_flushed
60. Innodb\_encryption\_rotation\_pages\_modified
61. Innodb\_encryption\_rotation\_pages\_read\_from
62. Innodb\_encryption\_rotation\_pages\_read\_from
63. Innodb\_have\_atomic\_builtins
64. Innodb\_have\_bzip2
65. Innodb\_have\_lz4
66. Innodb\_have\_lzma
67. Innodb\_have\_lzo
68. Innodb\_have\_punch\_hole
69. Innodb\_have\_snappy
70. Innodb\_history\_list\_length
71. Innodb\_ibuf\_discarded\_delete\_marks
72. Innodb\_ibuf\_discarded\_deletes
73. Innodb\_ibuf\_discarded\_inserts
74. Innodb\_ibuf\_free\_list
75. Innodb\_ibuf\_merged\_delete\_marks
76. Innodb\_ibuf\_merged\_deletes
77. Innodb\_ibuf\_merged\_inserts
78. Innodb\_ibuf\_merges
79. Innodb\_ibuf\_segment\_size
80. Innodb\_ibuf\_size
81. Innodb\_instant\_alter\_column

51. InnoDB\_instance\_and\_column  
82. InnoDB\_log\_waits  
83. InnoDB\_log\_write\_requests  
84. InnoDB\_log\_writes  
85. InnoDB\_lsn\_current  
86. InnoDB\_lsn\_flushed  
87. InnoDB\_lsn\_last\_checkpoint  
88. InnoDB\_master\_thread\_1\_second\_loops  
89. InnoDB\_master\_thread\_10\_second\_loops  
90. InnoDB\_master\_thread\_active\_loops  
91. InnoDB\_master\_thread\_background\_loops  
92. InnoDB\_master\_thread\_idle\_loops  
93. InnoDB\_master\_thread\_main\_flush\_loops  
94. InnoDB\_master\_thread\_sleeps  
95. InnoDB\_max\_trx\_id  
96. InnoDB\_mem\_adaptive\_hash  
97. InnoDB\_mem\_dictionary  
98. InnoDB\_mem\_total  
99. InnoDB\_mutex\_os\_waits  
00. InnoDB\_mutex\_spin\_rounds  
01. InnoDB\_mutex\_spin\_waits  
02. InnoDB\_num\_index\_pages\_written  
03. InnoDB\_num\_non\_index\_pages\_written  
04. InnoDB\_num\_open\_files  
05. InnoDB\_num\_page\_compressed\_trim\_op  
06. InnoDB\_num\_page\_compressed\_trim\_op\_s  
07. InnoDB\_num\_pages\_decrypted  
08. InnoDB\_num\_pages\_encrypted  
09. InnoDB\_num\_pages\_page\_compressed  
10. InnoDB\_num\_pages\_page\_compression\_er  
11. InnoDB\_num\_pages\_page\_decompressed  
12. InnoDB\_num\_pages\_page\_encryption\_error  
13. InnoDB\_oldest\_view\_low\_limit\_trx\_id  
14. InnoDB\_onlineddl\_pct\_progress  
15. InnoDB\_onlineddl\_rowlog\_pct\_used  
16. InnoDB\_onlineddl\_rowlog\_rows  
17. InnoDB\_os\_log\_fsyncs  
18. InnoDB\_os\_log\_pending\_fsyncs  
19. InnoDB\_os\_log\_pending\_writes  
20. InnoDB\_os\_log\_written  
21. InnoDB\_page\_compression\_saved  
22. InnoDB\_page\_compression\_trim\_sect512  
23. InnoDB\_page\_compression\_trim\_sect1024  
24. InnoDB\_page\_compression\_trim\_sect2048  
25. InnoDB\_page\_compression\_trim\_sect4096  
26. InnoDB\_page\_compression\_trim\_sect8192  
27. InnoDB\_page\_compression\_trim\_sect16384  
28. InnoDB\_page\_compression\_trim\_sect32768  
29. InnoDB\_page\_size  
30. InnoDB\_pages\_created  
31. InnoDB\_pages\_read  
32. InnoDB\_pages0\_read  
33. InnoDB\_pages\_written  
34. InnoDB\_purge\_trx\_id  
35. InnoDB\_purge\_undo\_no  
36. InnoDB\_read\_views\_memory  
37. InnoDB\_row\_lock\_current\_waits  
38. InnoDB\_row\_lock\_numbers  
39. InnoDB\_row\_lock\_time  
40. InnoDB\_row\_lock\_time\_avg  
41. InnoDB\_row\_lock\_time\_max  
42. InnoDB\_row\_lock\_waits  
43. InnoDB\_rows\_deleted  
44. InnoDB\_rows\_inserted  
45. InnoDB\_rows\_read  
46. InnoDB\_rows\_updated  
47. InnoDB\_s\_lock\_os\_waits  
48. InnoDB\_s\_lock\_spin\_rounds  
49. InnoDB\_s\_lock\_spin\_waits  
50. InnoDB\_scrub\_background\_page\_reorganiz  
51. InnoDB\_scrub\_background\_page\_split\_failu  
52. InnoDB\_scrub\_background\_page\_split\_failu  
53. InnoDB\_scrub\_background\_page\_split\_failu

- 54. [Innodb\\_scrub\\_background\\_page\\_split\\_failures](#)
- 55. [Innodb\\_scrub\\_background\\_page\\_splits](#)
- 56. [Innodb\\_scrub\\_log](#)
- 57. [Innodb\\_secondary\\_index\\_triggered\\_cluster](#)
- 58. [Innodb\\_secondary\\_index\\_triggered\\_cluster\\_waits](#)
- 59. [Innodb\\_system\\_rows\\_deleted](#)
- 60. [Innodb\\_system\\_rows\\_inserted](#)
- 61. [Innodb\\_system\\_rows\\_read](#)
- 62. [Innodb\\_system\\_rows\\_updated](#)
- 63. [Innodb\\_truncated\\_status\\_writes](#)
- 64. [Innodb\\_undo\\_truncations](#)
- 65. [Innodb\\_x\\_lock\\_os\\_waits](#)
- 66. [Innodb\\_x\\_lock\\_spin\\_rounds](#)
- 67. [Innodb\\_x\\_lock\\_spin\\_waits](#)

See [Server Status Variables](#) for a complete list of status variables that can be viewed with `SHOW STATUS`.

Much of the InnoDB/XtraDB information here can also be seen with a `SHOW ENGINE INNODB STATUS` statement.

See also the [Full list of MariaDB options, system and status variables](#).

### Innodb\_adaptive\_hash\_cells

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the `SHOW ENGINE INNODB STATUS` output.
    - In [MariaDB 5.5](#), this system variable is present in XtraDB.
    - In [MariaDB 10.1](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#)
  - **Removed:** [MariaDB 10.0.0](#)
- 

### Innodb\_adaptive\_hash\_hash\_searches

- **Description:** Hash searches as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the `SHOW ENGINE INNODB STATUS` output.
    - In [MariaDB 5.5](#), this system variable is present in XtraDB.
    - In [MariaDB 10.1](#), [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present. Use the `adaptive_hash_searches` counter in the `information_schema.INNODB_METRICS` table instead.
  - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#), [MariaDB 10.5.0](#)
  - **Removed:** [MariaDB 10.0.0](#)
- 

### Innodb\_adaptive\_hash\_heap\_buffers

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the `SHOW ENGINE INNODB STATUS` output.
  - In [MariaDB 5.5](#), this system variable is present in XtraDB.
  - In [MariaDB 10.1](#) and later, this system variable is not present.
- **Scope:** Global
- **Data Type:**  
numeric
- **Introduced:** [MariaDB 5.5](#)
- **Removed:** [MariaDB 10.0.0](#)

## Innodb\_adaptive\_hash\_non\_hash\_searches

- **Description:** Non-hash searches as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output. From [MariaDB 10.6.2](#), not updated if [innodb\\_adaptive\\_hash\\_index](#) is not enabled (the default).
  - In [MariaDB 10.1](#), [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present. Use the `adaptive_hash_searches_btree` counter in the [information\\_schema.INNODB\\_METRICS](#) table instead.
  - From [MariaDB 10.5](#), this status variable is present.
- **Scope:** Global
- **Data Type:**
  - numeric
- **Introduced:** [MariaDB 5.5](#), [MariaDB 10.5.0](#)
- **Removed:** [MariaDB 10.0.0](#)

## Innodb\_available\_undo\_logs

- **Description:** Total number available InnoDB [undo logs](#). Differs from the [innodb\\_undo\\_logs](#) system variable, which specifies the number of *active* undo logs.
- **Scope:** Global
- **Data Type:**
  - numeric

## Innodb\_background\_log\_sync

- **Description:** As shown in the BACKGROUND THREAD section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 5.5](#), this system variable is present in XtraDB.
  - In [MariaDB 10.1](#), [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
  - In [MariaDB 10.5](#), this system variable was reintroduced.
- **Scope:** Global
- **Data Type:**
  - numeric
- **Introduced:** [MariaDB 5.5](#) (XtraDB only), [MariaDB 10.5.0](#)

## Innodb\_buffer\_pool\_bytes\_data

- **Description:** Number of bytes contained in the [InnoDB buffer pool](#), both dirty (modified) and clean (unmodified). See also [Innodb\\_buffer\\_pool\\_pages\\_data](#), which can contain pages of different sizes in the case of compression.
- **Scope:** Global
- **Data Type:**
  - numeric

## Innodb\_buffer\_pool\_bytes\_dirty

- **Description:** Number of dirty (modified) bytes contained in the [InnoDB buffer pool](#). See also [Innodb\\_buffer\\_pool\\_pages\\_dirty](#), which can contain pages of different sizes in the case of compression.
- **Scope:** Global
- **Data Type:**
  - numeric

## Innodb\_buffer\_pool\_dump\_status

- **Description:** A text description of the progress or final status of the last Innodb buffer pool dump.
  - **Scope:** Global
  - **Data Type:**  
string
  - **Introduced:** MariaDB 10.0.0
- 

## Innodb\_buffer\_pool\_load\_incomplete

- **Description:** Whether or not the loaded buffer pool is incomplete, for example after a shutdown or abort during innodb buffer pool load from file caused an incomplete save.
  - **Scope:** Global
  - **Data Type:**  
boolean
  - **Introduced:** MariaDB 10.3.5
- 

## Innodb\_buffer\_pool\_load\_status

- **Description:** A text description of the progress or final status of the last Innodb buffer pool load.
  - **Scope:** Global
  - **Data Type:**  
string
  - **Introduced:** MariaDB 10.0.0
- 

## Innodb\_buffer\_pool\_pages\_data

- **Description:** Number of InnoDB buffer pool pages which contain data, both dirty (modified) and clean (unmodified). See also [Innodb\\_buffer\\_pool\\_bytes\\_data](#).
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

## Innodb\_buffer\_pool\_pages\_dirty

- **Description:** Number of InnoDB buffer pool pages which contain dirty (modified) data. See also [Innodb\\_buffer\\_pool\\_bytes\\_dirty](#).
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

## Innodb\_buffer\_pool\_pages\_flushed

- **Description:** Number of InnoDB buffer pool pages which have been flushed.
  - **Scope:** Global
  - **Data Type:**  
numeric
-

## Innodb\_buffer\_pool\_pages\_LRU\_flushed

- **Description:** Flush list as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#) , this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) , [MariaDB 10.3](#) , and [MariaDB 10.4](#) , this system variable is not present.
    - In [MariaDB 10.5](#) , this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

## Innodb\_buffer\_pool\_pages\_LRU\_freed

- **Description:** Monitor the number of pages that were freed by a buffer pool LRU eviction scan, without flushing.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 10.6.0](#)
- 

## Innodb\_buffer\_pool\_pages\_free

- **Description:** Number of free [InnoDB buffer pool](#) pages.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

## Innodb\_buffer\_pool\_pages\_made\_not\_young

- **Description:** Pages not young as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#) , this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) , [MariaDB 10.3](#) , and [MariaDB 10.4](#) , this system variable is not present.
    - In [MariaDB 10.5](#) , this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

## Innodb\_buffer\_pool\_pages\_made\_young

- **Description:** Pages made young as shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#) , this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) , [MariaDB 10.3](#) , and [MariaDB 10.4](#) , this system variable is not present.
    - In [MariaDB 10.5](#) , this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
-

## Innodb\_buffer\_pool\_pages\_misc

- **Description:** Number of InnoDB buffer pool pages set aside for internal use.

- **Scope:** Global

- **Data Type:**

numeric

---

## Innodb\_buffer\_pool\_pages\_old

- **Description:** Old database page, as shown in the BUFFER POOL AND MEMORY section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present for XtraDB.

- In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.

- In [MariaDB 10.5](#), this system variable was reintroduced.

- **Scope:** Global

- **Data Type:**

numeric

---

- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

## Innodb\_buffer\_pool\_pages\_total

- **Description:** Total number of InnoDB buffer pool pages.

- **Scope:** Global

- **Data Type:**

numeric

---

## Innodb\_buffer\_pool\_read\_ahead

- **Description:** Number of pages read into the InnoDB buffer pool by the read-ahead background thread.

- **Scope:** Global

- **Data Type:**

numeric

---

## Innodb\_buffer\_pool\_read\_ahead\_evicted

- **Description:** Number of pages read into the InnoDB buffer pool by the read-ahead background thread that were evicted without having been accessed by queries.

- **Scope:** Global

- **Data Type:**

numeric

---

## Innodb\_buffer\_pool\_read\_ahead\_rnd

- **Description:** Number of random read-aheads.

- **Scope:** Global

- **Data Type:**

numeric

---

## Innodb\_buffer\_pool\_read\_requests

- **Description:** Number of requests to read from the [InnoDB buffer pool](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

## Innodb\_buffer\_pool\_reads

- **Description:** Number of reads that could not be satisfied by the [InnoDB buffer pool](#) and had to be read from disk.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

## Innodb\_buffer\_pool\_resize\_status

- **Description:** Progress of the dynamic [InnoDB buffer pool](#) resizing operation. See [Setting Innodb Buffer Pool Size Dynamically](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
  - **Introduced:** [MariaDB 10.2.2](#)
- 

## Innodb\_buffer\_pool\_wait\_free

- **Description:** Number of times InnoDB waited for a free page before reading or creating a page. Normally, writes to the [InnoDB buffer pool](#) happen in the background. When no clean pages are available, dirty pages are flushed first in order to free some up. This counts the numbers of wait for this operation to finish. If this value is not small, look at increasing [innodb\\_buffer\\_pool\\_size](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

## Innodb\_buffer\_pool\_write\_requests

- **Description:** Number of requests to write to the [InnoDB buffer pool](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

## Innodb\_buffered\_aio\_submitted

- **Description:**
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.5.0](#)
- 

## Innodb\_checkpoint\_age

- **Description:** The checkpoint age, as shown in the LOG section of the [SHOW ENGINE INNODB STATUS](#) output. (This is equivalent to subtracting "Last checkpoint at" from "Log sequence number".)
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_checkpoint\_max\_age

- **Description:** Max checkpoint age, as shown in the LOG section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_checkpoint\_target\_age

- **Description:** Checkpoint age target, as shown in the LOG section of the [SHOW ENGINE INNODB STATUS](#) output. XtraDB only. Removed in [MariaDB 10.0](#) and replaced with MySQL 5.6's flushing implementation.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#)
  - **Removed:** [MariaDB 10.0](#)
- 

### Innodb\_current\_row\_locks

- **Description:** Number of current row locks on InnoDB tables as shown in the TRANSACTIONS section of the [SHOW ENGINE INNODB STATUS](#) output. Renamed from [InnoDB\\_row\\_lock\\_numbers](#) in XtraDB 5.5.8-20.1.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

### Innodb\_data\_fsyncs

- **Description:** Number of InnoDB fsync (sync-to-disk) calls. fsync call frequency can be influenced by the [innodb\\_flush\\_method](#) configuration option.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

### Innodb\_data\_pending\_fsyncs

- **Description:** Number of pending InnoDB fsync (sync-to-disk) calls. fsync call frequency can be influenced by the [innodb\\_flush\\_method](#) configuration option.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### `Innodb_data_pending_reads`

- **Description:** Number of pending InnoDB reads.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### `Innodb_data_pending_writes`

- **Description:** Number of pending InnoDB writes.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### `Innodb_data_read`

- **Description:** Number of InnoDB bytes read since server startup (not to be confused with [Innodb\\_data\\_reads](#) ).
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### `Innodb_data_reads`

- **Description:** Number of InnoDB read operations (not to be confused with [Innodb\\_data\\_read](#) ).
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### `Innodb_data_writes`

- **Description:** Number of InnoDB write operations.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### `Innodb_data_written`

- **Description:** Number of InnoDB bytes written since server startup.
- **Scope:** Global
- **Data Type:**  
    numeric

---

### Innodb dblwr\_pages\_written

- **Description:** Number of pages written to the [InnoDB doublewrite buffer](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb dblwr\_writes

- **Description:** Number of writes to the [InnoDB doublewrite buffer](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb\_deadlocks

- **Description:** Total number of InnoDB deadlocks. Deadlocks occur when at least two transactions are waiting for the other to finish, creating a circular dependency. InnoDB usually detects these quickly, returning an error.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#) , this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) , [MariaDB 10.3](#) , and [MariaDB 10.4](#) , this system variable is not present.
    - In [MariaDB 10.5](#) , this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
    numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_defragment\_compression\_failures

- **Description:** Number of defragment re-compression failures. See [Defragmenting InnoDB Tablespace](#)s .
  - **Scope:** Global
  - **Data Type:**  
    numeric
  - **Introduced:** [MariaDB 10.1.1](#)
- 

### Innodb\_defragment\_count

- **Description:** Number of defragment operations. See [Defragmenting InnoDB Tablespace](#)s .
  - **Scope:** Global
  - **Data Type:**  
    numeric
  - **Introduced:** [MariaDB 10.1.1](#)
- 

### Innodb\_defragment\_failures

- **Description:** Number of defragment failures. See [Defragmenting InnoDB Tablespace](#)s .
- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.1.1](#)
- 

#### Innodb\_dict\_tables

- **Description:** Number of entries in the XtraDB data dictionary cache. This Percona XtraDB variable was removed in MariaDB 10/XtraDB 5.6 as it was replaced with MySQL 5.6's [table\\_definition\\_cache](#) implementation.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [XtraDB 5.0.77-b13](#)

- **Removed:** [MariaDB 10.0](#)
- 

#### Innodb\_encryption\_n\_merge\_blocks\_decrypted

- **Description:**

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.1.28](#) , [MariaDB 10.2.9](#) , [MariaDB 10.3.2](#)
- 

#### Innodb\_encryption\_n\_merge\_blocks\_encrypted

- **Description:**

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.1.28](#) , [MariaDB 10.2.9](#) , [MariaDB 10.3.2](#)
- 

#### Innodb\_encryption\_n\_rowlog\_blocks\_decrypted

- **Description:**

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.1.28](#) , [MariaDB 10.2.9](#) , [MariaDB 10.3.2](#)
- 

#### Innodb\_encryption\_n\_rowlog\_blocks\_encrypted

- **Description:**

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.1.28](#) , [MariaDB 10.2.9](#) , [MariaDB 10.3.2](#)
- 

#### Innodb\_encryption\_n\_temp\_blocks\_decrypted

- **Description:**
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.2.26](#) , [MariaDB 10.3.17](#) , [MariaDB 10.4.7](#)
- 

Innodb\_encryption\_n\_temp\_blocks\_encrypted

- **Description:**
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.2.26](#) , [MariaDB 10.3.17](#) , [MariaDB 10.4.7](#)
- 

Innodb\_encryption\_num\_key\_requests

- **Description:** Was not present in [MariaDB 10.5.2](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.2.4](#)
- 

Innodb\_encryption\_rotation\_estimated\_iops

- **Description:** See [Table and Tablespace Encryption](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Removed:** [MariaDB 10.1.3](#)
- 

Innodb\_encryption\_rotation\_pages\_flushed

- **Description:** See [Table and Tablespace Encryption](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Removed:** [MariaDB 10.1.3](#)
- 

Innodb\_encryption\_rotation\_pages\_modified

- **Description:** See [Table and Tablespace Encryption](#) .
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Removed:** [MariaDB 10.1.3](#)
-

## Innodb\_encryption\_rotation\_pages\_read\_from\_cache

- **Description:** See [Table and Tablespace Encryption](#) .
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Removed:** [MariaDB 10.1.3](#)
- 

## Innodb\_encryption\_rotation\_pages\_read\_from\_disk

- **Description:** See [Table and Tablespace Encryption](#) .
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Removed:** [MariaDB 10.1.3](#)
- 

## Innodb\_have\_atomic\_builtin

- **Description:** Whether the server has been built with atomic instructions, provided by the CPU ensuring that critical low-level operations can't be interrupted. XtraDB only.
  - **Scope:** Global
  - **Data Type:**  
boolean
- 

## Innodb\_have\_bzip2

- **Description:** Whether the server has the bzip2 compression method available. See [InnoDB/XtraDB Page Compression](#) .
  - **Scope:** Global
  - **Data Type:**  
boolean
  - **Introduced:** [MariaDB 10.1.0](#)
- 

## Innodb\_have\_lz4

- **Description:** Whether the server has the lz4 compression method available. See [InnoDB/XtraDB Page Compression](#) .
  - **Scope:** Global
  - **Data Type:**  
boolean
  - **Introduced:** [MariaDB 10.1.0](#)
- 

## Innodb\_have\_lzma

- **Description:** Whether the server has the lzma compression method available. See [InnoDB/XtraDB Page Compression](#) .
  - **Scope:** Global
  - **Data Type:**  
boolean
  - **Introduced:** [MariaDB 10.1.0](#)
-

## Innodb\_have\_lzo

- **Description:** Whether the server has the lzo compression method available. See [InnoDB/XtraDB Page Compression](#).
  - **Scope:** Global
  - **Data Type:**  
boolean
  - **Introduced:** [MariaDB 10.1.0](#)
- 

## Innodb\_have\_punch\_hole

- **Description:**
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 10.2.4](#)
- 

## Innodb\_have\_snappy

- **Description:** Whether the server has the snappy compression method available. See [InnoDB/XtraDB Page Compression](#).
  - **Scope:** Global
  - **Data Type:**  
boolean
  - **Introduced:** [MariaDB 10.1.3](#)
- 

## Innodb\_history\_list\_length

- **Description:** History list length as shown in the TRANSACTIONS section of the [SHOW ENGINE INNODB STATUS](#) output. XtraDB only until introduced in [MariaDB 10.5.0](#).
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

## Innodb\_ibuf\_discarded\_delete\_marks

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

## Innodb\_ibuf\_discarded\_deletes

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
  - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.

- In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

#### Innodb\_ibuf\_discarded\_inserts

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

#### Innodb\_ibuf\_free\_list

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

#### Innodb\_ibuf\_merged\_delete\_marks

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

#### Innodb\_ibuf\_merged\_deletes

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

#### Innodb\_ibuf\_merged\_inserts

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
- In [MariaDB 10.5](#), this system variable was reintroduced.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_ibuf\_merges

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
- In [MariaDB 10.5](#), this system variable was reintroduced.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_ibuf\_segment\_size

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
- In [MariaDB 10.5](#), this system variable was reintroduced.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_ibuf\_size

- **Description:** As shown in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
- In [MariaDB 10.5](#), this system variable was reintroduced.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_instant\_alter\_column

- **Description:** See [Instant ADD COLUMN for InnoDB](#).

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.3.2](#)
- 

### Innodb\_log\_waits

- **Description:** Number of times InnoDB was forced to wait for log writes to be flushed due to the log buffer being too small.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

#### Innodb\_log\_write\_requests

- **Description:** Number of requests to write to the InnoDB redo log.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

#### Innodb\_log\_writes

- **Description:** Number of writes to the InnoDB redo log.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

#### Innodb\_lsn\_current

- **Description:** Log sequence number as shown in the LOG section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 
- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

#### Innodb\_lsn\_flushed

- **Description:** Flushed up to log sequence number as shown in the LOG section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 
- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

#### Innodb\_lsn\_last\_checkpoint

- **Description:** Log sequence number last checkpoint as shown in the LOG section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
  - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
  - In [MariaDB 10.5](#), this system variable was reintroduced.
- **Scope:** Global
- **Data Type:**  
numeric

- **Introduced:** MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0
- 

### Innodb\_master\_thread\_1\_second\_loops

- **Description:** As shown in the BACKGROUND THREAD section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 5.5](#), this system variable is present in XtraDB.
  - In [MariaDB 10.1](#) and later, this system variable is not present.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 5.5

- **Removed:** MariaDB 10.0
- 

### Innodb\_master\_thread\_10\_second\_loops

- **Description:** As shown in the BACKGROUND THREAD section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 5.5](#), this system variable is present in XtraDB.
  - In [MariaDB 10.1](#) and later, this system variable is not present

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 5.5

- **Removed:** MariaDB 10.0
- 

### Innodb\_master\_thread\_active\_loops

- **Description:** As shown in the BACKGROUND THREAD section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 10.1](#), this system variable is present in XtraDB.
  - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
  - In [MariaDB 10.5](#), this system variable was reintroduced.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.0.9 (XtraDB-only), MariaDB 10.5.0 :
- 

### Innodb\_master\_thread\_background\_loops

- **Description:** As shown in the BACKGROUND THREAD section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 5.5](#), this system variable is present in XtraDB.
  - In [MariaDB 10.1](#) and later, this system variable is not present

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 5.5

- **Removed:** MariaDB 10.0
- 

### Innodb\_master\_thread\_idle\_loops

- **Description:** As shown in the BACKGROUND THREAD section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 10.1](#), this system variable is present in XtraDB.
  - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
  - In [MariaDB 10.5](#), this system variable was reintroduced.

- **Scope:** Global
  - **Data Type:**  
numeric
- **Introduced:** MariaDB 10.0.9 (XtraDB-only), MariaDB 10.5.0 :
- 

### Innodb\_master\_thread\_main\_flush\_loops

- **Description:** As shown in the BACKGROUND THREAD section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#), this system variable is present in XtraDB.
    - In [MariaDB 10.1](#) and later, this system variable is not present
  - **Scope:** Global
  - **Data Type:**  
numeric
- **Introduced:** MariaDB 5.5
  - **Removed:** MariaDB 10.0
- 

### Innodb\_master\_thread\_sleeps

- **Description:** As shown in the BACKGROUND THREAD section of the [SHOW ENGINE INNODB STATUS](#) output. XtraDB only.
    - In [MariaDB 5.5](#), this system variable is present in XtraDB.
    - In [MariaDB 10.1](#), [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present. Use the `innodb_master_thread_sleeps` counter in the `information_schema.INNODB_METRICS` table instead.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
- **Introduced:** MariaDB 5.5
  - **Removed:** MariaDB 10.0
- 

### Innodb\_max\_trx\_id

- **Description:** As shown in the TRANSACTIONS section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
    - In [MariaDB 10.5](#), this system variable was reintroduced.
  - **Scope:** Global
  - **Data Type:**  
numeric
- **Introduced:** MariaDB 5.5 (XtraDB-only), MariaDB 10.5.0
- 

### Innodb\_mem\_adaptive\_hash

- **Description:** As shown in the BUFFER POOL AND MEMORY section of the [SHOW ENGINE INNODB STATUS](#) output.
  - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
  - In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
  - In [MariaDB 10.5](#), this system variable was reintroduced.
- **Scope:** Global
- **Data Type:**  
numeric

- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_mem\_dictionary

- **Description:** As shown in the BUFFER POOL AND MEMORY section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#), [MariaDB 10.3](#), and [MariaDB 10.4](#), this system variable is not present.
- In [MariaDB 10.5](#), this system variable was reintroduced.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 5.5](#) (XtraDB-only), [MariaDB 10.5.0](#)
- 

### Innodb\_mem\_total

- **Description:** As shown in the BUFFER POOL AND MEMORY section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#) and later, this system variable is not present.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 5.5](#)
- 

### Innodb\_mutex\_os\_waits

- **Description:** Mutex OS waits as shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#) and later, this system variable is not present.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 5.5](#)
- 

### Innodb\_mutex\_spin\_rounds

- **Description:** Mutex spin rounds as shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#) and later, this system variable is not present.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** [MariaDB 5.5](#)
- 

### Innodb\_mutex\_spin\_waits

- **Description:** Mutex spin waits as shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.

- In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
- In [MariaDB 10.2](#) and later, this system variable is not present.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 5.5
- 

Innodb\_num\_index\_pages\_written

- **Description:**
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** MariaDB 10.1.0
- 

Innodb\_num\_non\_index\_pages\_written

- **Description:**
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** MariaDB 10.1.0
- 

Innodb\_num\_open\_files

- **Description:** Number of open files held by InnoDB. InnoDB only.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** MariaDB 10.0.0
- 

Innodb\_num\_page\_compressed\_trim\_op

- **Description:** Number of trim operations performed.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** MariaDB 10.1.0
- 

Innodb\_num\_page\_compressed\_trim\_op\_saved

- **Description:** Number of trim operations not done because of an earlier trim.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** MariaDB 10.1.0
- 

Innodb\_num\_pages\_decrypted

- **Description:** Number of pages page decrypted. See [Table and Tablespace Encryption](#). Was originally named  
    Innodb\_num\_pages\_page\_decrypted  
    in [MariaDB 10.1.3](#), renamed to its current name in [MariaDB 10.1.4](#).
- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1.3
- 

### Innodb\_num\_pages\_encrypted

- **Description:** Number of pages page encrypted. See [Table and Tablespace Encryption](#). Was originally named

Innodb\_num\_pages\_page\_encrypted  
in MariaDB 10.1.3 , renamed to its current name in MariaDB 10.1.4 .

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1.3
- 

### Innodb\_num\_pages\_page\_compressed

- **Description:** Number of pages that are page compressed.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1.0
- 

### Innodb\_num\_pages\_page\_compression\_error

- **Description:** Number of compression errors.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1
- 

### Innodb\_num\_pages\_page\_decompressed

- **Description:** Number of pages compressed with page compression that are decompressed.

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1.0
- 

### Innodb\_num\_pages\_page\_encryption\_error

- **Description:** Number of page encryption errors. See [Table and Tablespace Encryption](#).

- **Scope:** Global

- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1.3

- **Removed:** MariaDB 10.1.4
-

## Innodb\_oldest\_view\_low\_limit\_trx\_id

- **Description:** As shown in the TRANSACTIONS section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

## Innodb\_onlineddl\_pct\_progress

- **Description:** Shows the progress of in-place alter table. It might be not so accurate because in-place alter is highly dependent on disk and buffer pool status. See [Monitoring progress and temporal memory usage of Online DDL in InnoDB](#).
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 10.1.1](#)
- 

## Innodb\_onlineddl\_rowlog\_pct\_used

- **Description:** Shows row log buffer usage in 5-digit integer (10000 means 100.00%). See [Monitoring progress and temporal memory usage of Online DDL in InnoDB](#).
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 10.1.1](#)
- 

## Innodb\_onlineddl\_rowlog\_rows

- **Description:** Number of rows stored in the row log buffer. See [Monitoring progress and temporal memory usage of Online DDL in InnoDB](#).
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 10.1.1](#)
- 

## Innodb\_os\_log\_fsyncs

- **Description:** Number of InnoDB log fsync (sync-to-disk) requests.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

## Innodb\_os\_log\_pending\_fsyncs

- **Description:** Number of pending InnoDB log fsync (sync-to-disk) requests.
- **Scope:** Global
- **Data Type:**  
numeric

---

### Innodb\_os\_log\_pending\_writes

- **Description:** Number of pending InnoDB log writes.
- **Scope:** Global
- **Data Type:**

numeric

---

### Innodb\_os\_log\_written

- **Description:** Number of bytes written to the InnoDB log.
- **Scope:** Global
- **Data Type:**

numeric

---

### Innodb\_page\_compression\_saved

- **Description:** Number of bytes saved by page compression.
- **Scope:**
- **Data Type:**
- **Introduced:** [MariaDB 10.1.0](#) , [MariaDB 10.0.15](#) Fusion-io

---

### Innodb\_page\_compression\_trim\_sect512

- **Description:** Number of TRIM operations performed for the page-compression/NVM Compression workload for the 512 byte block-size.
  - **Scope:**
  - **Data Type:**
  - **Introduced:** [MariaDB 10.1.0](#) , [MariaDB 10.0.15](#) Fusion-io
  - **Removed:** [MariaDB 10.2.4](#)
- 

### Innodb\_page\_compression\_trim\_sect1024

- **Description:** Number of TRIM operations performed for the page-compression/NVM Compression workload for the 1K block-size.
  - **Scope:**
  - **Data Type:**
  - **Introduced:** [MariaDB 10.1.2](#) , [MariaDB 10.0.15](#) Fusion-io
  - **Removed:** [MariaDB 10.2.4](#)
- 

### Innodb\_page\_compression\_trim\_sect2048

- **Description:** Number of TRIM operations performed for the page-compression/NVM Compression workload for the 2K block-size.
- **Scope:**
- **Data Type:**
- **Introduced:** [MariaDB 10.1.2](#) , [MariaDB 10.0.15](#) Fusion-io
- **Removed:** [MariaDB 10.2.4](#)

## Innodb\_page\_compression\_trim\_sect4096

- **Description:** Number of TRIM operations performed for the page-compression/NVM Compression workload for the 4K block-size.
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.1.0](#) , [MariaDB 10.0.15](#) Fusion-io  
• **Removed:** [MariaDB 10.2.4](#)
- 

## Innodb\_page\_compression\_trim\_sect8192

- **Description:** Number of TRIM operations performed for the page-compression/NVM Compression workload for the 8K block-size.
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.1.2](#) , [MariaDB 10.0.15](#) Fusion-io  
• **Removed:** [MariaDB 10.2.4](#)
- 

## Innodb\_page\_compression\_trim\_sect16384

- **Description:** Number of TRIM operations performed for the page-compression/NVM Compression workload for the 16K block-size.
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.1.2](#) , [MariaDB 10.0.15](#) Fusion-io  
• **Removed:** [MariaDB 10.2.4](#)
- 

## Innodb\_page\_compression\_trim\_sect32768

- **Description:** Number of TRIM operations performed for the page-compression/NVM Compression workload for the 32K block-size.
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.1.2](#) , [MariaDB 10.0.15](#) Fusion-io  
• **Removed:** [MariaDB 10.2.4](#)
- 

## Innodb\_page\_size

- **Description:** Page size used by InnoDB. Defaults to 16KB, can be compiled with a different value.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

## Innodb\_pages\_created

- **Description:** Number of InnoDB pages created.
- **Scope:** Global
- **Data Type:**  
    numeric

---

### Innodb\_pages\_read

- **Description:** Number of InnoDB pages read.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb\_pages0\_read

- **Description:** Counter for keeping track of reads of the first page of InnoDB data files, because the original implementation of data-at-rest-encryption for InnoDB introduced new code paths for reading the pages. Removed in [MariaDB 10.4.0](#) as the extra reads of the first page were removed, and the encryption subsystem will be initialized whenever we first read the first page of each data file, in `fil_node_open_file()`.
  - **Scope:** Global
  - **Data Type:**  
    numeric
  - **Introduced:** [MariaDB 10.2.4](#) , [MariaDB 10.1.21](#)
  - **Removed:** [MariaDB 10.4.0](#)
- 

### Innodb\_pages\_written

- **Description:** Number of InnoDB pages written.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb\_purge\_trx\_id

- **Description:** Purge transaction id as shown in the TRANSACTIONS section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#) , this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**  
    numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

### Innodb\_purge\_undo\_no

- **Description:** As shown in the TRANSACTIONS section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#) , this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**  
    numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

### Innodb\_read\_views\_memory

- **Description:** As shown in the BUFFER POOL AND MEMORY section of the [SHOW ENGINE INNODB STATUS](#) output. Shows the total of memory in bytes allocated for the InnoDB read view.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5.32](#)
- 

#### Innodb\_row\_lock\_current\_waits

- **Description:** Number of pending row lock waits on InnoDB tables.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

#### Innodb\_row\_lock\_numbers

- **Description:** Number of current row locks on InnoDB tables as shown in the TRANSACTIONS section of the [SHOW ENGINE INNODB STATUS](#) output. Renamed to [InnoDB\\_current\\_row\\_locks](#) in XtraDB 5.5.10-20.1.
  - **Scope:** Global
  - **Data Type:**  
numeric
  - **Introduced:** [MariaDB 5.5](#) / XtraDB 5.5.8-20
  - **Removed:** [MariaDB 5.5](#) / XtraDB 5.5.10-20.1
- 

#### Innodb\_row\_lock\_time

- **Description:** Total time in milliseconds spent getting InnoDB row locks.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

#### Innodb\_row\_lock\_time\_avg

- **Description:** Average time in milliseconds spent getting an InnoDB row lock.
  - **Scope:** Global
  - **Data Type:**  
numeric
- 

#### Innodb\_row\_lock\_time\_max

- **Description:** Maximum time in milliseconds spent getting an InnoDB row lock.
  - **Scope:** Global
  - **Data Type:**  
numeric
-

### Innodb\_row\_lock\_waits

- **Description:** Number of times InnoDB had to wait before getting a row lock.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb\_rows\_deleted

- **Description:** Number of rows deleted from InnoDB tables.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb\_rows\_inserted

- **Description:** Number of rows inserted into InnoDB tables.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb\_rows\_read

- **Description:** Number of rows read from InnoDB tables.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb\_rows\_updated

- **Description:** Number of rows updated in InnoDB tables.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### Innodb\_s\_lock\_os\_waits

- **Description:** As shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**  
    numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

### Innodb\_s\_lock\_spin\_rounds

- **Description:** As shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#) , this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**
    - numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

#### Innodb\_s\_lock\_spin\_waits

- **Description:** As shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#) , this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**
    - numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

#### Innodb\_scrub\_background\_page\_reorganizations

- **Description:** See [Table and Tablespace Encryption](#) .
  - **Scope:** Global
  - **Data Type:**
    - numeric
  - **Introduced:** [MariaDB 10.1.3](#)
  - **Removed:** [MariaDB 10.5.2](#)
- 

#### Innodb\_scrub\_background\_page\_split\_failures\_missing\_index

- **Description:** See [Table and Tablespace Encryption](#) .
  - **Scope:** Global
  - **Data Type:**
    - numeric
  - **Introduced:** [MariaDB 10.1.3](#)
  - **Removed:** [MariaDB 10.5.2](#)
- 

#### Innodb\_scrub\_background\_page\_split\_failures\_out\_of\_filespace

- **Description:** See [Table and Tablespace Encryption](#) .
  - **Scope:** Global
  - **Data Type:**
    - numeric
  - **Introduced:** [MariaDB 10.1.3](#)
  - **Removed:** [MariaDB 10.5.2](#)
- 

#### Innodb\_scrub\_background\_page\_split\_failures\_underflow

- **Description:** See [Table and Tablespace Encryption](#) .
- **Scope:** Global
- **Data Type:**
  - numeric

- **Introduced:** MariaDB 10.1.3
  - **Removed:** MariaDB 10.5.2
- 

### Innodb\_scrub\_background\_page\_split\_failures\_unknown

- **Description:** See [Table and Tablespace Encryption](#) .
- **Scope:** Global
- **Data Type:**

numeric

- **Introduced:** MariaDB 10.2.5 , MariaDB 10.1.3
  - **Removed:** MariaDB 10.5.2
- 

### Innodb\_scrub\_background\_page\_splits

- **Description:** See [Table and Tablespace Encryption](#) .
- **Scope:** Global
- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1
  - **Removed:** MariaDB 10.5.2
- 

### Innodb\_scrub\_log

- **Description:**
- **Scope:** Global
- **Data Type:**

numeric

- **Introduced:** MariaDB 10.2.4
  - **Removed:** MariaDB 10.5.2
- 

### Innodb\_secondary\_index\_triggered\_cluster\_reads

- **Description:** Used to track the effectiveness of the Prefix Index Queries Optimization ( [MDEV-6929](#) )
- **Scope:** Global
- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1.2
- 

### Innodb\_secondary\_index\_triggered\_cluster\_reads\_avoided

- **Description:** Used to track the effectiveness of the Prefix Index Queries Optimization ( [MDEV-6929](#) )
- **Scope:** Global
- **Data Type:**

numeric

- **Introduced:** MariaDB 10.1.2
- 

### Innodb\_system\_rows\_deleted

- **Description:**
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.0.15](#)
- 

#### Innodb\_system\_rows\_inserted

- **Description:**
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.0.15](#)
- 

#### Innodb\_system\_rows\_read

- **Description:**
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.0.15](#)
- 

#### Innodb\_system\_rows\_updated

- **Description:**
  - **Scope:**
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.0.15](#)
- 

#### Innodb\_truncated\_status\_writes

- **Description:** Number of times output from [SHOW ENGINE INNODB STATUS](#) has been truncated.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 5.5](#)
- 

#### Innodb\_undo\_truncations

- **Description:** Number of undo tablespace truncation operations.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.3.10](#)
-

## Innodb\_x\_lock\_os\_waits

- **Description:** As shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**
    - numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

## Innodb\_x\_lock\_spin\_rounds

- **Description:** As shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**
    - numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

## Innodb\_x\_lock\_spin\_waits

- **Description:** As shown in the SEMAPHORES section of the [SHOW ENGINE INNODB STATUS](#) output.
    - In [MariaDB 5.5](#) and [MariaDB 10.1](#), this system variable is present in XtraDB.
    - In [MariaDB 10.2](#) and later, this system variable is not present.
  - **Scope:** Global
  - **Data Type:**
    - numeric
  - **Introduced:** [MariaDB 5.5](#)
- 

## 4.3.2.6 AUTO\_INCREMENT Handling in InnoDB

### Contents

1. [AUTO\\_INCREMENT Lock Modes](#)
  1. [Traditional Lock Mode](#)
  2. [Consecutive Lock Mode](#)
  3. [Interleaved Lock Mode](#)
2. [Setting AUTO\\_INCREMENT Values](#)
3. [See Also](#)

## AUTO\_INCREMENT Lock Modes

The `innodb_autoinc_lock_mode` system variable determines the lock mode when generating `AUTO_INCREMENT` values for InnoDB tables. These modes allow InnoDB to make significant performance optimizations in certain circumstances.

The `innodb_autoinc_lock_mode` system variable may be removed in a future release. See [MDEV-19577](#) for more information.

### Traditional Lock Mode

When `innodb_autoinc_lock_mode` is set to

0

, InnoDB uses the traditional lock mode.

In this mode, InnoDB holds a table-level lock for all `INSERT` statements until the statement completes.

### Consecutive Lock Mode

When `innodb_autoinc_lock_mode` is set to

1

, InnoDB uses the consecutive lock mode.

In this mode, InnoDB holds a table-level lock for all bulk `INSERT` statements (such as `LOAD DATA` or `INSERT ... SELECT`) until the end of the statement. For simple `INSERT` statements, no table-level lock is held. Instead, a lightweight mutex is used which scales significantly better. This is the default setting.

## Interleaved Lock Mode

When `innodb_autoinc_lock_mode` is set to

2

, InnoDB uses the interleaved lock mode.

In this mode, InnoDB does not hold any table-level locks at all. This is the fastest and most scalable mode, but is not safe for `statement-based` replication.

## Setting AUTO\_INCREMENT Values

The `AUTO_INCREMENT` value for an InnoDB table can be set for a table by executing the `ALTER TABLE` statement and specifying the `AUTO_INCREMENT` table option. For example:

```
ALTER TABLE tab AUTO_INCREMENT=100;
```

However, in MariaDB 10.2.3 and before, InnoDB stores the table's `AUTO_INCREMENT` counter in memory. In these versions, when the server restarts, the counter is re-initialized to the highest value found in the table. This means that the above operation can be undone if the server is restarted before any rows are written to the table.

In MariaDB 10.2.4 and later, the `AUTO_INCREMENT` counter is persistent, so this restriction is no longer present. Persistent, however, does not mean transactional. Gaps may still occur in some cases, such as if a `INSERT IGNORE` statement fails, or if a user executes `ROLLBACK` or `ROLLBACK TO SAVEPOINT`.

For example:

```
CREATE TABLE t1 (pk INT AUTO_INCREMENT PRIMARY KEY, i INT, UNIQUE (i)) ENGINE=InnoDB;

INSERT INTO t1 (i) VALUES (1),(2),(3);
INSERT IGNORE INTO t1 (pk, i) VALUES (100,1);
Query OK, 0 rows affected, 1 warning (0.099 sec)

SELECT * FROM t1;
+---+---+
| pk | i   |
+---+---+
| 1  | 1   |
| 2  | 2   |
| 3  | 3   |
+---+---+

SHOW CREATE TABLE t1\G
***** 1. row *****
    Table: t1
Create Table: CREATE TABLE `t1` (
  `pk` int(11) NOT NULL AUTO_INCREMENT,
  `i` int(11) DEFAULT NULL,
  PRIMARY KEY (`pk`),
  UNIQUE KEY `i` (`i`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1
```

If the server is restarted at this point, then the `AUTO_INCREMENT` counter will revert to

101

, which is the persistent value set as part of the failed `INSERT IGNORE`.

```
# Restart server
SHOW CREATE TABLE t1\G
***** 1. row *****
    Table: t1
Create Table: CREATE TABLE `t1` (
  `pk` int(11) NOT NULL AUTO_INCREMENT,
  `i` int(11) DEFAULT NULL,
  PRIMARY KEY (`pk`),
  UNIQUE KEY `i` (`i`)
) ENGINE=InnoDB AUTO_INCREMENT=101 DEFAULT CHARSET=latin1
```

## See Also

- [AUTO\\_INCREMENT](#)
- [AUTO\\_INCREMENT FAQ](#)
- [LAST\\_INSERT\\_ID](#)
- Sequences - an alternative to auto\_increment available from [MariaDB 10.3](#)

## 4.3.2.7 InnoDB Buffer Pool

### Contents

1. [How the Buffer Pool Works](#)
2. [innodb\\_buffer\\_pool\\_size](#)
3. [innodb\\_buffer\\_pool\\_instances](#)
4. [innodb\\_old\\_blocks\\_pct](#) and [innodb\\_old\\_blocks\\_time](#)
5. [Dumping and Restoring the Buffer Pool](#)
6. [See Also](#)

The InnoDB buffer pool is a key component for optimizing MariaDB. It stores data and indexes, and you usually want it as large as possible so as to keep as much of the data and indexes in memory, reducing disk IO, as main bottleneck.

### How the Buffer Pool Works

The buffer pool attempts to keep frequently-used blocks in the buffer, and so essentially works as two sublists, a *new* sublist of recently-used information, and an *old* sublist of older information. By default, 37% of the list is reserved for the old list.

When new information is accessed that doesn't appear in the list, it is placed at the top of the old list, the oldest item in the old list is removed, and everything else bumps back one position in the list.

When information is accessed that appears in the *old* list, it is moved to the top the new list, and everything above moves back one position.

### innodb\_buffer\_pool\_size

The most important [server system variable](#) is [innodb\\_buffer\\_pool\\_size](#). This size should contain most of the active data set of your server so that SQL request can work directly with information in the buffer pool cache. Starting at several gigabytes of memory is a good starting point if you have that RAM available. Once warmed up to its normal load there should be very few [innodb\\_buffer\\_pool\\_reads](#) compared to [innodb\\_buffer\\_pool\\_read\\_requests](#). Look how these values change over a minute. If the change in [innodb\\_buffer\\_pool\\_reads](#) is less than 1% of the change in [innodb\\_buffer\\_pool\\_read\\_requests](#) then you have a good amount of usage. If you are getting the status variable [innodb\\_buffer\\_pool\\_wait\\_free](#) increasing then you don't have enough buffer pool (or your flushing isn't occurring frequently enough).

Be aware that before [MariaDB 10.4.4](#) the total memory allocated is about 10% more than the specified size as extra space is also reserved for control structures and buffers.

The larger the size, the longer it will take to initialize. On a modern 64-bit server with a 10GB memory pool, this can take five seconds or more. Increasing [innodb\\_buffer\\_pool\\_chunk\\_size](#) by several factors will reduce this significantly.

Make sure that the size is not too large, causing swapping. The benefit of a larger buffer pool size is more than undone if your operating system is regularly swapping.

Since [MariaDB 10.2.2](#), the buffer pool can be set dynamically, and new variables are introduced that may affect the size and performance. See [Setting Innodb Buffer Pool Size Dynamically](#).

### innodb\_buffer\_pool\_instances

The functionality described below was disabled in [MariaDB 10.5](#), and removed in [MariaDB 10.6](#), as the original reasons for splitting the buffer pool have mostly gone away.

If [innodb\\_buffer\\_pool\\_size](#) is set to more than 1GB, [innodb\\_buffer\\_pool\\_instances](#) divides the InnoDB buffer pool into a specific number of instances. The default was 1 in [MariaDB 5.5](#), but for large systems with buffer pools of many gigabytes, many instances can help reduce contention concurrency. The default is 8 in [MariaDB 10.0](#), with the exception of 32-bit Windows, where it depends on the value of [innodb\\_buffer\\_pool\\_size](#). Each instance manages its own data structures and takes an equal portion of the total buffer pool size, so for example if [innodb\\_buffer\\_pool\\_size](#) is 4GB and [innodb\\_buffer\\_pool\\_instances](#) is set to 4, each instance will be 1GB. Each instance should ideally be at least 1GB in size.

### innodb\_old\_blocks\_pct and innodb\_old\_blocks\_time

The default 37% reserved for the old list can be adjusted by changing the value of [innodb\\_old\\_blocks\\_pct](#). It can accept anything between 5% and 95%.

The [innodb\\_old\\_blocks\\_time](#) variable specifies the delay before a block can be moved from the old to the new sublist.

0

means no delay, while the default has been set to

1000

Before changing either of these values from their defaults, make sure you understand the impact and how your system currently uses the buffer. Their

main reason for existence is to reduce the impact of full table scans, which are usually infrequent, but large, and previously could clear everything from the buffer. Setting a non-zero delay could help in situations where full table scans are performed in quick succession.

Temporarily changing these values can also be useful to avoid the negative impact of a full table scan, as explained in [InnoDB logical backups](#).

## Dumping and Restoring the Buffer Pool

When the server starts, the buffer pool is empty. As it starts to access data, the buffer pool will slowly be populated. As more data will be accessed, the most frequently accessed data will be put into the buffer pool, and old data may be evicted. This means that a certain period of time is necessary before the buffer pool is really useful. This period of time is called the warmup.

Since [MariaDB 10.0](#), InnoDB can dump the buffer pool before the server shuts down, and restore it when it starts again. If this feature is used (default since [MariaDB 10.2](#)), no warmup is necessary. Use the `innodb_buffer_pool_dump_at_shutdown` and `innodb_buffer_pool_load_at_startup` system variables to enable or disable the buffer pool dump at shutdown and the restore at startup respectively.

It is also possible to dump the InnoDB buffer pool at any moment while the server is running, and it is possible to restore the last buffer pool dump at any moment. To do this, the special `innodb_buffer_pool_dump_now` and `innodb_buffer_pool_load_now` system variables can be set to ON. When selected, their value is always OFF.

A buffer pool restore, both at startup or at any other moment, can be aborted by setting `innodb_buffer_pool_load_abort` to ON.

The file which contains the buffer pool dump is specified via the `innodb_buffer_pool_filename` system variable.

## See Also

- [InnoDB Change Buffering](#)
- [Information Schema INNODB\\_BUFFER\\_POOL\\_STATS Table](#)
- [Setting Innodb Buffer Pool Size Dynamically](#)

### 4.3.2.8 InnoDB Change Buffering

Benchmarks attached to [MDEV-19514](#) show that the change buffer sometimes reduces performance, and in the best case seem to bring a few per cent improvement to throughput. However, such improvement could come with a price: If the buffered changes are never merged ([MDEV-19514](#), motivated by the reduction of random crashes and the removal of an `innodb_force_recovery` option that can inflict further corruption), then the InnoDB system tablespace can grow out of control ([MDEV-21952](#)).

Because of all this, the change buffer has been disabled by default from [MariaDB 10.5.15](#), [MariaDB 10.6.7](#), [MariaDB 10.7.3](#) and [MariaDB 10.8.2](#) ([MDEV-27734](#)) and the feature is deprecated and ignored from [MariaDB 10.9.0](#) ([MDEV-27735](#)).

## Contents

1. [Change Buffer Related Status Variables](#)
2. [See Also](#)

INSERT, UPDATE and DELETE statements can be particularly heavy operations to perform, as all indexes need to be updated after each change. For this reason these changes are often buffered.

Pages are modified in the `buffer pool`, and not immediately on disk. When rows are deleted, a flag is set, thus rows are not immediately deleted on disk. Later the changes will be written to disk ("flushed") by InnoDB background threads. Pages that have been modified in memory and not yet flushed are called dirty pages. The buffering of data changes is called Change Buffer.

Before [MariaDB 5.5](#), only inserted rows could be buffered, so this buffer was called Insert Buffer. The old name still appears in several places, for example in the output of `SHOW ENGINE INNODB STATUS`.

The change buffer only contains changes to the indexes. Inserts to UNIQUE secondary indexes cannot be buffered unless `unique_checks=0` is used. Delete-mark and purge buffering of UNIQUE secondary indexes is allowed.

The Change Buffer is an optimization because:

- A page can be modified several times in memory and be flushed to disk only once.
- Dirty pages are flushed together, so the number of IO operations is lower.

If the server crashes, usually the Change Buffer is not empty. However, changes are not lost because they are written to the transaction logs, so they can be applied at server restart.

The main server system variable here is `innodb_change_buffering`, which determines which form of change buffering, if any, to use.

The following settings are available:

- inserts
  - Only buffer insert operations
- deletes
  - Only buffer delete operations
- changes
  - Buffer both insert and delete operations
- purges
  - Buffer the actual physical deletes that occur in the background

- all
  - Buffer inserts, deletes and purges. Default setting from [MariaDB 5.5](#) until [MariaDB 10.5.14](#) , [MariaDB 10.6.6](#) , [MariaDB 10.7.2](#) and [MariaDB 10.8.1](#) .
- none
  - Don't buffer any operations. Default from [MariaDB 10.5.15](#) , [MariaDB 10.6.7](#) , [MariaDB 10.7.3](#) and [MariaDB 10.8.2](#) .

Modifying the value of this variable only affects the buffering of new operations. The merging of already buffered changes is not affected.

The `innodb_change_buffer_max_size` server system variable, determines the maximum size of the change buffer, expressed as a percentage of the buffer pool.

## Change Buffer Related Status Variables

- `Innodb_buffer_pool_pages_dirty` : Number of dirty pages in the Change Buffer.
- `innodb_buffer_pool_bytes_dirty` : Total size of the dirty pages, in bytes.
- `innodb_buffer_pool_wait_free` : How many times InnoDB was forced to flush dirty pages to write new data, because the buffer pool had no more free pages.

## See Also

- [InnoDB Buffer Pool](#)

### 4.3.2.9 InnoDB Doublewrite Buffer

The [InnoDB](#) doublewrite buffer was implemented to recover from half-written pages. This can happen when there's a power failure while InnoDB is writing a page to disk. On reading that page, InnoDB can discover the corruption from the mismatch of the page checksum. However, in order to recover, an intact copy of the page would be needed.

The double write buffer provides such a copy.

Whenever InnoDB flushes a page to disk, it is first written to the double write buffer. Only when the buffer is safely flushed to disk will InnoDB write the page to the final destination. When recovering, InnoDB scans the double write buffer and for each valid page in the buffer checks if the page in the data file is valid too.

## Doublewrite Buffer Settings

To turn off the doublewrite buffer, set the `innodb_doublewrite` system variable to

`0`

. This is safe on filesystems that write pages atomically - that is, a page write fully succeeds or fails. But with other filesystems, it is not recommended for production systems. An alternative option is atomic writes. See [atomic write support](#) for more details.

### 4.3.2.10 InnoDB Tablespaces

#### 4.3.2.10.1 InnoDB System Tablespaces

##### Contents

1. [Changing Sizes](#)
  1. [Increasing the Size](#)
  2. [Decreasing the Size](#)
2. [Using Raw Disk Partitions](#)
  1. [Raw Disk Partitions on Windows](#)
3. [System Tables within the InnoDB System Tablespace](#)

When InnoDB needs to store general information relating to the system as a whole, rather than a specific table, the specific file it writes to is the system tablespace. By default, this is the

`ibdata1`

file located in the data directory, (as defined by either the

`datadir`

or

`innodb_data_home_dir`

system variables). InnoDB uses the system tablespace to store the data dictionary, change buffer, and undo logs.

You can define the system tablespace filename or filenames, size and other options by setting the

`innodb_data_file_path`

system variable. This system variable can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
innodb_data_file_path=ibdata1:50M:autoextend
```

This system variable defaults to the file

```
ibdata1
, and it defaults to a minimum size of
12M
, and it defaults with the
autoextend
attribute enabled.
```

## Changing Sizes

InnoDB defaults to allocating 12M to the

```
ibdata1
file for the system tablespace. While this is sufficient for most use cases, it may not be for all. You may find after using MariaDB for a while that the allocation is too small for the system tablespace or it grows too large for your disk. Fortunately, you can adjust this size as need later.
```

### Increasing the Size

When setting the

```
innodb_data_file_path
```

system variable, you can define a size for each file given. In cases where you need a larger system tablespace, add the `autoextend` option to the last value.

```
[mariadb]
...
innodb_data_file_path=ibdata1:12M;ibdata2:50M:autoextend
```

Under this configuration, when the last system tablespace grows beyond the size allocation, InnoDB increases the size of the file by increments. To control the allocation increment, set the

```
innodb_autoextend_increment
```

system variable.

### Decreasing the Size

In cases where the InnoDB system tablespace has grown too large, the process to reduce it in size is a little more complicated than increasing the size. MariaDB does not allow you to remove data from the tablespace file itself. Instead you need to delete the tablespace files themselves, then restore the database from backups.

The backup utility mysqldump produces backup files containing the SQL statements needed to recreate the database. As a result, it restores a database with the bare minimum data rather than any additional information that might have built up in the tablespace file.

Use mysqldump to backup all of your InnoDB database tables, including the system tables in the

```
mysql
database that use InnoDB. You can find out what they are using the Information Schema.
```

```
SELECT TABLE_NAME FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'mysql' AND ENGINE = 'InnoDB';
```

If you only use InnoDB, you may find it easier to back up all databases and tables.

```
$ mysqldump -u root -p --all-databases > full-backup.sql
```

Then stop the MariaDB Server and remove the InnoDB tablespace files. In the data directory or the InnoDB data home directory, delete all the `ibdata` and

```
ib_log  
files as well as any file with an  
.ibd  
or  
.frm  
extension.
```

Once this is done, restart the server and import the dump file:

```
$ mysql -u root -p < full-backup.sql
```

## Using Raw Disk Partitions

Instead of having InnoDB write to the file system, you can set it to use raw disk partitions. On Windows and some Linux distributions, this allows you to perform non-buffered I/O without the file system overhead. Note that in many use cases this may not actually improve performance. Run tests to verify if there are any real gains for your application usage.

To enable a raw disk partition, first start MariaDB with the

```
newraw  
option set on the tablespace. For example:
```

```
[mariadb]  
...  
innodb_data_file_path=/dev/sdc:10Gnewraw
```

When the MariaDB Server starts, it initializes the partition. Don't create or change any data, (any data written to InnoDB at this stage will be lost on restart). Once the server has successfully started, stop it then edit the configuration file again, changing the

```
newraw  
keyword to  
raw
```

```
[mariadb]  
...  
innodb_data_file_path=/dev/sdc:10Graw
```

When you start MariaDB again, it'll read and write InnoDB data to the given disk partition instead of the file system.

## Raw Disk Partitions on Windows

When defining a raw disk partition for InnoDB on the Windows operating system, use the same procedure as defined above, but when defining the path for the

```
innodb_data_file_path
```

system variable, use

```
./
```

at the start. For example:

```
[mariadb]  
...  
innodb_data_file_path=//./E::10Graw
```

The given path is synonymous with the Windows syntax for accessing the physical drive.

## System Tables within the InnoDB System Tablespace

InnoDB creates some system tables within the InnoDB System Tablespace:

- `SYS_DATAFILES`
- `SYS_FOREIGN`
- `SYS_FOREIGN_COLS`

- SYS\_TABLESPACES
- SYS\_VIRTUAL
- SYS\_ZIP\_DICT
- SYS\_ZIP\_DICT\_COLS

These tables cannot be queried. However, you might see references to them in some places, such as in the

[INNODB\\_SYS\\_TABLES](#)

table in the

[information\\_schema](#)

database.

## 4.3.2.10.2 InnoDB File-Per-Table Tablespaces

### Contents

1. [File-Per-Table Tablespace Locations](#)
2. [Copying Transportable Tablespaces](#)
  1. [Copying Transportable Tablespaces for Non-partitioned Tables](#)
    1. [Exporting Transportable Tablespaces for Non-partitioned Tables](#)
    2. [Importing Transportable Tablespaces for Non-partitioned Tables](#)
  2. [Copying Transportable Tablespaces for Partitioned Tables](#)
    1. [Exporting Transportable Tablespaces for Partitioned Tables](#)
    2. [Importing Transportable Tablespaces for Partitioned Tables](#)
      1. [For Each Partition](#)
  3. [Known Problems with Copying Transportable Tablespaces](#)
    1. [Differing Storage Formats for Temporal Columns](#)
    2. [Differing ROW\\_FORMAT Values](#)
    3. [Foreign Key Constraints](#)
  3. [Tablespace Encryption](#)
  4. [See Also](#)

When you create a table using the [InnoDB storage engine](#), data written to that table is stored on the file system in a data file called a tablespace. Tablespace files contain both the data and indexes.

When

[innodb\\_file\\_per\\_table=ON](#)

is set, InnoDB uses one tablespace file per InnoDB table. These tablespace files have the [.ibd](#) extension. When

[innodb\\_file\\_per\\_table=OFF](#)

is set, InnoDB stores all tables in the [InnoDB system tablespace](#).

InnoDB versions in MySQL 5.7 and above also support an additional type of tablespace called [general tablespaces](#) that are created with

[CREATE TABLESPACE](#)

. However, InnoDB versions in MariaDB Server do not currently support general tablespaces or

## File-Per-Table Tablespace Locations

By default, InnoDB's file-per-table tablespaces are created in the system's data directory, which is defined by the

`datadir`

system variable. The system variable

`innodb_data_home_dir`

will not change the location of file-per-table tablespaces.

In the event that you have a specific tablespace that you need stored in a dedicated path, you can set the location using the

`DATA DIRECTORY`

table option when you create the table.

For instance,

```
CREATE TABLE test.t1 (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50)
) ENGINE=InnoDB
DATA DIRECTORY = "/data/contact";
```

MariaDB then creates a database directory on the configured path and the file-per-table tablespace will be created inside that directory. On Unix-like operating systems, you can see the file using the `ls` command:

```
# ls -al /data/contact/test
drwxrwx--- 2 mysql mysql 4096 Dec 8 18:46 .
drwxr-xr-x 3 mysql mysql 4096 Dec 8 18:46 ..
-rw-rw---- 1 mysql mysql 98304 Dec 8 20:41 t1.ibd
```

Note, the system user that runs the MariaDB Server process (which is usually  
`mysql`) must have write permissions on the given path.

## Copying Transportable Tablespaces

InnoDB's file-per-table tablespaces are transportable, which means that you can copy a file-per-table tablespace from one MariaDB Server to another server. You may find this useful in cases where you need to transport full tables between servers and don't want to use backup tools like

`mariabackup`

or

`mysqldump`

. In fact, this process can even be used with

`mariabackup`

in some cases, such as when `restoring partial backups` or when `restoring individual tables or partitions from a backup`.

## Copying Transportable Tablespaces for Non-partitioned Tables

You can copy the transportable tablespace of a non-partitioned table from one server to another by exporting the tablespace file from the original server, and then importing the tablespace file into the new server.

### Exporting Transportable Tablespaces for Non-partitioned Tables

You can export a non-partitioned table by locking the table and copying the table's  
`.ibd`

and

.cfg

files from the relevant [tablespace location](#) for the table to a backup location. For example, the process would go like this:

- First, use the

```
FLUSH TABLES ... FOR EXPORT
```

statement on the target table:

```
FLUSH TABLES test.t1 FOR EXPORT;
```

This forces the server to close the table and provides your connection with a read lock on the table.

- Then, while your connection still holds the lock on the table, copy the tablespace file and the metadata file to a safe directory:

```
# cp /data/contacts/test/t1.ibd /data/saved-tablespaces/
# cp /data/contacts/test/t1.cfg /data/saved-tablespaces/
```

- Then, once you've copied the files, you can release the lock with

```
UNLOCK TABLES
```

:

```
UNLOCK TABLES;
```

## Importing Transportable Tablespaces for Non-partitioned Tables

You can import a non-partitioned table by discarding the table's original tablespace, copying the table's

.ibd

and

.cfg

files from the backup location to the relevant [tablespace location](#) for the table, and then telling the server to import the tablespace. For example, the process would go like this:

- First, on the destination server, you need to create a copy of the table. Use the same

```
CREATE TABLE
```

statement that was used to create the table on the original server:

```
CREATE TABLE test.t1 (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50)
) ENGINE=InnoDB;
```

- Then, use

```
ALTER TABLE ... DISCARD TABLESPACE
```

to discard the new table's tablespace:

```
ALTER TABLE test.t1 DISCARD TABLESPACE;
```

- Then, copy the

.ibd

and

.cfg

files from the original server to the relevant directory on the target MariaDB Server:

```
# scp /data/tablespaces/t1.ibd target-server.com:/var/lib/mysql/test/
# scp /data/tablespaces/t1.cfg target-server.com:/var/lib/mysql/test/
```

File-per-table tablespaces can be imported with just the

.ibd

file in many cases. If you do not have the tablespace's

.cfg

file for whatever reason, then it is usually worth trying to import the tablespace with just the

.ibd

file.

- Then, once the files are in the proper directory on the target server, use

```
ALTER TABLE ... IMPORT TABLESPACE
```

to import the new table's tablespace:

```
ALTER TABLE test.t1 IMPORT TABLESPACE;
```

## Copying Transportable Tablespaces for Partitioned Tables

Currently, MariaDB does not directly support the transport of tablespaces from partitioned tables. See [MDEV-10568](#) for more information about that. It is still possible to transport partitioned tables if we use a workaround. You can copy the transportable tablespaces of a partitioned table from one server to another by exporting the tablespace file of each partition from the original server, and then importing the tablespace file of each partition into the new server.

### Exporting Transportable Tablespaces for Partitioned Tables

You can export a partitioned table by locking the table and copying the

.ibd

and

.cfg

files of each partition from the relevant [tablespace location](#) for the partition to a backup location. For example, the process would go like this:

- First, let's create a test table with some data on the original server:

```
CREATE TABLE test.t2 (
    employee_id INT,
    name VARCHAR(50),
) ENGINE=InnoDB
PARTITION BY RANGE (employee_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);

INSERT INTO test.t2 (name, employee_id) VALUES
    ('Geoff Montee', 1),
    ('Chris Calendar', 6),
    ('Kyle Joiner', 11),
    ('Will Fong', 16);
```

- Then, we need to export the partitioned tablespace from the original server, which follows the same process as exporting non-partitioned tablespaces. That means that we need to use the

```
FLUSH TABLES ... FOR EXPORT
```

statement on the target table:

```
FLUSH TABLES test.t2 FOR EXPORT;
```

This forces the server to close the table and provides your connection with a read lock on the table.

- Then, if we grep the database directory in the data directory for the newly created

t2

table, we can see a number of

.ibd

and

.cfg

files for the table:

```
# ls -l /var/lib/mysql/test/ | grep t2
total 428
-rw-rw--- 1 mysql mysql 827 Dec 5 16:08 t2frm
-rw-rw--- 1 mysql mysql 48 Dec 5 16:08 t2.par
-rw-rw--- 1 mysql mysql 579 Dec 5 18:47 t2#P#p0.cfg
-rw-r----- 1 mysql mysql 98304 Dec 5 16:43 t2#P#p0.ibd
-rw-rw--- 1 mysql mysql 579 Dec 5 18:47 t2#P#p1.cfg
-rw-rw--- 1 mysql mysql 98304 Dec 5 16:08 t2#P#p1.ibd
-rw-rw--- 1 mysql mysql 579 Dec 5 18:47 t2#P#p2.cfg
-rw-rw--- 1 mysql mysql 98304 Dec 5 16:08 t2#P#p2.ibd
-rw-rw--- 1 mysql mysql 579 Dec 5 18:47 t2#P#p3.cfg
-rw-rw--- 1 mysql mysql 98304 Dec 5 16:08 t2#P#p3.ibd
```

- Then, while our connection still holds the lock on the table, we need to copy the tablespace files and the metadata files to a safe directory:

```
$ mkdir /tmp/backup
$ sudo cp /var/lib/mysql/test/*.*ibd /tmp/backup
$ sudo cp /var/lib/mysql/test/*.*cfg /tmp/backup
```

- Then, once we've copied the files, we can release the lock with

UNLOCK TABLES

:

```
UNLOCK TABLES;
```

## Importing Transportable Tablespaces for Partitioned Tables

You can import a partitioned table by creating a placeholder table, discarding the placeholder table's original tablespace, copying the partition's

.ibd

and

.cfg

files from the backup location to the relevant [tablespace location](#) for the placeholder table, and then telling the server to import the tablespace. At that point, the server can exchange the tablespace for the placeholder table with the one for the partition. For example, the process would go like this:

- First, we need to copy the saved tablespace files from the original server to the target server:

```
$ scp /tmp/backup/t2* user@target-host:/tmp/backup
```

- Then, we need to import the partitioned tablespaces onto the target server. The import process for partitioned tables is more complicated than the import process for non-partitioned tables. To start with, if it doesn't already exist, then we need to create a partitioned table on the target server that matches the partitioned table on the original server:

```
CREATE TABLE test.t2 (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50),
    employee_id INT
) ENGINE=InnoDB
PARTITION BY RANGE (employee_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

- Then, using this table as a model, we need to create a placeholder of this table with the same structure that does not use partitioning. This can be done with a

CREATE TABLE... AS SELECT

statement:

```
CREATE TABLE test.t2_placeholder AS
SELECT * FROM test.t2 WHERE NULL;
```

This statement will create a new table called

t2\_placeholder

that has the same schema structure as

t2

, but it does not use partitioning and it contains no rows.

## For Each Partition

From this point forward, the rest of our steps need to happen for each individual partition. For each partition, we need to do the following process:

- First, we need to use

```
ALTER TABLE ... DISCARD TABLESPACE
```

to discard the placeholder table's tablespace:

```
ALTER TABLE test.t2_placeholder DISCARD TABLESPACE;
```

- Then, copy the  
.ibd  
and  
.cfg  
files for the next partition to the relevant directory for the  
t2\_placeholder  
table on the target MariaDB Server:

```
# cp /tmp/backup/t2#p0.cfg /var/lib/mysql/test/t2_placeholder.cfg
# cp /tmp/backup/t2#p0.ibd /var/lib/mysql/test/t2_placeholder.ibd
# chown mysql:mysql /var/lib/mysql/test/t2_placeholder*
```

File-per-table tablespaces can be imported with just the  
.ibd  
file in many cases. If you do not have the tablespace's  
.cfg  
file for whatever reason, then it is usually worth trying to import the tablespace with just the  
.ibd  
file.

- Then, once the files are in the proper directory on the target server, we need to use

```
ALTER TABLE ... IMPORT TABLESPACE
```

to import the new table's tablespace:

```
ALTER TABLE test.t2_placeholder IMPORT TABLESPACE;
```

The placeholder table now contains data from the  
p0  
partition on the source server.

```
SELECT * FROM test.t2_placeholder;
```

employee_id	name
1	Geoff Montee

- Then, it's time to transfer the partition from the placeholder to the target table. This can be done with an

```
ALTER TABLE... EXCHANGE PARTITION
```

statement:

```
ALTER TABLE test.t2 EXCHANGE PARTITION p0 WITH TABLE test.t2_placeholder;
```

The target table now contains the first partition from the source table.

```
SELECT * FROM test.t2;
```

employee_id	name
1	Geoff Montee

- Repeat this procedure for each partition you want to import. For each partition, we need to discard the placeholder table's tablespace, and then import the partitioned table's tablespace into the placeholder table, and then exchange the tablespaces between the placeholder table and the partition of our target table.

When this process is complete for all partitions, the target table will contain the imported data:

```
SELECT * FROM test.t2;
```

employee_id	name
1	Geoff Montee
6	Chris Calendar
11	Kyle Joiner
16	Will Fong

- Then, we can remove the placeholder table from the database:

```
DROP TABLE test.t2_placeholder;
```

## Known Problems with Copying Transportable Tablespaces

### Differing Storage Formats for Temporal Columns

MariaDB 10.1.2 added the

`mysql56_temporal_format`

system variable, which enables a new MySQL 5.6-compatible storage format for the

`TIME`

,

`DATETIME`

and

`TIMESTAMP`

data types.

If a file-per-tablespace file contains columns that use one or more of these temporal data types and if the tablespace file's original table was created with a certain storage format for these columns, then the tablespace file can only be imported into tables that were also created with the same storage format for these columns as the original table. Otherwise, you will see errors like the following:

```
ALTER TABLE dt_test IMPORT TABLESPACE;
ERROR 1808 (HY000): Schema mismatch (Column dt precise type mismatch.)
```

See [MDEV-15225](#) for more information.

See the pages for the

`TIME`

,

`DATETIME`

and

`TIMESTAMP`

data types to determine how to update the storage format for temporal columns in tables that were created before MariaDB 10.1.2 or that were created with

`mysql56_temporal_format=OFF`

## Differing ROW\_FORMAT Values

InnoDB file-per-table tablespaces can use different [row formats](#). A specific row format can be specified when creating a table either by setting the

`ROW_FORMAT`

table option or by the setting the

`innodb_default_row_format`

system variable. See [Setting a Table's Row Format](#) for more information on how to set an InnoDB table's row format.

If a file-per-tablespace file was created with a certain row format, then the tablespace file can only be imported into tables that were created with the same row format as the original table. Otherwise, you will see errors like the following:

```
ALTER TABLE t0 IMPORT TABLESPACE;
ERROR 1808 (HY000): Schema mismatch (Expected FSP_SPACE_FLAGS=0x21, .ibd file contains 0x0.)
```

The error message will be a bit more descriptive in [MariaDB 10.2.17](#) and later:

```
ALTER TABLE t0 IMPORT TABLESPACE;
ERROR 1808 (HY000): Schema mismatch (Table flags don't match, server table has 0x1 and the meta-data file has 0x0; .cfg file use
```

Be sure to check a tablespace's row format before moving it from one server to another. Keep in mind that the default row format can change between major versions of MySQL or MariaDB. See [Checking a Table's Row Format](#) for information on how to check an InnoDB table's row format.

See [MDEV-15049](#) and [MDEV-16851](#) for more information.

## Foreign Key Constraints

DISCARD on a table with foreign key constraints is only possible after disabling [foreign\\_key\\_checks](#):

```
SET SESSION foreign_key_checks=0;
ALTER TABLE t0 DISCARD TABLESPACE;
```

IMPORT on the other hand does not enforce foreign key constraints. So when importing tablespaces, referential integrity can only be guaranteed to import all tables bound by foreign key constraint at the same time, from an EXPORT of those tables taken with the same transactional state.

## Tablespace Encryption

MariaDB supports data-at-rest encryption for the InnoDB storage engine. When enabled, the Server encrypts data before writing it to the tablespace and decrypts reads from the tablespace before returning result-sets. This means that a malicious user attempting to exfiltrate sensitive data won't be able to import the tablespace onto a different server as shown above without the encryption key.

For more information on data encryption, see [Encrypting Data for InnoDB](#).

## See Also

- [Geoff Montee:Importing InnoDB Partitions in MySQL 5.6 and MariaDB 10.0/10.1](#)

### 4.3.2.10.3 InnoDB Temporary Tablespaces

MariaDB starting with 10.2

The use of the temporary tablespaces in InnoDB was introduced in [MariaDB 10.2](#). In earlier versions, temporary tablespaces exist as part of the InnoDB `system` tablespace or were file-per-table depending on the configuration of the `innodb_file_per_table` system variable.

When the user creates a temporary table using the `CREATE TEMPORARY TABLE` statement and the engine is set as InnoDB, MariaDB creates a temporary tablespace file. When the table is not compressed, MariaDB writes to a shared temporary tablespace as defined by the `innodb_temp_data_file_path` system variable. MariaDB does not allow the creation of `ROW_FORMAT=COMPRESSED` temporary tables. All temporary tables will be uncompressed. MariaDB deletes temporary tablespaces when the server shuts down gracefully and is recreated when it starts again. It cannot be placed on a raw device.

Internal temporary tablespaces, (that is, temporary tables that cannot be kept in memory) use either Aria or MyISAM, depending on the `aria_used_for_temp_tables` system variable. You can set the default storage engine for user-created temporary tables using the `default_tmp_storage_engine` system variable.

## Sizing Temporary Tablespaces

In order to size temporary tablespaces, use the `innodb_temp_data_file_path` system variable. This system variable can be specified as a command-line

argument to `mysqld` or it can be specified in a relevant server option group in an [option file](#). For example:

```
[mariadb]
...
innodb_temp_data_file_path=ibtmp1:32M:autoextend
```

This system variable's syntax is the same as the `innodb_data_file_path` system variable. That is, a file name, size and option. By default, it writes a 12MB autoextending file to

ibtmp1  
in the data directory.

To see the current size of the temporary tablespace from MariaDB, query the Information Schema:

```
SELECT FILE_NAME AS "File Name",
       INITIAL_SIZE AS "Initial Size",
       DATA_FREE AS "Free Space",
       TOTAL_EXTENTS * EXTENT_SIZE AS "Total Size",
       MAXIMUM_SIZE AS "Max"
  FROM information_Schema.FILES
 WHERE TABLESPACE_NAME = "innodb_temporary";

+-----+-----+-----+-----+
| File Name | Initial Size | Free Space | Total Size | Max   |
+-----+-----+-----+-----+
| ./ibtmp1 |      12582912 |     621456 |    12592912 | NULL |
+-----+-----+-----+-----+
```

To increase the size of the temporary tablespace, you can add a path to an additional tablespace file to the value of the the `innodb_temp_data_file_path` system variable. Providing additional paths allows you to spread the temporary tablespace between multiple tablespace files. The last file can have the `autoextend` attribute, which ensures that you won't run out of space. For example:

```
[mariadb]
...
innodb_temp_data_file_path=ibtmp1:32M;ibtmp2:32M:autoextend
```

Unlike normal tablespaces, temporary tablespaces are deleted when you stop MariaDB. To shrink temporary tablespaces to their minimum sizes, restart the server.

### 4.3.2.11 InnoDB File Format

#### Contents

- 1. [Setting a Table's File Format](#)
- 2. [Supported File Formats](#)
  - 1. [Antelope](#)
  - 2. [Barracuda](#)
  - 3. [Future Formats](#)
- 3. [Checking a Table's File Format](#).
- 4. [Compatibility](#)
- 5. [See Also](#)

Prior to [MariaDB 10.3](#), the XtraDB/InnoDB storage engine supports two different file formats.

## Setting a Table's File Format

In [MariaDB 10.2](#) and before, the default file format for InnoDB tables can be chosen by setting the `innodb_file_format`.

In [MariaDB 10.2.1](#) and before, the default file format is

Antelope  
. In [MariaDB 10.2.2](#) and later, the default file format is  
Barracuda  
and  
Antelope  
is deprecated.

A table's tablespace is tagged with the lowest InnoDB file format that supports the table's [row format](#). So, even if the Barracuda file format is enabled, tables that use the COMPACT or REDUNDANT

row formats will be tagged with the  
Antelope  
file format in the `information_schema.INNODB_SYS_TABLES` table.

## Supported File Formats

The `InnoDB` storage engine supports two different file formats:

- Antelope
- Barracuda

### Antelope

In `MariaDB 10.2.1` and before, the default file format is

Antelope  
. In `MariaDB 10.2.2` and later, the  
Antelope  
file format is deprecated.

Antelope  
is the original InnoDB file format. It supports the  
COMPACT  
and  
REDUNDANT  
row formats, but not the  
DYNAMIC  
or  
COMPRESSED  
row formats.

### Barracuda

In `MariaDB 10.1` and before, the

Barracuda  
file format is only supported if the `innodb_file_per_table` system variable is set to  
ON  
. In `MariaDB 10.2.2` and later, the default file format is  
Barracuda  
and  
Antelope  
is deprecated.

Barracuda  
is a newer InnoDB file format. It supports the  
COMPACT  
,  
REDUNDANT  
,  
DYNAMIC  
and  
COMPRESSED  
row formats. Tables with large BLOB or TEXT columns in particular could benefit from the dynamic row format.

## Future Formats

InnoDB might use new file formats in the future. Each format will have an identifier from 0 to 25, and a name. The names have already been decided, and are animal names listed in an alphabetical order: Antelope, Barracuda, Cheetah, Dragon, Elk, Fox, Gazelle, Hornet, Impala, Jaguar, Kangaroo, Leopard, Moose, Nautilus, Ocelot, Porpoise, Quail, Rabbit, Shark, Tiger, Urchin, Viper, Whale, Xenops, Yak and Zebra.

## Checking a Table's File Format.

In `MariaDB 10.0` and later, the `information_schema.INNODB_SYS_TABLES` table can be queried to see the file format used by a table.

A table's tablespace is tagged with the lowest InnoDB file format that supports the table's `row format`. So, even if the Barracuda file format is enabled, tables that use the COMPACT or REDUNDANT row formats will be tagged with the Antelope file format in the `information_schema.INNODB_SYS_TABLES` table.

## Compatibility

Each tablespace is tagged with the id of the most recent file format used by one of its tables. All versions of XtraDB/InnoDB can read tables that use an older file format. However, it can not read from more recent formats. For this reason, each time XtraDB/InnoDB opens a table it checks the tablespace's format, and returns an error if a newer format is used.

This check can be skipped via the `innodb_file_format_check` variable. Beware that, if XtraDB/InnoDB tries to repair a table in an unknown format, the table will be corrupted! This happens on restart if `innodb_file_format_check` is disabled and the server crashed, or it was closed with fast shutdown.

To downgrade a table from the Barracuda format to Antelope, the table's

`ROW_FORMAT` can be set to a value supported by Antelope, via an `ALTER TABLE` statement. This recreates the indexes.

The Antelope format can be used to make sure that tables work on MariaDB and MySQL servers which are older than 5.5.

## See Also

- [InnoDB Storage Formats](#)

### 4.3.2.12

#### 4.3.2.12.1 InnoDB Row Formats Overview

##### Contents

1. Default Row Format
2. Setting a Table's Row Format
3. Checking a Table's Row Format
4. Row Formats
  1. REDUNDANT Row Format
  2. COMPACT Row Format
  3. DYNAMIC Row Format
  4. COMPRESSED Row Format
5. Maximum Row Size
6. Known Issues
  1. Upgrading Causes Row Size Too Large Errors

The InnoDB storage engine supports four different row formats:

- REDUNDANT
- COMPACT
- DYNAMIC
- COMPRESSED

In MariaDB 10.1 and before, the latter two row formats are only supported if the InnoDB file format is

Barracuda.  
Therefore, the `innodb_file_format` system variable must be set to  
Barracuda  
to use these row formats in those versions.

In MariaDB 10.1 and before, the latter two row formats are also only supported if the table is in a file per-table tablespace. Therefore, the `innodb_file_per_table` system variable must be set to

ON  
to use these row formats in those versions.

## Default Row Format

MariaDB starting with 10.2.2

In MariaDB 10.2.2 and later, the `innodb_default_row_format` system variable can be used to set the default row format for InnoDB tables. The possible values are:

- redundant
- compact
- dynamic

This system variable's default value is

dynamic  
, which means that the default row format is  
DYNAMIC

This system variable cannot be set to

compressed  
, which means that the default row format cannot be  
COMPRESSED

For example, the following statements would create a table with the

DYNAMIC  
row format:

```
SET SESSION innodb_strict_mode=ON;  
  
SET GLOBAL innodb_default_row_format='dynamic';  
  
CREATE TABLE tab (  
    id int,  
    str varchar(50)  
) ENGINE=InnoDB;
```

#### MariaDB until 10.2.1

In MariaDB 10.2.1 and before, the default row format is

COMPACT

For example, the following statements would create a table with the

COMPACT  
row format:

```
SET SESSION innodb_strict_mode=ON;  
  
CREATE TABLE tab (  
    id int,  
    str varchar(50)  
) ENGINE=InnoDB;
```

## Setting a Table's Row Format

One way to specify an InnoDB table's row format is by setting the `ROW_FORMAT` table option to the relevant row format in a `CREATE TABLE` or `ALTER TABLE` statement. For example:

```
SET SESSION innodb_strict_mode=ON;  
  
SET GLOBAL innodb_file_per_table=ON;  
  
SET GLOBAL innodb_file_format='Barracuda';  
  
CREATE TABLE tab (  
    id int,  
    str varchar(50)  
) ENGINE=InnoDB ROW_FORMAT=DYNAMIC;
```

In MariaDB 10.1 and before, InnoDB can silently ignore and override some row format choices if you do not have the `innodb_file_format` system

variable set to  
Barracuda  
and the `innodb_file_per_table` system variable set to  
ON

## Checking a Table's Row Format

The `SHOW TABLE STATUS` statement can be used to see the row format used by a table. For example:

```
SHOW TABLE STATUS FROM db1 WHERE Name='tab' \G
***** 1. row *****
      Name: tab
      Engine: InnoDB
      Version: 10
      Row_format: Dynamic
      Rows: 0
      Avg_row_length: 0
      Data_length: 16384
      Max_data_length: 0
      Index_length: 0
      Data_free: 0
      Auto_increment: NULL
      Create_time: 2019-04-18 20:24:04
      Update_time: NULL
      Check_time: NULL
      Collation: latin1_swedish_ci
      Checksum: NULL
      Create_options: row_format=DYNAMIC
      Comment:
```

The `information_schema.INNODB_SYS_TABLES` table can also be queried to see the row format used by a table. For example:

```
SELECT * FROM information_schema.INNODB_SYS_TABLES WHERE name='db1/tab' \G
***** 1. row *****
      TABLE_ID: 42
      NAME: db1/tab
      FLAG: 33
      N_COLS: 4
      SPACE: 27
      FILE_FORMAT: Barracuda
      ROW_FORMAT: Dynamic
      ZIP_PAGE_SIZE: 0
      SPACE_TYPE: Single
```

A table's tablespace is tagged with the lowest InnoDB file format that supports the table's row format. So, even if the Barracuda file format is enabled, tables that use the COMPACT or REDUNDANT row formats will be tagged with the Antelope file format in the `information_schema.INNODB_SYS_TABLES` table.

## Row Formats

### REDUNDANT Row Format

The REDUNDANT row format is the original non-compacted row format.

The REDUNDANT row format was the only available row format before MySQL 5.0.3. In that release, this row format was retroactively named the

#### REDUNDANT

row format. In the same release, the

#### COMPACT

row format was introduced as the new default row format.

See [InnoDB REDUNDANT Row Format](#) for more information.

## COMPACT Row Format

MariaDB until [10.2.1](#)

In [MariaDB 10.2.1](#) and before, the default row format is

#### COMPACT

.

The

#### COMPACT

row format is similar to the

#### REDUNDANT

row format, but it stores data in a more compact manner that requires about 20% less storage.

This row format was originally introduced in MySQL 5.0.3.

See [InnoDB COMPACT Row Format](#) for more information.

## DYNAMIC Row Format

MariaDB starting with [10.2.2](#)

In [MariaDB 10.2.2](#) and later, the default row format is

#### DYNAMIC

.

The

#### DYNAMIC

row format is similar to the

#### COMPACT

row format, but tables using the

#### DYNAMIC

row format can store even more data on overflow pages than tables using the

#### COMPACT

row format. This results in more efficient data storage than tables using the

#### COMPACT

row format, especially for tables containing columns using the [VARBINARY](#) , [VARCHAR](#) , [BLOB](#) and [TEXT](#) data types. However, InnoDB tables using the

#### COMPRESSED

row format are more efficient.

See [InnoDB DYNAMIC Row Format](#) for more information.

## COMPRESSED Row Format

From [MariaDB 10.1](#) , an alternative way to compress InnoDB tables is by using [InnoDB Page Compression](#) .

The

#### COMPRESSED

row format is similar to the

#### COMPACT

row format, but tables using the

#### COMPRESSED

row format can store even more data on overflow pages than tables using the

#### COMPACT

row format. This results in more efficient data storage than tables using the

COMPACT

row format, especially for tables containing columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types.

The

COMPRESSED

row format also supports compression of all data and index pages.

See [InnoDB COMPRESSED Row Format](#) for more information.

## Maximum Row Size

Several factors help determine the maximum row size of an InnoDB table.

First, MariaDB enforces a 65,535 byte limit on a table's maximum row size. The total size of a table's `BLOB` and `TEXT` columns do not count towards this limit. Only the pointers for a table's `BLOB` and `TEXT` columns count towards this limit. MariaDB enforces this limit for all storage engines, so this limit also applies to InnoDB tables. Therefore, this limit is the absolute maximum row size for an InnoDB table.

If you try to create a table that exceeds MariaDB's global limit on a table's maximum row size, then you will see an error like this:

```
ERROR 1118 (42000): Row size too large. The maximum row size for the used table type,
not counting BLOBS, is 65535. This includes storage overhead, check the manual. You
have to change some columns to TEXT or BLOBS
```

However, InnoDB also has its own limits on the maximum row size, so an InnoDB table's maximum row size could be smaller than MariaDB's global limit.

Second, the maximum amount of data that an InnoDB table can store in a row's main data page depends on the value of the `innodb_page_size` system variable. At most, the data that a single row can consume on the row's main data page is half of the value of the `innodb_page_size` system variable. With the default value of

16k

, that would mean that a single row can consume at most around 8 KB on the row's main data page. However, the limit on the row's main data page is not the absolute limit on the row's size.

Third, all InnoDB row formats can store certain kinds of data in overflow pages, so the maximum row size of an InnoDB table can be larger than the maximum amount of data that can be stored in the row's main data page.

Some row formats can store more data in overflow pages than others. For example, the

DYNAMIC

and

COMPRESSED

row formats can store the most data in overflow pages. To see how to determine the how the various InnoDB row formats can use overflow pages, see the following sections:

- [InnoDB REDUNDANT Row Format: Overflow Pages with the REDUNDANT Row Format](#)
- [InnoDB COMPACT Row Format: Overflow Pages with the COMPACT Row Format](#)
- [InnoDB DYNAMIC Row Format: Overflow Pages with the DYNAMIC Row Format](#)
- [InnoDB COMPRESSED Row Format: Overflow Pages with the COMPRESSED Row Format](#)

If a table's definition can allow rows that the table's InnoDB row format can't actually store, then InnoDB will raise errors or warnings in certain scenarios.

If the table were using the

REDUNDANT

or

COMPACT

row formats, then the error or warning would be the following:

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB or using ROW_FORMAT=DYNAMIC or ROW_FORMAT=COMPRESSED
may help. In current row format, BLOB prefix of 768 bytes is stored inline.
```

And if the table were using the

DYNAMIC

or

COMPRESSED

row formats, then the error or warning would be the following:

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.
```

These messages are raised in the following cases:

- If `InnoDB strict mode` is **enabled** and if a `DDL` statement is executed that touches the table, such as `CREATE TABLE` or `ALTER TABLE`, then InnoDB will raise an `error` with this message
- If `InnoDB strict mode` is **disabled** and if a `DDL` statement is executed that touches the table, such as `CREATE TABLE`  
or

## ALTER TABLE

- , then InnoDB will raise a **warning** with this message.
- Regardless of whether **InnoDB strict mode** is enabled, if a **DML** statement is executed that attempts to write a row that the table's InnoDB row format can't store, then InnoDB will raise an **error** with this message.

For information on how to solve the problem, see [Troubleshooting Row Size Too Large Errors with InnoDB](#).

## Known Issues

### Upgrading Causes Row Size Too Large Errors

Prior to [MariaDB 10.2.26](#), [MariaDB 10.3.17](#), and [MariaDB 10.4.7](#), MariaDB doesn't properly calculate the row sizes while executing DDL. In these versions, *unsafe* tables can be created, even if **InnoDB strict mode** is enabled. The calculations were fixed by [MDEV-19292](#) in [MariaDB 10.2.26](#), [MariaDB 10.3.17](#), and [MariaDB 10.4.7](#).

As a side effect, some tables that could be created or altered in previous versions may get rejected with the following error in these releases and any later releases.

```
ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to  
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.
```

And users could also see the following message as an error or warning in the [error log](#):

```
[Warning] InnoDB: Cannot add field col in table db1.tab because after adding it, the row size is 8478 which is greater than maxi
```

InnoDB used the wrong calculations to determine row sizes for quite a long time, so a lot of users may unknowingly have *unsafe* tables that the InnoDB row format can't actually store.

InnoDB does not currently have an easy way to check which existing tables have this problem. See [MDEV-20400](#) for more information.

For information on how to solve the problem, see [Troubleshooting Row Size Too Large Errors with InnoDB](#).

### 4.3.2.12.2 InnoDB REDUNDANT Row Format

#### Contents

1. [Using the REDUNDANT Row Format](#)
2. [Index Prefixes with the REDUNDANT Row Format](#)
3. [Overflow Pages with the REDUNDANT Row Format](#)

The

REDUNDANT  
row format is the original non-compacted row format.

The

REDUNDANT  
row format was the only available row format before MySQL 5.0.3. In that release, this row format was retroactively named the  
REDUNDANT  
row format. In the same release, the  
COMPACT  
row format was introduced as the new default row format.

## Using the

### REDUNDANT Row Format

The easiest way to create an InnoDB table that uses the

REDUNDANT  
row format is by setting the **ROW\_FORMAT** table option to  
REDUNDANT  
in a **CREATE TABLE** or **ALTER TABLE** statement.

It is recommended to set the **innodb\_strict\_mode** system variable to  
ON  
when using this format.

The

REDUNDANT

row format is supported by both the Antelope and the Barracuda file formats, so tables with this row format can be created regardless of the value of the `innodb_file_format` system variable.

For example:

```
SET SESSION innodb_strict_mode=ON;

CREATE TABLE tab (
    id int,
    str varchar(50)
) ENGINE=InnoDB ROW_FORMAT=REDUNDANT;
```

## Index Prefixes with the REDUNDANT Row Format

The REDUNDANT row format supports index prefixes up to 767 bytes.

## Overflow Pages with the REDUNDANT Row Format

All InnoDB row formats can store certain kinds of data in overflow pages. This allows for the maximum row size of an InnoDB table to be larger than the maximum amount of data that can be stored in the row's main data page. See [Maximum Row Size](#) for more information about the other factors that can contribute to the maximum row size for InnoDB tables.

In the REDUNDANT row format variable-length columns, such as columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types, can be partially stored in overflow pages.

InnoDB only considers using overflow pages if the table's row size is greater than half of `innodb_page_size`. If the row size is greater than this, then InnoDB chooses variable-length columns to be stored on overflow pages until the row size is less than half of `innodb_page_size`.

For `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` columns, only values longer than 767 bytes are considered for storage on overflow pages. Bytes that are stored to track a value's length do not count towards this limit. This limit is only based on the length of the actual column's data.

Fixed-length columns greater than 767 bytes are encoded as variable-length columns, so they can also be stored in overflow pages if the table's row size is greater than half of `innodb_page_size`. Even though a column using the `CHAR` data type can hold at most 255 characters, a `CHAR` column can still exceed 767 bytes in some cases. For example, a

```
char(255)  
column can exceed 767 bytes if the character set is  
utf8mb4
```

If a column is chosen to be stored on overflow pages, then the first 767 bytes of the column's value and a 20-byte pointer to the column's first overflow page are stored on the main page. Each overflow page is the size of `[innodb-system-variables#innodb_page_size] * innodb_page_size]`. If a column is too large to be stored on a single overflow page, then it is stored on multiple overflow pages. Each overflow page contains part of the data and a 20-byte pointer to the next overflow page, if a next page exists.

### 4.3.2.12.3 InnoDB COMPACT Row Format

MariaDB until 10.2.1

In MariaDB 10.2.1 and before, the default row format is

COMPACT

## Contents

1. Using the COMPACT Row Format
2. Index Prefixes with the COMPACT Row Format
3. Overflow Pages with the COMPACT Row Format

The

COMPACT  
row format is similar to the  
REDUNDANT  
row format, but it stores data in a more compact manner that requires about 20% less storage.

## Using the COMPACT Row Format

MariaDB starting with 10.2.2

In MariaDB 10.2.2 and later, the easiest way to create an InnoDB table that uses the

COMPACT  
row format is by setting the `ROW_FORMAT` table option to to  
COMPACT  
in a `CREATE TABLE` or `ALTER TABLE` statement.

It is recommended to set the `innodb_strict_mode` system variable to  
ON  
when using this row format.

The

COMPACT  
row format is supported by both the  
Antelope  
and the  
Barracuda  
`file formats`, so tables with this row format can be created regardless of the value of the `innodb_file_format` system variable.

For example:

```
SET SESSION innodb_strict_mode=ON;

CREATE TABLE tab (
    id int,
    str varchar(50)
) ENGINE=InnoDB ROW_FORMAT=COMPACT;
```

MariaDB until 10.2.1

In MariaDB 10.2.1 and before, the default row format is

COMPACT  
. Therefore, in these versions, the easiest way to create an InnoDB table that uses the  
COMPACT  
row format is by **not** setting the `ROW_FORMAT` table option at all in the `CREATE TABLE` or `ALTER TABLE` statement.

It is recommended to set the `innodb_strict_mode` system variable to  
ON  
when using this row format.

The

COMPACT  
row format is supported by both the  
Antelope  
and the  
Barracuda  
`file formats`, so tables with this row format can be created regardless of the value of the `innodb_file_format` system variable.

For example:

```
SET SESSION innodb_strict_mode=ON;

CREATE TABLE tab (
    id int,
    str varchar(50)
) ENGINE=InnoDB;
```

## Index Prefixes with the COMPACT Row Format

The

COMPACT

row format supports index prefixes up to 767 bytes.

## Overflow Pages with the COMPACT Row Format

All InnoDB row formats can store certain kinds of data in overflow pages. This allows for the maximum row size of an InnoDB table to be larger than the maximum amount of data that can be stored in the row's main data page. See [Maximum Row Size](#) for more information about the other factors that can contribute to the maximum row size for InnoDB tables.

In the

COMPACT

row format variable-length columns, such as columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types, can be partially stored in overflow pages.

InnoDB only considers using overflow pages if the table's row size is greater than half of `innodb_page_size`. If the row size is greater than this, then InnoDB chooses variable-length columns to be stored on overflow pages until the row size is less than half of `innodb_page_size`.

For `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` columns, only values longer than 767 bytes are considered for storage on overflow pages. Bytes that are stored to track a value's length do not count towards this limit. This limit is only based on the length of the actual column's data.

Fixed-length columns greater than 767 bytes are encoded as variable-length columns, so they can also be stored in overflow pages if the table's row size is greater than half of `innodb_page_size`. Even though a column using the `CHAR` data type can hold at most 255 characters, a `CHAR` column can still exceed 767 bytes in some cases. For example, a

`char(255)`

column can exceed 767 bytes if the `character set` is

`utf8mb4`

If a column is chosen to be stored on overflow pages, then the first 767 bytes of the column's value and a 20-byte pointer to the column's first overflow page are stored on the main page. Each overflow page is the size of `innodb_page_size`. If a column is too large to be stored on a single overflow page, then it is stored on multiple overflow pages. Each overflow page contains part of the data and a 20-byte pointer to the next overflow page, if a next page exists.

### 4.3.2.12.4 InnoDB DYNAMIC Row Format

MariaDB starting with 10.2.2

In MariaDB 10.2.2 and later, the default row format is

DYNAMIC

#### Contents

1. [Using the DYNAMIC Row Format](#)
2. [Index Prefixes with the DYNAMIC Row Format](#)
3. [Overflow Pages with the DYNAMIC Row Format](#)

The

DYNAMIC

row format is similar to the

COMPACT

row format, but tables using the DYNAMIC row format can store even more data on overflow pages than tables using the COMPACT row format. This results in more efficient data storage than tables using the COMPACT row format, especially for tables containing columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types. However, InnoDB tables using the COMPRESSED row format are more efficient.

The

DYNAMIC row format was originally introduced in [MariaDB 5.5](#).

## Using the DYNAMIC Row Format

MariaDB starting with [10.2.2](#)

In [MariaDB 10.2.2](#) and later, the default row format is

DYNAMIC

, as long as the `innodb_default_row_format` system variable has not been modified. Therefore, in these versions, the easiest way to create an InnoDB table that uses the

DYNAMIC

row format is by **not** setting the `ROW_FORMAT` table option at all in a `CREATE TABLE` or `ALTER TABLE` statement.

It is recommended to set the `innodb_strict_mode` system variable to

ON

when using this row format.

For example:

```
SET SESSION innodb_strict_mode=ON;

SET GLOBAL innodb_default_row_format='dynamic';

CREATE TABLE tab (
    id int,
    str varchar(50)
) ENGINE=InnoDB;
```

MariaDB until [10.2.1](#)

In [MariaDB 10.2.1](#) and before, the easiest way to create an InnoDB table that uses the

DYNAMIC

row format is by setting the `ROW_FORMAT` table option to to

DYNAMIC

in a `CREATE TABLE` or `ALTER TABLE` statement.

It is recommended to set the `innodb_strict_mode` system variable to

ON

when using this row format.

The

DYNAMIC

row format is only supported by the

Barracuda

`file format`. As a side effect, in [MariaDB 10.1](#) and before, the

DYNAMIC

row format is only supported if the `InnoDB file format` is

Barracuda

. Therefore, the `innodb_file_format` system variable must be set to

Barracuda

to use these row formats in those versions.

In [MariaDB 10.1](#) and before, the

DYNAMIC

row format is also only supported if the table is in a `file per-table` tablespace. Therefore, the `innodb_file_per_table` system variable must be set to

ON

to use this row format in those versions.

For example:

```

SET SESSION innodb_strict_mode=ON;

SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

CREATE TABLE tab (
    id int,
    str varchar(50)
) ENGINE=InnoDB ROW_FORMAT=DYNAMIC;

```

## Index Prefixes with the DYNAMIC Row Format

The

DYNAMIC

row format supports index prefixes up to 3072 bytes. In MariaDB 10.2 and before, the `innodb_large_prefix` system variable is used to configure the maximum index prefix length. In these versions, if `innodb_large_prefix` is set to

ON

, then the maximum prefix length is 3072 bytes, and if it is set to

OFF

, then the maximum prefix length is 767 bytes.

## Overflow Pages with the DYNAMIC Row Format

All InnoDB row formats can store certain kinds of data in overflow pages. This allows for the maximum row size of an InnoDB table to be larger than the maximum amount of data that can be stored in the row's main data page. See [Maximum Row Size](#) for more information about the other factors that can contribute to the maximum row size for InnoDB tables.

In the

DYNAMIC

row format variable-length columns, such as columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types, can be completely stored in overflow pages.

InnoDB only considers using overflow pages if the table's row size is greater than half of `innodb_page_size`. If the row size is greater than this, then InnoDB chooses variable-length columns to be stored on overflow pages until the row size is less than half of `innodb_page_size`.

For `BLOB` and `TEXT` columns, only values longer than 40 bytes are considered for storage on overflow pages. For `VARBINARY` and `VARCHAR` columns, only values longer than 255 bytes are considered for storage on overflow pages. Bytes that are stored to track a value's length do not count towards these limits. These limits are only based on the length of the actual column's data.

These limits differ from the limits for the

COMPACT

row format, where the limit is 767 bytes for all types.

Fixed-length columns greater than 767 bytes are encoded as variable-length columns, so they can also be stored in overflow pages if the table's row size is greater than half of `innodb_page_size`. Even though a column using the `CHAR` data type can hold at most 255 characters, a `CHAR` column can still exceed 767 bytes in some cases. For example, a

`char(255)`

column can exceed 767 bytes if the `character set` is

`utf8mb4`

If a column is chosen to be stored on overflow pages, then the entire value of the column is stored on overflow pages, and only a 20-byte pointer to the column's first overflow page is stored on the main page. Each overflow page is the size of `innodb_page_size`. If a column is too large to be stored on a single overflow page, then it is stored on multiple overflow pages. Each overflow page contains part of the data and a 20-byte pointer to the next overflow page, if a next page exists.

This behavior differs from the behavior of the

COMPACT

row format, which always stores the column prefix on the main page. This allows tables using the

DYNAMIC

row format to contain a high number of columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types.

## 4.3.2.12.5 InnoDB COMPRESSED Row Format

## Contents

1. Using the COMPRESSED Row Format
2. Compression with the COMPRESSED Row Format
3. Monitoring Performance of the COMPRESSED Row Format
4. Index Prefixes with the COMPRESSED Row Format
5. Overflow Pages with the COMPRESSED Row Format
6. Read-Only
7. See Also

In MariaDB 10.1 and later, an alternative (and usually superior) way to compress InnoDB tables is by using [InnoDB Page Compression](#). See [Comparison with the COMPRESSED Row Format](#).

The

COMPRESSED row format is similar to the COMPACT row format, but tables using the COMPRESSED row format can store even more data on overflow pages than tables using the COMPACT row format. This results in more efficient data storage than tables using the COMPACT row format, especially for tables containing columns using the [VARBINARY](#), [VARCHAR](#), [BLOB](#) and [TEXT](#) data types.

The

COMPRESSED row format also supports compression of all data and index pages.

## Using the COMPRESSED Row Format

An InnoDB table that uses the

COMPRESSED row format can be created by setting the [ROW\\_FORMAT](#) table option to COMPRESSED

and by setting the [KEY\\_BLOCK\\_SIZE](#) table option to one of the following values in a [CREATE TABLE](#) or [ALTER TABLE](#) statement, where the units are in

KB

:

•

1

•

2

•

4

•

8

•

16

16k

is the default value of the [innodb\\_page\\_size](#) system variable, so using

16

will usually result in minimal compression unless one of the following is true:

- The table has many columns that can be stored in overflow pages, such as columns that use the [VARBINARY](#), [VARCHAR](#), [BLOB](#) and [TEXT](#) data types.

- The server is using a non-default `innodb_page_size` value that is greater than 16k

In MariaDB 10.1 and later, the value of the `innodb_page_size` system variable can be set to

32k  
and  
64k

. This is especially useful because the larger page size permits more columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types.

Regardless, even when the value of the `innodb_page_size` system variable is set to some value higher than

16k

,

16

is still the maximum value for the `KEY_BLOCK_SIZE` table option for InnoDB tables using the `COMPRESSED` row format.

The

`COMPRESSED` row format cannot be set as the default row format with the `innodb_default_row_format` system variable.

The

`COMPRESSED` row format is only supported by the `Barracuda` file format . As a side effect, in MariaDB 10.1 and before, the `COMPRESSED` row format is only supported if the `InnoDB file format` is `Barracuda` . Therefore, the `innodb_file_format` system variable must be set to `Barracuda` to use these row formats in those versions.

In MariaDB 10.1 and before, the

`COMPRESSED` row format is also only supported if the table is in a `file per-table` tablespace. Therefore, the `innodb_file_per_table` system variable must be set to `ON` to use this row format in those versions.

It is also recommended to set the `innodb_strict_mode` system variable to

`ON` when using this row format.

InnoDB automatically uses the

`COMPRESSED` row format for a table if the `KEY_BLOCK_SIZE` table option is set to some value in a `CREATE TABLE` or `ALTER TABLE` statement. For example:

```
SET SESSION innodb_strict_mode=ON;
SET GLOBAL innodb_file_per_table=ON;
SET GLOBAL innodb_file_format='Barracuda';

CREATE TABLE tab (
    id int,
    str varchar(50)
) ENGINE=InnoDB KEY_BLOCK_SIZE=4;
```

If the `KEY_BLOCK_SIZE` table option is **not** set to some value, but the `ROW_FORMAT` table option is set to

`COMPRESSED` in a `CREATE TABLE` or `ALTER TABLE` statement, then InnoDB uses a default value of 8 for the `KEY_BLOCK_SIZE` table option. For example:

```

SET SESSION innodb_strict_mode=ON;

SET GLOBAL innodb_file_per_table=ON;

SET GLOBAL innodb_file_format='Barracuda';

CREATE TABLE tab (
    id int,
    str varchar(50)
) ENGINE=InnoDB ROW_FORMAT=COMPRESSED;

```

## Compression with the COMPRESSED Row Format

The

COMPRESSED  
row format supports compression of all data and index pages.

To avoid compressing and uncompressing pages too many times, InnoDB tries to keep both compressed and uncompressed pages in the [buffer pool](#) when there is enough room. This results in a bigger cache. When there is not enough room, an adaptive LRU algorithm is used to decide whether compressed or uncompressed pages should be evicted from the buffer: for CPU-bound workloads, the compressed pages are evicted first; for I/O-bound workloads, the uncompressed pages are evicted first. Of course, when necessary, both the compressed and uncompressed version of the same data can be evicted from the buffer.

Each compressed page has an uncompressed *modification log*, stored within the page itself. InnoDB writes small changes into it. When the space in the modification log runs out, the page is uncompressed, changes are applied, and the page is recompressed again. This is done to avoid some unnecessary decompression and compression operations.

Sometimes a *compression failure* might happen, because the data has grown too much to fit the page. When this happens, the page (and the index node) is split into two different pages. This process can be repeated recursively until the data fit the pages. This can be CPU-consuming on some busy servers which perform many write operations.

Before writing a compressed page into a data file, InnoDB writes it into the [redo log](#). This ensures that the [redo log](#) can always be used to recover tables after a crash, even if the compression library is updated and some incompatibilities are introduced. But this also means that the [redo log](#) will grow faster and might need more space, or the frequency of checkpoints might need to increase.

## Monitoring Performance of the COMPRESSED Row Format

The following

[INFORMATION\\_SCHEMA](#)  
tables can be used to monitor the performances of InnoDB compressed tables:

- [INNODB\\_CMP](#) and [INNODB\\_CMP\\_RESET](#)
- [INNODB\\_CMP\\_PER\\_INDEX](#) and [INNODB\\_CMP\\_PER\\_INDEX\\_RESET](#)
- [INNODB\\_CMPPMEM](#) and [INNODB\\_CMPPMEM\\_RESET](#)

## Index Prefixes with the COMPRESSED Row Format

The

COMPRESSED  
row format supports index prefixes up to 3072 bytes. In [MariaDB 10.2](#) and before, the [innodb\\_large\\_prefix](#) system variable is used to configure the maximum index prefix length. In these versions, if [innodb\\_large\\_prefix](#) is set to

ON  
, then the maximum prefix length is 3072 bytes, and if it is set to  
OFF  
, then the maximum prefix length is 767 bytes.

## Overflow Pages with the COMPRESSED Row Format

All InnoDB row formats can store certain kinds of data in overflow pages. This allows for the maximum row size of an InnoDB table to be larger than the maximum amount of data that can be stored in the row's main data page. See [Maximum Row Size](#) for more information about the other factors that can contribute to the maximum row size for InnoDB tables.

In the

COMPRESSED

row format variable-length columns, such as columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types, can be completely stored in overflow pages.

InnoDB only considers using overflow pages if the table's row size is greater than half of `innodb_page_size`. If the row size is greater than this, then InnoDB chooses variable-length columns to be stored on overflow pages until the row size is less than half of `innodb_page_size`.

For `BLOB` and `TEXT` columns, only values longer than 40 bytes are considered for storage on overflow pages. For `VARBINARY` and `VARCHAR` columns, only values longer than 255 bytes are considered for storage on overflow pages. Bytes that are stored to track a value's length do not count towards these limits. These limits are only based on the length of the actual column's data.

These limits differ from the limits for the

COMPACT

row format, where the limit is 767 bytes for all types.

Fixed-length columns greater than 767 bytes are encoded as variable-length columns, so they can also be stored in overflow pages if the table's row size is greater than half of `innodb_page_size`. Even though a column using the `CHAR` data type can hold at most 255 characters, a `CHAR` column can still exceed 767 bytes in some cases. For example, a

`char(255)`

column can exceed 767 bytes if the `character set` is

`utf8mb4`

If a column is chosen to be stored on overflow pages, then the entire value of the column is stored on overflow pages, and only a 20-byte pointer to the column's first overflow page is stored on the main page. Each overflow page is the size of `innodb_page_size`. If a column is too large to be stored on a single overflow page, then it is stored on multiple overflow pages. Each overflow page contains part of the data and a 20-byte pointer to the next overflow page, if a next page exists.

This behavior differs from the behavior of the

COMPACT

row format, which always stores the column prefix on the main page. This allows tables using the

COMPRESSED

row format to contain a high number of columns using the `VARBINARY`, `VARCHAR`, `BLOB` and `TEXT` data types.

## Read-Only

MariaDB starting with [10.6](#)

From [MariaDB 10.6.0](#) until [MariaDB 10.6.5](#), tables that are of the

COMPRESSED

row format are read-only by default. This was intended to be the first step towards removing write support and deprecating the feature.

This plan has been scrapped, and from [MariaDB 10.6.6](#),

COMPRESSED

tables are no longer read-only by default.

From [MariaDB 10.6.0](#) to [MariaDB 10.6.5](#), set the `innodb_read_only_compressed` variable to

OFF

to make the tables writable.

## See Also

- [InnoDB Page Compression](#)
- [Storage-Engine Independent Column Compression](#)

## 4.3.2.12.6 Troubleshooting Row Size Too Large Errors with InnoDB

## 4.3.2.13 InnoDB Strict Mode

## Contents

1. Configuring InnoDB Strict Mode
2. InnoDB Strict Mode Errors
  1. Wrong Create Options
  2. COMPRESSED Row Format
  3. Row Size Too Large

InnoDB strict mode is similar to [SQL strict mode](#). When it is enabled, certain InnoDB warnings become errors instead.

## Configuring InnoDB Strict Mode

MariaDB starting with [10.2.2](#)

In [MariaDB 10.2.2](#) and later, InnoDB strict mode is enabled by default.

InnoDB strict mode can be enabled or disabled by configuring the

`innodb_strict_mode`

server system variable.

Its global value can be changed dynamically with

`SET GLOBAL`

. For example:

```
SET GLOBAL innodb_strict_mode=ON;
```

Its value for the current session can also be changed dynamically with

`SET SESSION`

. For example:

```
SET SESSION innodb_strict_mode=ON;
```

It can also be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
innodb_strict_mode=ON
```

## InnoDB Strict Mode Errors

### Wrong Create Options

If InnoDB strict mode is enabled, and if a DDL statement is executed and invalid or conflicting [table options](#) are specified, then an error is raised. The error will only be a generic error that says the following:

```
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
```

However, more details about the error can be found by executing

`SHOW WARNINGS`

For example, the error is raised in the following cases:

- The

`KEY_BLOCK_SIZE`

table option is set to a non-zero value, but the

`ROW_FORMAT`

table option is set to some row format other than the

[COMPRESSED](#)

row format. For example:

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
KEY_BLOCK_SIZE=4
ROW_FORMAT=DYNAMIC;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1478 | InnoDB: cannot specify ROW_FORMAT = DYNAMIC with KEY_BLOCK_SIZE. |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.000 sec)
```

- The

[KEY\\_BLOCK\\_SIZE](#)

table option is set to a non-zero value, but the configured value is larger than either

16

or the value of the

[innodb\\_page\\_size](#)

system variable, whichever is smaller.

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
KEY_BLOCK_SIZE=16;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1478 | InnoDB: KEY_BLOCK_SIZE=16 cannot be larger than 8. |
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.000 sec)
```

- The

[KEY\\_BLOCK\\_SIZE](#)

table option is set to a non-zero value, but the

[innodb\\_file\\_per\\_table](#)

system variable is not set to

ON

```

SET GLOBAL innodb_file_per_table=OFF;
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
KEY_BLOCK_SIZE=4;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Warning | 1478 | InnoDB: KEY_BLOCK_SIZE requires innodb_file_per_table.
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB
+-----+-----+
3 rows in set (0.000 sec)

```

- The

#### KEY\_BLOCK\_SIZE

table option is set to a non-zero value, but it is not set to one of the supported values: [1, 2, 4, 8, 16].

```

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
KEY_BLOCK_SIZE=5;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Warning | 1478 | InnoDB: invalid KEY_BLOCK_SIZE = 5. Valid values are [1, 2, 4, 8, 16]
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB
+-----+-----+
3 rows in set (0.000 sec)

```

- The

#### ROW\_FORMAT

table option is set to the

#### COMPRESSED

row format, but the

#### innodb\_file\_per\_table

system variable is not set to

ON

```

SET GLOBAL innodb_file_per_table=OFF;
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
ROW_FORMAT=COMPRESSED;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1478 | InnoDB: ROW_FORMAT=COMPRESSED requires innodb_file_per_table. |
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.000 sec)

```

- The

`ROW_FORMAT`

table option is set to a value, but it is not set to one of the values supported by InnoDB:

`REDUNDANT`

,

`COMPACT`

,

`DYNAMIC`

, and

`COMPRESSED`

.

```

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
ROW_FORMAT=PAGE;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1478 | InnoDB: invalid ROW_FORMAT specifier. |
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.000 sec)

```

- Either the

`KEY_BLOCK_SIZE`

table option is set to a non-zero value or the

`ROW_FORMAT`

table option is set to the

`COMPRESSED`

row format, but the

`innodb_page_size`

system variable is set to a value greater than  
16k

```
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
ROW_FORMAT=COMPRESSED;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
```

SHOW WARNINGS;

Level	Code	Message
Warning	1478	InnoDB: Cannot create a COMPRESSED table when innodb_page_size > 16k.
Error	1005	Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
Warning	1030	Got error 140 "Wrong create options" from storage engine InnoDB

3 rows in set (0.00 sec)

- The

`DATA DIRECTORY`

table option is set, but the

`innodb_file_per_table`

system variable is not set to  
ON

```
SET GLOBAL innodb_file_per_table=OFF;
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
DATA DIRECTORY='/mariadb';
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
```

SHOW WARNINGS;

Level	Code	Message
Warning	1478	DATA DIRECTORY requires innodb_file_per_table.
Error	1005	Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
Warning	1030	Got error 140 "Wrong create options" from storage engine InnoDB

3 rows in set (0.000 sec)

- The

`DATA DIRECTORY`

table option is set, but the table is a `temporary table`.

```

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TEMPORARY TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
DATA DIRECTORY='/mariadb';
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Warning | 1478 | InnoDB: DATA DIRECTORY cannot be used for TEMPORARY tables.
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB
+-----+-----+
3 rows in set (0.000 sec)

```

- The

INDEX DIRECTORY

table option is set.

```

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
INDEX DIRECTORY='/mariadb';
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Warning | 1478 | InnoDB: INDEX DIRECTORY is not supported
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB
+-----+-----+
3 rows in set (0.000 sec)

```

- The

PAGE\_COMPRESSED

table option is set to

1

, so InnoDB page compression is enabled, but the

ROW\_FORMAT

table option is set to some row format other than the

COMPACT

or

DYNAMIC

row formats.

```

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
PAGE_COMPRESSED=1
ROW_FORMAT=COMPRESSED;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 140 | InnoDB: PAGE_COMPRESSED table can't have ROW_TYPE=COMPRESSED |
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.000 sec)

```

- The

PAGE\_COMPRESSED

table option is set to

1

, so InnoDB page compression is enabled, but the

innodb\_file\_per\_table

system variable is not set to

ON

.

```

SET GLOBAL innodb_file_per_table=OFF;
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
PAGE_COMPRESSED=1;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 140 | InnoDB: PAGE_COMPRESSED requires innodb_file_per_table. |
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.000 sec)

```

- The

PAGE\_COMPRESSED

table option is set to

1

, so InnoDB page compression is enabled, but the

KEY\_BLOCK\_SIZE

table option is also specified.

```

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
PAGE_COMPRESSED=1
KEY_BLOCK_SIZE=4;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 140 | InnoDB: PAGE_COMPRESSED table can't have key_block_size |
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.000 sec)

```

- The

`PAGE_COMPRESSION_LEVEL`

table option is set, but the

`PAGE_COMPRESSED`

table option is set to

0

, so `InnoDB page compression` is disabled.

```

SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
PAGE_COMPRESSED=0
PAGE_COMPRESSION_LEVEL=9;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 140 | InnoDB: PAGE_COMPRESSION_LEVEL requires PAGE_COMPRESSED |
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.000 sec)

```

## MariaDB until 10.2

In `MariaDB 10.2` and before, the error is raised in the following additional cases:

- The

`KEY_BLOCK_SIZE`

table option is set to a non-zero value, but the

`innodb_file_format`

system variable is not set to

`Barracuda`

```

SET GLOBAL innodb_file_format='Antelope';
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
KEY_BLOCK_SIZE=4;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1478 | InnoDB: KEY_BLOCK_SIZE requires innodb_file_format > Antelope. |
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.00 sec)

```

- The

[ROW\\_FORMAT](#)

table option is set to either the

[COMPRESSED](#)

or the

[DYNAMIC](#)

row format, but the

[innodb\\_file\\_format](#)

system variable is not set to

Barracuda

```

SET GLOBAL innodb_file_format='Antelope';
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
ROW_FORMAT=COMPRESSED;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1478 | InnoDB: ROW_FORMAT=COMPRESSED requires innodb_file_format > Antelope. |
| Error | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options") |
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB |
+-----+-----+
3 rows in set (0.00 sec)

```

- The

[PAGE\\_COMPRESSED](#)

table option is set to

1

, so [InnoDB page compression](#) is enabled, but the

[innodb\\_file\\_format](#)

system variable is not set to

Barracuda

```

SET GLOBAL innodb_file_format='Antelope';
SET SESSION innodb_strict_mode=ON;

CREATE OR REPLACE TABLE tab (
    id int PRIMARY KEY,
    str varchar(50)
)
PAGE_COMPRESSED=1;
SHOW WARNINGS;
ERROR 1005 (HY000): Can't create table `db1`.`tab` (errno: 140 "Wrong create options")

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Warning | 140 | InnoDB: PAGE_COMPRESSED requires innodb_file_format > Antelope.
| Error   | 1005 | Can't create table `db1`.`tab` (errno: 140 "Wrong create options")
| Warning | 1030 | Got error 140 "Wrong create options" from storage engine InnoDB
+-----+-----+
3 rows in set (0.00 sec)

```

## COMPRESSED Row Format

If InnoDB strict mode is enabled, and if a table uses the

`COMPRESSED`

row format, and if the table's

`KEY_BLOCK_SIZE`

is too small to contain a row, then an error is returned by the statement.

## Row Size Too Large

If InnoDB strict mode is enabled, and if a table exceeds its row format's `maximum row size`, then InnoDB will return an error.

```

ERROR 1118 (42000): Row size too large (> 8126). Changing some columns to
TEXT or BLOB may help. In current row format, BLOB prefix of 0 bytes is stored inline.

```

See [Troubleshooting Row Size Too Large Errors with InnoDB](#) for more information.

### 4.3.2.14 InnoDB Redo Log

#### Contents

1. [Overview](#)
2. [Flushing Effects on Performance and Consistency](#)
  1. [Binary Log Group Commit and Redo Log Flushing](#)
3. [Redo Log Group Capacity](#)
  1. [Changing the Redo Log Group Capacity](#)
4. [Log Sequence Number \(LSN\)](#)
5. [Checkpoints](#)
  1. [Determining the Checkpoint Age](#)
    1. [Determining the Checkpoint Age in InnoDB](#)
    2. [Determining the Checkpoint Age in XtraDB](#)
  6. [Determining the Redo Log Occupancy](#)

## Overview

The redo log is used by [InnoDB](#) during crash recovery. The redo log files have names like

`ib_logfileN`  
, where  
N

is an integer. From [MariaDB 10.5](#), there is only one redo log, so the file will always be named `ib_logfile0`.  
If the `innodb_log_group_home_dir` system variable is configured, then the redo log files will be created in that directory. Otherwise, they will be created in the directory defined by the `datadir` system variable.

## Flushing Effects on Performance and Consistency

The `innodb_flush_log_at_trx_commit` system variable determines how often the transactions are flushed to the redo log, and it is important to achieve a good balance between speed and reliability.

## Binary Log Group Commit and Redo Log Flushing

In [MariaDB 10.0](#) and above, when both `innodb_flush_log_at_trx_commit=1` (the default) is set and the [binary log](#) is enabled, there is now one less sync to disk inside InnoDB during commit (2 syncs shared between a group of transactions instead of 3). See [Binary Log Group Commit and InnoDB Flushing Performance](#) for more information.

## Redo Log Group Capacity

The redo log group capacity is the total combined size of all InnoDB redo logs. The relevant factors are:

- From [MariaDB 10.5](#), there is 1 redo log. For [MariaDB 10.4](#) and before, the number of redo log files is configured by the `innodb_log_files_in_group` system variable.
- The size of each redo log file is configured by the `innodb_log_file_size` system variable.

Therefore, redo log group capacity is determined by the following calculation:

```
innodb_log_group_capacity  
=  
  
  innodb_log_file_size  
  
  *  
  
  innodb_log_files_in_group
```

For example, if `innodb_log_file_size` is set to

2G

and `innodb_log_files_in_group` is set to

2

, then we would have the following:

- ```
  innodb_log_group_capacity  
  =  
  
    innodb_log_file_size  
  
    *  
  
    innodb_log_files_in_group
```
- =  
2G  
\*  
2
- =  
4G

## Changing the Redo Log Group Capacity

The number (until [MariaDB 10.4](#) only - from [MariaDB 10.5](#) there is only 1 redo log) or size of redo log files can be changed with the following process:

- Stop the server.
- To change the log file size, configure `innodb_log_file_size`. To increase the number of log files (until [MariaDB 10.4](#) only), configure `innodb_log_files_in_group`.
- Start the server.

# Log Sequence Number (LSN)

Records within the InnoDB redo log are identified via a log sequence number (LSN).

## Checkpoints

When InnoDB performs a checkpoint, it writes the LSN of the oldest dirty page in the [InnoDB buffer pool](#) to the InnoDB redo log. If a page is the oldest dirty page in the [InnoDB buffer pool](#), then that means that all pages with lower LSNs have been flushed to the physical InnoDB tablespace files. If the server were to crash, then InnoDB would perform crash recovery by only applying log records with LSNs that are greater than or equal to the LSN of the oldest dirty page written in the last checkpoint.

Checkpoints are one of the tasks performed by the InnoDB master background thread. This thread schedules checkpoints 7 seconds apart when the server is very active, but checkpoints can happen more frequently when the server is less active.

Dirty pages are not actually flushed from the buffer pool to the physical InnoDB tablespace files during a checkpoint. That process happens asynchronously on a continuous basis by InnoDB's write I/O background threads configured by the [innodb\\_write\\_io\\_threads](#) system variable. If you want to make this process more aggressive, then you can decrease the value of the [innodb\\_max\\_dirty\\_pages\\_pct](#) system variable. You may also need to better tune InnoDB's I/O capacity on your system by setting the [innodb\\_io\\_capacity](#) system variable.

## Determining the Checkpoint Age

The checkpoint age is the amount of data written to the InnoDB redo log since the last checkpoint.

### Determining the Checkpoint Age in InnoDB

MariaDB starting with 10.2

In [MariaDB 10.2](#) and later, MariaDB uses InnoDB. In those versions, the checkpoint age can be determined by the process shown below.

To determine the InnoDB checkpoint age, do the following:

- Query [SHOW ENGINE INNODB STATUS](#).
- Find the

LOG  
section. For example:

```
---  
LOG  
---  
Log sequence number 252794398789379  
Log flushed up to 252794398789379  
Pages flushed up to 252792767756840  
Last checkpoint at 252792767756840  
0 pending log flushes, 0 pending chkp writes  
23930412 log i/o's done, 2.03 log i/o's/second
```

- Perform the following calculation:

```
innodb_checkpoint_age  
=  
Log sequence number  
-  
Last checkpoint at
```

In the example above, that would be:

- innodb\_checkpoint\_age  
=  
Log sequence number  
-  
Last checkpoint at
- = 252794398789379 - 252792767756840
- = 1631032539 bytes
- = 1631032539 bytes / (1024 \* 1024 \* 1024) (GB/bytes)
- = 1.5 GB of redo log written since last checkpoint

### Determining the Checkpoint Age in XtraDB

MariaDB until 10.1

In MariaDB 10.1 and before, MariaDB uses XtraDB by default. In those versions, the checkpoint age can be determined by the [Innodb\\_checkpoint\\_age](#) status variable.

## Determining the Redo Log Occupancy

The redo log occupancy is the percentage of the InnoDB redo log capacity that is taken up by dirty pages that have not yet been flushed to the physical InnoDB tablespace files in a checkpoint. Therefore, it's determined by the following calculation:

```
innodb_log_occupancy
=
innodb_checkpoint_age
/
innodb_log_group_capacity
```

For example, if

```
innodb_checkpoint_age
is
1.5G
and
innodb_log_group_capacity
is
4G
```

, then we would have the following:

- innodb\_log\_occupancy  
=
- innodb\_checkpoint\_age  
/  
innodb\_log\_group\_capacity
- =  
1.5G  
/  
4G
- =  
0.375

If the calculated value for redo log occupancy is too close to

1.0  
, then the InnoDB redo log capacity may be too small for the current workload.

### 4.3.2.15 InnoDB Undo Log

#### Contents

1. [Overview](#)
2. [Implementation Details](#)
3. [Effects of Long-Running Transactions](#)
4. [Configuration](#)

## Overview

When a [transaction](#) writes data, it always inserts them in the table indexes or data (in the buffer pool or in physical files). No private copies are created.

The old versions of data being modified by active XtraDB/InnoDB transactions are stored in the undo log. The original data can then be restored, or viewed by a consistent read.

## Implementation Details

Before a row is modified, it is copied into the undo log. Each normal row contains a pointer to the most recent version of the same row in the undo log. Each row in the undo log contains a pointer to previous version, if any. So, each modified row has an history chain.

Rows are never physically deleted until a transaction ends. If they were deleted, the restore would be impossible. Thus, rows are simply marked for deletion.

Each transaction uses a view of the records. The [transaction level](#) determines how this view is created. For example, READ UNCOMMITTED usually uses the current version of rows, even if they are not committed (*dirty reads*). Other isolation levels require that the most recent committed version of rows is searched in the undo log. READ COMMITTED uses a different view for each table, while REPEATABLE READ and SERIALIZABLE use the same view for all tables.

There is also a global history list of the data. When a transaction is committed, its history is added to this history list. The order of the list is the chronological order of the commits.

The purge thread deletes the rows in the undo log which are not needed by any existing view. The rows for which a most recent version exists are deleted, as well as the delete-marked rows.

If InnoDB needs to restore an old version, it will simply replace the newer version with the older one. When a transaction inserts a new row, there is no older version. However, in that case, the restore can be done by deleting the inserted rows.

## Effects of Long-Running Transactions

Understanding how the undo log works helps with understanding the negative effects long transactions.

- Long transactions generate several old versions of the rows in the undo log. Those rows will probably be needed for a longer time, because other long transactions will need them. Since those transactions will generate more modified rows, a sort of combinatory explosion can be observed. Thus, the undo log requires more space.
- Transaction may need to read very old versions of the rows in the history list, thus their performance will degrade.

Of course read-only transactions do not write more entries in the undo log; however, they delay the purging of existing entries.

Also, long transactions can more likely result in deadlocks, but this problem is not related to the undo log.

## Configuration

The undo log is not a log file that can be viewed on disk in the usual sense, such as the [error log](#) or [slow query log](#), rather an area of storage.

The undo log is usually part of the physical system tablespace, but from [MariaDB 10.0](#), the [innodb\\_undo\\_directory](#) and [innodb\\_undo\\_tablespaces](#) system variables can be used to split into different tablespaces and store in a different location (perhaps on a different storage device).

Each insert or update portion of the undo log is known as a rollback segment. The [innodb\\_undo\\_logs](#) system variable specifies the number of rollback segments to be used per transaction.

The related [innodb\\_available\\_undo\\_logs](#) status variable stores the total number of available InnoDB undo logs.

### 4.3.2.16 InnoDB Page Flushing

#### Contents

1. [Page Flushing with InnoDB Page Cleaner Threads](#)
1. [Page Flushing with Multiple InnoDB Page Cleaner Threads](#)
2. [Page Flushing with a Single InnoDB Page Cleaner Thread](#)
2. [Page Flushing with Multi-threaded Flush Threads](#)
3. [Configuring the InnoDB I/O Capacity](#)
4. [See Also](#)

## Page Flushing with InnoDB Page Cleaner Threads

InnoDB page cleaner threads flush dirty pages from the [InnoDB buffer pool](#). These dirty pages are flushed using a least-recently used (LRU) algorithm.

### Page Flushing with Multiple InnoDB Page Cleaner Threads

MariaDB 10.2.2 - 10.5.1

The [innodb\\_page\\_cleaners](#) system variable was added in [MariaDB 10.2.2](#), and makes it possible to use multiple InnoDB page cleaner threads. It is deprecated and ignored from [MariaDB 10.5.1](#), as the original reasons for splitting the buffer pool have mostly gone away.

The number of InnoDB page cleaner threads can be configured by setting the [innodb\\_page\\_cleaners](#) system variable. This system variable can be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
innodb_page_cleaners=8
```

In [MariaDB 10.3.3](#) and later, this system variable can also be changed dynamically with [SET GLOBAL](#). For example:

```
SET GLOBAL innodb_page_cleaners=8;
```

This system variable's default value is either

4

or the configured value of the [innodb\\_buffer\\_pool\\_instances](#) system variable, whichever is lower.

## Page Flushing with a Single InnoDB Page Cleaner Thread

In [MariaDB 10.2.1](#) and before, and from [MariaDB 10.5.1](#), when the original reasons for splitting the buffer pool have mostly gone away, only a single InnoDB page cleaner thread is supported.

## Page Flushing with Multi-threaded Flush Threads

**MariaDB 10.1.0 - 10.3.2**

InnoDB's multi-thread flush feature was first added in [MariaDB 10.1.0](#). It was deprecated in [MariaDB 10.2.9](#) and removed in [MariaDB 10.3.2](#).

In [MariaDB 10.3.1](#) and before, InnoDB's multi-thread flush feature can be used. This is especially useful in [MariaDB 10.1](#), which only supports a single page cleaner thread.

InnoDB's multi-thread flush feature can be enabled by setting the [innodb\\_use\\_mtflush](#) system variable. The number of threads can be configured by setting the [innodb\\_mtflush\\_threads](#) system variable. This system variable can be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
innodb_use_mtflush = ON
innodb_mtflush_threads = 8
```

The [innodb\\_mtflush\\_threads](#) system variable's default value is

8

. The maximum value is

64

. In multi-core systems, it is recommended to set its value close to the configured value of the [innodb\\_buffer\\_pool\\_instances](#) system variable.

However, it is also recommended to use your own benchmarks to find a suitable value for your particular application.

InnoDB's multi-thread flush feature was deprecated in [MariaDB 10.2.9](#) and removed from [MariaDB 10.3.2](#). In later versions of MariaDB, use multiple InnoDB page cleaner threads instead.

## Configuring the InnoDB I/O Capacity

Increasing the amount of I/O capacity available to InnoDB can also help increase the performance of page flushing.

The amount of I/O capacity available to InnoDB can be configured by setting the [innodb\\_io\\_capacity](#) system variable. This system variable can be changed dynamically with [SET GLOBAL](#). For example:

```
SET GLOBAL innodb_io_capacity=20000;
```

This system variable can also be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
innodb_io_capacity=20000
```

The maximum amount of I/O capacity available to InnoDB in an emergency defaults to either

2000

or twice [innodb\\_io\\_capacity](#), whichever is higher, or can be directly configured by setting the [innodb\\_io\\_capacity\\_max](#) system variable. This system variable can be changed dynamically with [SET GLOBAL](#). For example:

```
SET GLOBAL innodb_io_capacity_max=20000;
```

This system variable can also be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
innodb_io_capacity_max=20000
```

## See Also

- [Significant performance boost with new MariaDB page compression on FusionIO](#)

### 4.3.2.17 InnoDB Purge

#### Contents

1. [Optimizing Purge Performance](#)
  1. [Configuring the Purge Threads](#)
  2. [Configuring the Purge Batch Size](#)
  3. [Configuring the Max Purge Lag](#)
  4. [Configuring the Purge Rollback Segment Truncation Frequency](#)
  5. [Configuring the Purge Undo Log Truncation](#)
2. [Purge's Effect on Row Metadata](#)

When a transaction updates a row in an InnoDB table, InnoDB's MVCC implementation keeps old versions of the row in the [InnoDB undo log](#). The old versions are kept at least until all transactions older than the transaction that updated the row are no longer open. At that point, the old versions can be deleted. InnoDB has purge process that is used to delete these old versions.

## Optimizing Purge Performance

### Configuring the Purge Threads

The number of purge threads can be set by configuring the

```
innodb_purge_threads
```

system variable. This system variable can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
innodb_purge_threads = 6
```

### Configuring the Purge Batch Size

The purge batch size is defined as the number of [InnoDB redo log](#) records that must be written before triggering purge. The purge batch size can be set by configuring the

```
innodb_purge_batch_size
```

system variable. This system variable can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
innodb_purge_batch_size = 50
```

### Configuring the Max Purge Lag

If purge operations are lagging on a busy server, then this can be a tough situation to recover from. As a solution, InnoDB allows you to set the max purge lag. The max purge lag is defined as the maximum number of [InnoDB undo log](#) that can be waiting to be purged from the history until InnoDB begins delaying DML statements.

The max purge lag can be set by configuring the

```
innodb_max_purge_lag
```

system variable. This system variable can be changed dynamically with

```
SET GLOBAL
```

. For example:

```
SET GLOBAL innodb_max_purge_lag=1000;
```

This system variable can also be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
innodb_max_purge_lag = 1000
```

The maximum delay can be set by configuring the

```
innodb_max_purge_lag_delay
```

system variable. This system variable can be changed dynamically with

```
SET GLOBAL
```

. For example:

```
SET GLOBAL innodb_max_purge_lag_delay=100;
```

This system variable can also be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
innodb_max_purge_lag_delay = 100
```

## Configuring the Purge Rollback Segment Truncation Frequency

MariaDB starting with [10.2.2](#)

The

```
innodb_purge_rseg_truncate_frequency
```

system variable was first added in [MariaDB 10.2.2](#).

The purge rollback segment truncation frequency is defined as the number of purge loops that are run before unnecessary rollback segments are truncated. The purge rollback segment truncation frequency can be set by configuring the

```
innodb_purge_rseg_truncate_frequency
```

system variable. This system variable can be changed dynamically with

```
SET GLOBAL
```

. For example:

```
SET GLOBAL innodb_purge_rseg_truncate_frequency=64;
```

This system variable can also be specified as a command-line argument to

`mysqld`

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
innodb_purge_rseg_truncate_frequency = 64
```

## Configuring the Purge Undo Log Truncation

MariaDB starting with [10.2.2](#)

The

`innodb_undo_log_truncate`

and

`innodb_max_undo_log_size`

system variables were first added in [MariaDB 10.2.2](#).

Purge undo log truncation occurs when InnoDB truncates an entire [InnoDB undo log](#) tablespace, rather than deleting individual [InnoDB undo log](#) records.

Purge undo log truncation can be enabled by configuring the

`innodb_undo_log_truncate`

system variable. This system variable can be changed dynamically with

`SET GLOBAL`

. For example:

```
SET GLOBAL innodb_undo_log_truncate=ON;
```

This system variable can also be specified as a command-line argument to

`mysqld`

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
innodb_undo_log_truncate = ON
```

An [InnoDB undo log](#) tablespace is truncated when it exceeds the maximum size that is configured for [InnoDB undo log](#) tablespaces. The maximum size can be set by configuring the

`innodb_max_undo_log_size`

system variable. This system variable can be changed dynamically with

`SET GLOBAL`

. For example:

```
SET GLOBAL innodb_max_undo_log_size='64M';
```

This system variable can also be specified as a command-line argument to

`mysqld`

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
innodb_max_undo_log_size = 64M
```

## Purge's Effect on Row Metadata

An InnoDB table's clustered index has three hidden system columns that are automatically generated. These hidden system columns are:

- - DB\_ROW\_ID
    - If the table has no other PRIMARY KEY or no other UNIQUE KEY defined as NOT NULL that can be promoted to the table's PRIMARY KEY, then InnoDB will use a hidden system column called DB\_ROW\_ID
      - . InnoDB will automatically generate the value for the column from a global InnoDB-wide 48-bit sequence (instead of being table-local).
- - DB\_TRX\_ID
    - The transaction ID of either the transaction that last changed the row or the transaction that currently has the row locked.
- - DB\_ROLL\_PTR
    - A pointer to the [InnoDB undo log](#) that contains the row's previous record. The value of DB\_ROLL\_PTR is only valid if DB\_TRX\_ID belongs to the current read view. The oldest valid read view is the purge view.

If a row's last [InnoDB undo log](#) record is purged, this can obviously affect the value of the row's

DB\_ROLL\_PTR column, because there would no longer be any [InnoDB undo log](#) record for the pointer to reference.

In [MariaDB 10.2](#) and before, the purge process wouldn't touch the value of the row's

DB\_TRX\_ID column.

However, in [MariaDB 10.3](#) and later, the purge process will set a row's

DB\_TRX\_ID column to 0

after all of the row's associated [InnoDB undo log](#) records have been deleted. This change allows InnoDB to perform an optimization: if a query wants to read a row, and if the row's

DB\_TRX\_ID column is set to 0

, then it knows that no other transaction has the row locked. Usually, InnoDB needs to lock the transaction system's mutex in order to safely check whether a row is locked, but this optimization allows InnoDB to confirm that the row can be safely read without any heavy internal locking.

This optimization can speed up reads, but it comes at a noticeable cost at other times. For example, it can cause the purge process to use more I/O after inserting a lot of rows, since the value of each row's

DB\_TRX\_ID column will have to be reset.

### 4.3.2.18 Information Schema InnoDB Tables

### 4.3.2.19 InnoDB Online DDL

#### 4.3.2.19.1 InnoDB Online DDL Overview

## Contents

- 1. Alter Algorithms
- 2. Specifying an Alter Algorithm
  - 1. Specifying an Alter Algorithm Using the ALGORITHM Clause
  - 2. Specifying an Alter Algorithm Using System Variables
- 3. Supported Alter Algorithms
  - 1. DEFAULT Algorithm
  - 2. COPY Algorithm
    - 1. Using the COPY Algorithm with InnoDB
  - 3. INPLACE Algorithm
    - 1. Using the INPLACE Algorithm with InnoDB
    - 2. Operations Supported by InnoDB with the INPLACE Algorithm
  - 4. NOCOPY Algorithm
    - 1. Operations Supported by InnoDB with the NOCOPY Algorithm
  - 5. INSTANT Algorithm
    - 1. Operations Supported by InnoDB with the INSTANT Algorithm
- 4. Alter Locking Strategies
- 5. Specifying an Alter Locking Strategy
  - 1. Specifying an Alter Locking Strategy Using the LOCK Clause
  - 2. Specifying an Alter Locking Strategy Using ALTER ONLINE TABLE
- 6. Supported Alter Locking Strategies
  - 1. DEFAULT Locking Strategy
  - 2. NONE Locking Strategy
  - 3. SHARED Locking Strategy
  - 4. EXCLUSIVE Locking Strategy

InnoDB tables support online DDL, which permits concurrent DML and uses optimizations to avoid unnecessary table copying.

The `ALTER TABLE` statement supports two clauses that are used to implement online DDL:

- `ALGORITHM` - This clause controls how the DDL operation is performed.
- `LOCK` - This clause controls how much concurrency is allowed while the DDL operation is being performed.

## Alter Algorithms

InnoDB supports multiple algorithms for performing DDL operations. This offers a significant performance improvement over previous versions. The supported algorithms are:

- - DEFAULT
    - This implies the default behavior for the specific operation.
- - COPY
- - INPLACE
- - NOCOPY
    - This was added in [MariaDB 10.3.7](#).
- - INSTANT
    - This was added in [MariaDB 10.3.7](#).

## Specifying an Alter Algorithm

The set of alter algorithms can be considered as a hierarchy. The hierarchy is ranked in the following order, with *least efficient* algorithm at the top, and *most efficient* algorithm at the bottom:

- - COPY
-

INPLACE

•

NOCOPY

•

INSTANT

When a user specifies an alter algorithm for a DDL operation, MariaDB does not necessarily use that specific algorithm for the operation. It interprets the choice in the following way:

- If the user specifies

COPY

, then InnoDB uses the

COPY

algorithm.

- If the user specifies any other algorithm, then InnoDB interprets that choice as the *least efficient* algorithm that the user is willing to accept. This means that if the user specifies

INPLACE

, then InnoDB will use the *most efficient* algorithm supported by the specific operation from the set (

INPLACE

,

NOCOPY

,

INSTANT

). Likewise, if the user specifies

NOCOPY

, then InnoDB will use the *most efficient* algorithm supported by the specific operation from the set (

NOCOPY

,

INSTANT

).

There is also a special value that can be specified:

- If the user specifies

DEFAULT

, then InnoDB uses its default choice for the operation. The default choice is to use the most efficient algorithm supported by the operation. The default choice will also be used if no algorithm is specified. Therefore, if you want InnoDB to use the most efficient algorithm supported by an operation, then you usually do not have to explicitly specify any algorithm at all.

## Specifying an Alter Algorithm Using the

ALGORITHM

Clause

InnoDB supports the

ALGORITHM

clause.

The ALGORITHM clause can be used to specify the *least efficient* algorithm that the user is willing to accept. It is supported by the ALTER TABLE and CREATE INDEX statements.

For example, if a user wanted to add a column to a table, but only if the operation used an algorithm that is at least as efficient as the

INPLACE

, then they could execute the following:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50)
);

ALTER TABLE tab ADD COLUMN c varchar(50), ALGORITHM=INPLACE;
```

In MariaDB 10.3 and later, the above operation would actually use the

INSTANT

algorithm, because the

ADD COLUMN

operation supports the

INSTANT

algorithm, and the

INSTANT  
algorithm is more efficient than the  
INPLACE  
algorithm.

## Specifying an Alter Algorithm Using System Variables

MariaDB starting with 10.3

In MariaDB 10.3 and later, the `alter_algorithm` system variable can be used to pick the *least efficient* algorithm that the user is willing to accept.

For example, if a user wanted to add a column to a table, but only if the operation used an algorithm that is at least as efficient as the

INPLACE

, then they could execute the following:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD COLUMN c varchar(50);
```

In MariaDB 10.3 and later, the above operation would actually use the

INSTANT

algorithm, because the

ADD COLUMN

operation supports the

INSTANT

algorithm, and the

INSTANT

algorithm is more efficient than the

INPLACE

algorithm.

MariaDB until 10.2

In MariaDB 10.2 and before, the `old_alter_table` system variable can be used to specify whether the

COPY

algorithm should be used.

For example, if a user wanted to add a column to a table, but they wanted to use the

COPY

algorithm instead of the default algorithm for the operation, then they could execute the following:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50)
);

SET SESSION old_alter_table=1;
ALTER TABLE tab ADD COLUMN c varchar(50);
```

## Supported Alter Algorithms

The supported algorithms are described in more details below.

### DEFAULT Algorithm

The default behavior, which occurs if

`ALGORITHM=DEFAULT`

is specified, or if

`ALGORITHM`

is not specified at all, usually only makes a copy if the operation doesn't support being done in-place at all. In this case, the *most efficient* available algorithm will usually be used.

This means that, if an operation supports the

INSTANT  
algorithm, then it will use that algorithm by default. If an operation does not support the  
INSTANT  
algorithm, but it does support the  
NOCOPY  
algorithm, then it will use that algorithm by default. If an operation does not support the  
NOCOPY  
algorithm, but it does support the  
INPLACE  
algorithm, then it will use that algorithm by default.

## COPY Algorithm

The

COPY  
algorithm refers to the original [ALTER TABLE](#) algorithm.

When the

COPY  
algorithm is used, MariaDB essentially does the following operations:

```
-- Create a temporary table with the new definition
CREATE TEMPORARY TABLE tmp_tab (
...
);

-- Copy the data from the original table
INSERT INTO tmp_tab
    SELECT * FROM original_tab;

-- Drop the original table
DROP TABLE original_tab;

-- Rename the temporary table, so that it replaces the original one
RENAME TABLE tmp_tab TO original_tab;
```

This algorithm is very inefficient, but it is generic, so it works for all storage engines.

If the

COPY  
algorithm is specified with the

[ALGORITHM](#)

clause or with the

[alter\\_algorithm](#)

system variable, then the

COPY  
algorithm will be used even if it is not necessary. This can result in a lengthy table copy. If multiple

[ALTER TABLE](#)

operations are required that each require the table to be rebuilt, then it is best to specify all operations in a single

[ALTER TABLE](#)

statement, so that the table is only rebuilt once.

Using the

COPY  
Algorithm with InnoDB

If the

COPY  
algorithm is used with an [InnoDB](#) table, then the following statements apply:

- The table will be rebuilt using the current values of the [innodb\\_file\\_per\\_table](#), [innodb\\_file\\_format](#), and [innodb\\_default\\_row\\_format](#) system variables.

- The operation will have to create a temporary table to perform the table copy. This temporary table will be in the same directory as the original table, and its file name will be in the format

```
#  
  
sql${PID}_${THREAD_ID}_${TMP_TABLE_COUNT}  
, where  
 ${PID}  
is the process ID of  
mysqld  
,  
 ${THREAD_ID}  
is the connection ID, and  
 ${TMP_TABLE_COUNT}  
is the number of temporary tables that the connection has open. Therefore, the
```

`datadir`

may contain files with file names like

```
#  
  
sql1234_12_1.ibd
```

- The operation inserts one record at a time into each index, which is very inefficient.
- InnoDB does not use a sort buffer.
- In MariaDB 10.2.13 , MariaDB 10.3.5 and later, the table copy operation creates a lot fewer InnoDB undo log writes. See MDEV-11415 for more information.
- The table copy operation creates a lot of InnoDB redo log writes.

## INPLACE Algorithm

The

`COPY`  
algorithm can be incredibly slow, because the whole table has to be copied and rebuilt. The  
`INPLACE`  
algorithm was introduced as a way to avoid this by performing operations in-place and avoiding the table copy and rebuild, when possible.

When the

`INPLACE`  
algorithm is used, the underlying storage engine uses optimizations to perform the operation while avoiding the table copy and rebuild. However,

`INPLACE`  
is a bit of a misnomer, since some operations may still require the table to be rebuilt for some storage engines. Regardless, several operations can be performed without a full copy of the table for some storage engines.

A more accurate name for the algorithm would have been the

`ENGINE`  
algorithm, since the `storage engine` decides how to implement the algorithm.

If an `ALTER TABLE` operation supports the

`INPLACE`  
algorithm, then it can be performed using optimizations by the underlying storage engine, but it may rebuilt.

If the

`INPLACE`  
algorithm is specified with the `ALGORITHM` clause or with the `alter_algorithm` system variable and if the `ALTER TABLE` operation does not support the  
`INPLACE`  
algorithm, then an error will be raised. For example:

```
SET SESSION alter_algorithm='INPLACE';  
  
ALTER TABLE tab MODIFY COLUMN c int;  
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

In this case, raising an error is preferable, if the alternative is for the operation to make a copy of the table, and perform unexpectedly slowly.

## Using the

### INPLACE Algorithm with InnoDB

If the

INPLACE

algorithm is used with an [InnoDB](#) table, then the following statements apply:

- The operation might have to write sort files in the directory defined by the `innodb_tmpdir` system variable.
- The operation might also have to write a temporary log file to track data changes by [DML queries](#) executed during the operation. The maximum size for this log file is configured by the `innodb_online_alter_log_max_size` system variable.
- Some operations require the table to be rebuilt, even though the algorithm is inaccurately called "in-place". This includes operations such as adding or dropping columns, adding a primary key, changing a column to `NULL`, etc.
- If the operation requires the table to be rebuilt, then the operation might have to create temporary tables.
  - It may have to create a temporary intermediate table for the actual table rebuild operation.
    - In [MariaDB 10.2.19](#) and later, this temporary table will be in the same directory as the original table, and its file name will be in the format

```
#  
  
sql${PID}_${THREAD_ID}_${TMP_TABLE_COUNT}  
, where  
${PID}  
is the process ID of  
mysqld  
,  
${THREAD_ID}  
is the connection ID, and  
${TMP_TABLE_COUNT}  
is the number of temporary tables that the connection has open. Therefore, the
```

`datadir`

may contain files with file names like

```
#  
  
sql1234_12_1.ibd
```

- In [MariaDB 10.2.18](#) and before, this temporary table will be in the same directory as the original table, and its file name will be in the format

```
#  
  
sql-ib${TABLESPACE_ID}-${RAND}  
, where  
${TABLESPACE_ID}  
is the table's tablespace ID within InnoDB and  
${RAND}  
is a randomly initialized number. Therefore, the datadir may contain files with file names like
```

```
#  
  
sql-ib230291-1363966925.ibd
```

- When it replaces the original table with the rebuilt table, it may also have to rename the original table using a temporary table name.

- If the server is [MariaDB 10.3](#) or later or if it is running [MariaDB 10.2](#) and the `innodb_safe_truncate` system variable is set to `OFF`, then the format will actually be

```
#  
  
sql-ib${TABLESPACE_ID}-${RAND}  
, where  
${TABLESPACE_ID}  
is the table's tablespace ID within InnoDB and  
${RAND}  
is a randomly initialized number. Therefore, the datadir may contain files with file names like
```

#

sql\_ib230291-1363966925.ibd

- If the server is running [MariaDB 10.1](#) or before or if it is running [MariaDB 10.2](#) and the `innodb_safe_truncate` system variable is set to `ON`, then the renamed table will have a temporary table name in the format  
#  
`sql-ib${TABLESPACE_ID}`  
, where  
`${TABLESPACE_ID}`  
is the table's tablespace ID within InnoDB. Therefore, the `datadir` may contain files with file names like  
#  
`sql-ib230291.ibd`

- The storage needed for the above items can add up to the size of the original table, or more in some cases.
- Some operations are instantaneous, if they only require the table's metadata to be changed. This includes operations such as renaming a column, changing a column's `DEFAULT` value, etc.

## Operations Supported by InnoDB with the INPLACE Algorithm

With respect to the allowed operations, the

`INPLACE`  
algorithm supports a subset of the operations supported by the  
`COPY`  
algorithm, and it supports a superset of the operations supported by the  
`NOCOPY`  
algorithm.

See [InnoDB Online DDL Operations with ALGORITHM=INPLACE](#) for more information.

## NOCOPY Algorithm

MariaDB starting with [10.3](#)

In [MariaDB 10.3](#) and later, the  
`NOCOPY`  
algorithm is supported.

The

`INPLACE`  
algorithm can sometimes be surprisingly slow in instances where it has to rebuild the clustered index, because when the clustered index has to be rebuilt, the whole table has to be rebuilt. The  
`NOCOPY`  
algorithm was introduced as a way to avoid this.

If an `ALTER TABLE` operation supports the

`NOCOPY`  
algorithm, then it can be performed without rebuilding the clustered index.

If the

`NOCOPY`  
algorithm is specified with the `ALGORITHM` clause or with the `alter_algorithm` system variable and if the `ALTER TABLE` operation does not support the  
`NOCOPY`  
algorithm, then an error will be raised. For example:

```
SET SESSION alter_algorithm='NOCOPY';

ALTER TABLE tab MODIFY COLUMN c int;
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

In this case, raising an error is preferable, if the alternative is for the operation to rebuild the clustered index, and perform unexpectedly slowly.

## Operations Supported by InnoDB with the NOCOPY Algorithm

With respect to the allowed operations, the

NOCOPY algorithm supports a subset of the operations supported by the INPLACE algorithm, and it supports a superset of the operations supported by the INSTANT algorithm.

See [InnoDB Online DDL Operations with ALGORITHM=NOCOPY](#) for more information.

## INSTANT Algorithm

MariaDB starting with 10.3

In MariaDB 10.3 and later, the INSTANT algorithm is supported.

The

INPLACE algorithm can sometimes be surprisingly slow in instances where it has to modify data files. The INSTANT algorithm was introduced as a way to avoid this.

If an [ALTER TABLE](#) operation supports the

INSTANT algorithm, then it can be performed without modifying any data files.

If the

INSTANT algorithm is specified with the [ALGORITHM](#) clause or with the [alter\\_algorithm](#) system variable and if the [ALTER TABLE](#) operation does not support the INSTANT algorithm, then an error will be raised. For example:

```
SET SESSION alter_algorithm='INSTANT';

ALTER TABLE tab MODIFY COLUMN c int;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

In this case, raising an error is preferable, if the alternative is for the operation to modify data files, and perform unexpectedly slowly.

## Operations Supported by InnoDB with the INSTANT Algorithm

With respect to the allowed operations, the

INSTANT algorithm supports a subset of the operations supported by the NOCOPY algorithm.

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT](#) for more information.

# Alter Locking Strategies

InnoDB supports multiple locking strategies for performing DDL operations. This offers a significant performance improvement over previous versions. The supported locking strategies are:

- DEFAULT
  - This implies the default behavior for the specific operation.
- NONE

- SHARED

- EXCLUSIVE

Regardless of which locking strategy is used to perform a DDL operation, InnoDB will have to exclusively lock the table for a short time at the start and end of the operation's execution. This means that any active transactions that may have accessed the table must be committed or aborted for the operation to continue. This applies to most DDL statements, such as [ALTER TABLE](#), [CREATE INDEX](#), [DROP INDEX](#), [OPTIMIZE TABLE](#), [RENAME TABLE](#), etc.

## Specifying an Alter Locking Strategy

### Specifying an Alter Locking Strategy Using the LOCK Clause

The [ALTER TABLE](#) statement supports the [LOCK](#) clause.

The [LOCK](#) clause can be used to specify the locking strategy that the user is willing to accept. It is supported by the [ALTER TABLE](#) and [CREATE INDEX](#) statements.

For example, if a user wanted to add a column to a table, but only if the operation is non-locking, then they could execute the following:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50)
);

ALTER TABLE tab ADD COLUMN c varchar(50), ALGORITHM=INPLACE, LOCK=NONE;
```

If the [LOCK](#) clause is not explicitly set, then the operation uses

LOCK=DEFAULT

### Specifying an Alter Locking Strategy Using ALTER ONLINE TABLE

[ALTER ONLINE TABLE](#) is equivalent to

LOCK=NONE

. Therefore, the [ALTER ONLINE TABLE](#) statement can be used to ensure that your [ALTER TABLE](#) operation allows all concurrent DML.

## Supported Alter Locking Strategies

The supported algorithms are described in more details below.

To see which locking strategies InnoDB supports for each operation, see the pages that describe which operations are supported for each algorithm:

- [InnoDB Online DDL Operations with ALGORITHM=INPLACE](#)
- [InnoDB Online DDL Operations with ALGORITHM=NOCOPY](#)
- [InnoDB Online DDL Operations with ALGORITHM=INSTANT](#)

### DEFAULT Locking Strategy

The default behavior, which occurs if

LOCK=DEFAULT

is specified, or if

LOCK

is not specified at all, acquire the least restrictive lock on the table that is supported for the specific operation. This permits the maximum amount of concurrency that is supported for the specific operation.

### NONE Locking Strategy

The

NONE

locking strategy performs the operation without acquiring any lock on the table. This permits **all** concurrent DML.

If this locking strategy is not permitted for an operation, then an error is raised.

## SHARED Locking Strategy

The

SHARED

locking strategy performs the operation after acquiring a read lock on the table. This permit **read-only** concurrent DML.

If this locking strategy is not permitted for an operation, then an error is raised.

## EXCLUSIVE Locking Strategy

The

EXCLUSIVE

locking strategy performs the operation after acquiring a write lock on the table. This does **not** permit concurrent DML.

## 4.3.2.19.2 InnoDB Online DDL Operations with the INPLACE Alter Algorithm

## Contents

- 1. Supported Operations by Inheritance
- 2. Column Operations
  - 1. [ALTER TABLE ... ADD COLUMN](#)
  - 2. [ALTER TABLE ... DROP COLUMN](#)
  - 3. [ALTER TABLE ... MODIFY COLUMN](#)
    - 1. Reordering Columns
    - 2. [Changing the Data Type of a Column](#)
    - 3. [Changing a Column to NULL](#)
    - 4. [Changing a Column to NOT NULL](#)
    - 5. [Adding a New ENUM Option](#)
    - 6. [Adding a New SET Option](#)
    - 7. [Removing System Versioning from a Column](#)
  - 4. [ALTER TABLE ... ALTER COLUMN](#)
    - 1. [Setting a Column's Default Value](#)
    - 2. [Removing a Column's Default Value](#)
  - 5. [ALTER TABLE ... CHANGE COLUMN](#)
- 3. Index Operations
  - 1. [ALTER TABLE ... ADD PRIMARY KEY](#)
  - 2. [ALTER TABLE ... DROP PRIMARY KEY](#)
  - 3. [ALTER TABLE ... ADD INDEX and CREATE INDEX](#)
    - 1. [Adding a Plain Index](#)
    - 2. [Adding a Fulltext Index](#)
    - 3. [Adding a Spatial Index](#)
  - 4. [ALTER TABLE ... DROP INDEX and DROP INDEX](#)
  - 5. [ALTER TABLE ... ADD FOREIGN KEY](#)
  - 6. [ALTER TABLE ... DROP FOREIGN KEY](#)
- 4. Table Operations
  - 1. [ALTER TABLE ... AUTO\\_INCREMENT=...](#)
  - 2. [ALTER TABLE ... ROW\\_FORMAT=...](#)
  - 3. [ALTER TABLE ... KEY\\_BLOCK\\_SIZE=...](#)
  - 4. [ALTER TABLE ... PAGE\\_COMPRESSED=... and ALTER TABLE ... PAGE\\_COMPRESSION\\_LEVEL=...](#)
  - 5. [ALTER TABLE ... DROP SYSTEM VERSIONING](#)
  - 6. [ALTER TABLE ... DROP CONSTRAINT](#)
  - 7. [ALTER TABLE ... FORCE](#)
  - 8. [ALTER TABLE ... ENGINE=InnoDB](#)
  - 9. [OPTIMIZE TABLE ...](#)
  - 10. [ALTER TABLE ... RENAME TO and RENAME TABLE ...](#)
- 5. Limitations
  - 1. [Limitations Related to Fulltext Indexes](#)
  - 2. [Limitations Related to Spatial Indexes](#)
  - 3. [Limitations Related to Generated \(Virtual and Persistent/Stored\) Columns](#)

## Supported Operations by Inheritance

When the [ALGORITHM](#) clause is set to

[INPLACE](#)

, the supported operations are a superset of the operations that are supported when the [ALGORITHM](#) clause is set to [NOCOPY](#)

. Similarly, when the [ALGORITHM](#) clause is set to

[NOCOPY](#)

, the supported operations are a superset of the operations that are supported when the [ALGORITHM](#) clause is set to [INSTANT](#)

Therefore, when the [ALGORITHM](#) clause is set to

INPLACE

, some operations are supported by inheritance. See the following additional pages for more information about these supported operations:

- [InnoDB Online DDL Operations with ALGORITHM=NOCOPY](#)
- [InnoDB Online DDL Operations with ALGORITHM=INSTANT](#)

## Column Operations

### ALTER TABLE ... ADD COLUMN

InnoDB supports adding columns to a table with [ALGORITHM](#) set to

INPLACE

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

With the exception of adding an [auto-increment](#) column, this operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD COLUMN c varchar(50);
Query OK, 0 rows affected (0.006 sec)
```

This applies to [ALTER TABLE ... ADD COLUMN](#) for InnoDB tables.

### ALTER TABLE ... DROP COLUMN

InnoDB supports dropping columns from a table with [ALGORITHM](#) set to

INPLACE

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP COLUMN c;
Query OK, 0 rows affected (0.021 sec)
```

This applies to [ALTER TABLE ... DROP COLUMN](#) for InnoDB tables.

### ALTER TABLE ... MODIFY COLUMN

This applies to [ALTER TABLE ... MODIFY COLUMN](#) for InnoDB tables.

## Reordering Columns

InnoDB supports reordering columns within a table with `ALGORITHM` set to `INPLACE`.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to `NONE`.

. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c varchar(50) AFTER a;
Query OK, 0 rows affected (0.022 sec)
```

## Changing the Data Type of a Column

InnoDB does **not** support modifying a column's data type with `ALGORITHM` set to

`INPLACE`

in most cases. There are some exceptions:

- In [MariaDB 10.2.2](#) and later, InnoDB supports increasing the length of  
VARCHAR  
columns with `ALGORITHM` set to  
`INPLACE`, unless it would require changing the number of bytes required to represent the column's length. A  
VARCHAR  
column that is between 0 and 255 bytes in size requires 1 byte to represent its length, while a  
VARCHAR  
column that is 256 bytes or longer requires 2 bytes to represent its length. This means that the length of a column cannot be increased with `ALGORITHM` set to  
`INPLACE`  
if the original length was less than 256 bytes, and the new length is 256 bytes or more.
- In [MariaDB 10.4.3](#) and later, InnoDB supports increasing the length of  
VARCHAR  
columns with `ALGORITHM` set to  
`INPLACE` in the cases where the operation supports having the `ALGORITHM` clause set to  
`INSTANT`.

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Changing the Data Type of a Column](#) for more information.

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c int;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

But this succeeds in [MariaDB 10.2.2](#) and later, because the original length of the column is less than 256 bytes, and the new length is still less than 256 bytes:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) CHARACTER SET=latin1;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c varchar(100);
Query OK, 0 rows affected (0.005 sec)

```

But this fails in [MariaDB 10.2.2](#) and later, because the original length of the column is less than 256 bytes, and the new length is greater than 256 bytes:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(255)
) CHARACTER SET=latin1;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c varchar(256);
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY

```

## Changing a Column to NULL

InnoDB supports modifying a column to allow `NULL` values with `ALGORITHM` set to `INPLACE`

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to `NONE`

- . When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50) NOT NULL
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c varchar(50) NULL;
Query OK, 0 rows affected (0.021 sec)

```

## Changing a Column to NOT NULL

InnoDB supports modifying a column to **not** allow `NULL` values with `ALGORITHM` set to

- `INPLACE`
- . It is required for `strict mode` to be enabled in `SQL_MODE`. The operation will fail if the column contains any `NULL` values. Changes that would interfere with referential integrity are also not permitted.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to `NONE`

- . When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c varchar(50) NOT NULL;
Query OK, 0 rows affected (0.021 sec)

```

## Adding a New ENUM

## Option

InnoDB supports adding a new **ENUM** option to a column with **ALGORITHM** set to

INPLACE  
. In order to add a new **ENUM** option with **ALGORITHM** set to  
INPLACE  
, the following requirements must be met:

- It must be added to the end of the list.
- The storage requirements must not change.

This operation only changes the table's metadata, so the table does not have to be rebuilt..

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the **LOCK** clause to

NONE  
. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c ENUM('red', 'green')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c ENUM('red', 'green', 'blue');
Query OK, 0 rows affected (0.004 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c ENUM('red', 'green')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c ENUM('red', 'blue', 'green');
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

## Adding a New

SET  
Option

InnoDB supports adding a new **SET** option to a column with **ALGORITHM** set to

INPLACE  
. In order to add a new **SET** option with **ALGORITHM** set to  
INPLACE  
, the following requirements must be met:

- It must be added to the end of the list.
- The storage requirements must not change.

This operation only changes the table's metadata, so the table does not have to be rebuilt..

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the **LOCK** clause to

NONE  
. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c SET('red', 'green')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c SET('red', 'green', 'blue');
Query OK, 0 rows affected (0.004 sec)
```

But this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c SET('red', 'green')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c SET('red', 'blue', 'green');
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY

```

## Removing System Versioning from a Column

In MariaDB 10.3.8 and later, InnoDB supports removing system versioning from a column with `ALGORITHM` set to

- INPLACE
  - . In order for this to work, the `system_versioning_alter_history` system variable must be set to `KEEP`
  - . See [MDEV-16330](#) for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

- NONE
  - . When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50) WITH SYSTEM VERSIONING
);

SET SESSION system_versioning_alter_history='KEEP';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab MODIFY COLUMN c varchar(50) WITHOUT SYSTEM VERSIONING;
Query OK, 0 rows affected (0.005 sec)

```

## ALTER TABLE ... ALTER COLUMN

This applies to `ALTER TABLE ... ALTER COLUMN` for InnoDB tables.

### Setting a Column's Default Value

InnoDB supports modifying a column's `DEFAULT` value with `ALGORITHM` set to

- INPLACE
  - .

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

- NONE
  - . When this strategy is used, all concurrent DML is permitted. For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ALTER COLUMN c SET DEFAULT 'No value explicitly provided.';
Query OK, 0 rows affected (0.005 sec)

```

### Removing a Column's Default Value

InnoDB supports removing a column's `DEFAULT` value with `ALGORITHM` set to

- INPLACE
  - .

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

- NONE
  - .

. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50) DEFAULT 'No value explicitly provided.'
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ALTER COLUMN c DROP DEFAULT;
Query OK, 0 rows affected (0.005 sec)
```

## ALTER TABLE ... CHANGE COLUMN

InnoDB supports renaming a column with [ALGORITHM](#) set to

INPLACE

, unless the column's data type or attributes changed in addition to the name.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab CHANGE COLUMN c str varchar(50);
Query OK, 0 rows affected (0.006 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab CHANGE COLUMN c num int;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

This applies to [ALTER TABLE ... CHANGE COLUMN](#) for InnoDB tables.

## Index Operations

### ALTER TABLE ... ADD PRIMARY KEY

InnoDB supports adding a primary key to a table with [ALGORITHM](#) set to

INPLACE

If the new primary key column is not defined as [NOT NULL](#), then it is highly recommended for [strict mode](#) to be enabled in [SQL\\_MODE](#). Otherwise, [NULL](#) values will be silently converted to the default value for the given data type, which is probably not the desired behavior in this scenario.

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int,
    b varchar(50),
    c varchar(50)
);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD PRIMARY KEY (a);
Query OK, 0 rows affected (0.021 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
    a int,
    b varchar(50),
    c varchar(50)
);

INSERT INTO tab VALUES (NULL, NULL, NULL);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD PRIMARY KEY (a);
ERROR 1265 (01000): Data truncated for column 'a' at row 1
```

And this fails:

```
CREATE OR REPLACE TABLE tab (
    a int,
    b varchar(50),
    c varchar(50)
);

INSERT INTO tab VALUES (1, NULL, NULL);
INSERT INTO tab VALUES (1, NULL, NULL);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD PRIMARY KEY (a);
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

This applies to [ALTER TABLE ... ADD PRIMARY KEY](#) for InnoDB tables.

## ALTER TABLE ... DROP PRIMARY KEY

InnoDB does **not** support dropping a primary key with [ALGORITHM](#) set to

INPLACE  
in most cases.

If you try to do so, then you will see an error. InnoDB only supports this operation with [ALGORITHM](#) set to

COPY  
. Concurrent DML is \*not\* permitted.

However, there is an exception. If you are dropping a primary key, and adding a new one at the same time, then that operation can be performed with [ALGORITHM](#) set to

INPLACE  
. This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to  
NONE  
. When this strategy is used, all concurrent DML is permitted.

For example, this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP PRIMARY KEY;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Dropping a primary key is not allowed without also adding a new

```

But this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP PRIMARY KEY, ADD PRIMARY KEY (b);
Query OK, 0 rows affected (0.020 sec)

```

This applies to [ALTER TABLE ... DROP PRIMARY KEY](#) for InnoDB tables.

ALTER TABLE ... ADD INDEX  
and  
CREATE INDEX

This applies to [ALTER TABLE ... ADD INDEX](#) and [CREATE INDEX](#) for InnoDB tables.

### Adding a Plain Index

InnoDB supports adding a plain index to a table with [ALGORITHM](#) set to

- . INPLACE
- . The table is not rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

- . NONE
- . When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD INDEX b_index (b);
Query OK, 0 rows affected (0.010 sec)

```

And this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
CREATE INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.011 sec)

```

### Adding a Fulltext Index

InnoDB supports adding a [FULLTEXT](#) index to a table with [ALGORITHM](#) set to

INPLACE

. The table is not rebuilt in some cases.

However, there are some limitations, such as:

- Adding a **FULLTEXT** index to a table that does not have a user-defined **FTS\_DOC\_ID** column will require the table to be rebuilt once. When the table is rebuilt, the system adds a hidden **FTS\_DOC\_ID** column. From that point forward, adding additional **FULLTEXT** indexes to the same table will not require the table to be rebuilt when **ALGORITHM** is set to **INPLACE**.
- Only one **FULLTEXT** index may be added at a time when **ALGORITHM** is set to **INPLACE**.
- If a table has more than one **FULLTEXT** index, then it cannot be rebuilt by any **ALTER TABLE** operations when **ALGORITHM** is set to **INPLACE**.
- If a table has a **FULLTEXT** index, then it cannot be rebuilt by any **ALTER TABLE** operations when the **LOCK** clause is set to **NONE**.

This operation supports a read-only locking strategy. This strategy can be explicitly chosen by setting the **LOCK** clause to **SHARED**.

. When this strategy is used, read-only concurrent DML is permitted.

For example, this succeeds, but requires the table to be rebuilt, so that the hidden **FTS\_DOC\_ID** column can be added:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.055 sec)
```

And this succeeds in the same way as above:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
CREATE FULLTEXT INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.041 sec)
```

And this succeeds, and the second command does not require the table to be rebuilt:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.043 sec)

ALTER TABLE tab ADD FULLTEXT INDEX c_index (c);
Query OK, 0 rows affected (0.017 sec)
```

But this second command fails, because only one **FULLTEXT** index can be added at a time:

```

CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50),
    d varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.041 sec)

ALTER TABLE tab ADD FULLTEXT INDEX c_index (c), ADD FULLTEXT INDEX d_index (d);
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: InnoDB presently supports one FULLTEXT index creation at a time.

```

And this third command fails, because a table cannot be rebuilt when it has more than one [FULLTEXT](#) index:

```

CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.040 sec)

ALTER TABLE tab ADD FULLTEXT INDEX c_index (c);
Query OK, 0 rows affected (0.015 sec)

ALTER TABLE tab FORCE;
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: InnoDB presently supports one FULLTEXT index creation at a time.

```

## Adding a Spatial Index

InnoDB supports adding a [SPATIAL](#) index to a table with [ALGORITHM](#) set to  
INPLACE

However, there are some limitations, such as:

- If a table has a [SPATIAL](#) index, then it cannot be rebuilt by any [ALTER TABLE](#) operations when the [LOCK](#) clause is set to  
NONE

This operation supports a read-only locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to  
SHARED  
. When this strategy is used, read-only concurrent DML is permitted.

For example, this succeeds:

```

CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD SPATIAL INDEX c_index (c);
Query OK, 0 rows affected (0.006 sec)

```

And this succeeds in the same way as above:

```

CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='INPLACE';
CREATE SPATIAL INDEX c_index ON tab (c);
Query OK, 0 rows affected (0.006 sec)

```

## ALTER TABLE ... DROP INDEX and DROP INDEX

InnoDB supports dropping indexes from a table with [ALGORITHM](#) set to  
INPLACE

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to  
NONE  
. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50),
    INDEX b_index (b)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP INDEX b_index;
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50),
    INDEX b_index (b)
);

SET SESSION alter_algorithm='INPLACE';
DROP INDEX b_index ON tab;
```

This applies to [ALTER TABLE ... DROP INDEX](#) and [DROP INDEX](#) for InnoDB tables.

## ALTER TABLE ... ADD FOREIGN KEY

InnoDB supports adding foreign key constraints to a table with [ALGORITHM](#) set to  
INPLACE  
. In order to add a new foreign key constraint to a table with [ALGORITHM](#) set to  
INPLACE  
, the [foreign\\_key\\_checks](#) system variable needs to be set to  
OFF  
. If it is set to  
ON  
, then  
[ALGORITHM=COPY](#)  
is required.

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to  
NONE  
. When this strategy is used, all concurrent DML is permitted.

For example, this fails:

```

CREATE OR REPLACE TABLE tab1 (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  d int
);

CREATE OR REPLACE TABLE tab2 (
  a int PRIMARY KEY,
  b varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
ERROR 1846 (0A000): ALGORITHM=INPLACE is not supported. Reason: Adding foreign keys needs foreign_key_checks=OFF. Try ALGORITHM=

```

But this succeeds:

```

CREATE OR REPLACE TABLE tab1 (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  d int
);

CREATE OR REPLACE TABLE tab2 (
  a int PRIMARY KEY,
  b varchar(50)
);

SET SESSION foreign_key_checks=OFF;
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
Query OK, 0 rows affected (0.011 sec)

```

This applies to [ALTER TABLE ... ADD FOREIGN KEY](#) for InnoDB tables.

## ALTER TABLE ... DROP FOREIGN KEY

InnoDB supports dropping foreign key constraints from a table with [ALGORITHM](#) set to [INPLACE](#).

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to [NONE](#). When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab2 (
  a int PRIMARY KEY,
  b varchar(50)
);

CREATE OR REPLACE TABLE tab1 (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  d int,
  FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab1 DROP FOREIGN KEY tab2_fk;
Query OK, 0 rows affected (0.005 sec)

```

This applies to [ALTER TABLE ... DROP FOREIGN KEY](#) for InnoDB tables.

# Table Operations

## ALTER TABLE ... AUTO\_INCREMENT=...

InnoDB supports changing a table's `AUTO_INCREMENT` value with `ALGORITHM` set to

`INPLACE`

. This operation should finish instantly. The table is not rebuilt.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

`NONE`

. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab AUTO_INCREMENT=100;
Query OK, 0 rows affected (0.004 sec)
```

This applies to `ALTER TABLE ... AUTO_INCREMENT=...` for InnoDB tables.

## ALTER TABLE ... ROW\_FORMAT=...

InnoDB supports changing a table's `row format` with `ALGORITHM` set to

`INPLACE`

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

`NONE`

. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
) ROW_FORMAT=DYNAMIC;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ROW_FORMAT=COMPRESSED;
Query OK, 0 rows affected (0.025 sec)
```

This applies to `ALTER TABLE ... ROW_FORMAT=...` for InnoDB tables.

## ALTER TABLE ... KEY\_BLOCK\_SIZE=...

InnoDB supports changing a table's `KEY_BLOCK_SIZE` with `ALGORITHM` set to

`INPLACE`

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

`NONE`

. When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) ROW_FORMAT=COMPRESSED
KEY_BLOCK_SIZE=4;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab KEY_BLOCK_SIZE=2;
Query OK, 0 rows affected (0.021 sec)

```

This applies to [KEY\\_BLOCK\\_SIZE=...](#) for InnoDB tables.

ALTER TABLE ... PAGE\_COMPRESSED=...  
 and  
 ALTER TABLE ... PAGE\_COMPRESSION\_LEVEL=...

In MariaDB 10.3.10 and later, InnoDB supports setting a table's [PAGE\\_COMPRESSED](#) value to

```

1
with ALGORITHM set to
INPLACE
. InnoDB also supports changing a table's PAGE\_COMPRESSED value from
1
to
0
with ALGORITHM set to
INPLACE
.

```

In these versions, InnoDB also supports changing a table's [PAGE\\_COMPRESSION\\_LEVEL](#) value with [ALGORITHM](#) set to  
 INPLACE

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

```

NONE
. When this strategy is used, all concurrent DML is permitted.

```

See [MDEV-16328](#) for more information.

For example, this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab PAGE_COMPRESSED=1;
Query OK, 0 rows affected (0.006 sec)

```

And this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) PAGE_COMPRESSED=1;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab PAGE_COMPRESSED=0;
Query OK, 0 rows affected (0.020 sec)

```

And this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) PAGE_COMPRESSED=1
PAGE_COMPRESSION_LEVEL=5;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab PAGE_COMPRESSION_LEVEL=4;
Query OK, 0 rows affected (0.006 sec)

```

This applies to [PAGE\\_COMPRESSED=...](#) and [PAGE\\_COMPRESSION\\_LEVEL=...](#) for InnoDB tables.

## ALTER TABLE ... DROP SYSTEM VERSIONING

InnoDB supports dropping [system versioning](#) from a table with [ALGORITHM](#) set to [INPLACE](#)

This operation supports the read-only locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to [SHARED](#). When this strategy is used, read-only concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) WITH SYSTEM VERSIONING;

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP SYSTEM VERSIONING;

```

This applies to [ALTER TABLE ... DROP SYSTEM VERSIONING](#) for InnoDB tables.

## ALTER TABLE ... DROP CONSTRAINT

In MariaDB 10.3.6 and later, InnoDB supports dropping a [CHECK](#) constraint from a table with [ALGORITHM](#) set to [INPLACE](#). See [MDEV-16331](#) for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to [NONE](#). When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  CONSTRAINT b_not_empty CHECK (b != '')
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab DROP CONSTRAINT b_not_empty;
Query OK, 0 rows affected (0.004 sec)

```

This applies to [ALTER TABLE ... DROP CONSTRAINT](#) for InnoDB tables.

## ALTER TABLE ... FORCE

InnoDB supports forcing a table rebuild with [ALGORITHM](#) set to

INPLACE

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to  
NONE  
. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab FORCE;
Query OK, 0 rows affected (0.022 sec)
```

This applies to [ALTER TABLE ... FORCE](#) for InnoDB tables.

## ALTER TABLE ... ENGINE=InnoDB

InnoDB supports forcing a table rebuild with [ALGORITHM](#) set to

INPLACE

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to  
NONE  
. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ENGINE=InnoDB;
Query OK, 0 rows affected (0.022 sec)
```

This applies to [ALTER TABLE ... ENGINE=InnoDB](#) for InnoDB tables.

## OPTIMIZE TABLE ...

InnoDB supports optimizing a table with [ALGORITHM](#) set to

INPLACE

If the [innodb\\_defragment](#) system variable is set to

OFF

, and if the [innodb\\_optimize\\_fulltext\\_only](#) system variable is also set to

OFF

, then

[OPTIMIZE TABLE](#)

will be equivalent to

[ALTER TABLE ... FORCE](#)

The table is rebuilt, which means that all of the data is reorganized substantially, and the indexes are rebuilt. As a result, the operation is quite expensive.

If either of the previously mentioned system variables is set to

ON

, then

```
OPTIMIZE TABLE
```

will optimize some data without rebuilding the table. However, the file size will not be reduced.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SHOW GLOBAL VARIABLES WHERE Variable_name IN('innodb_defragment', 'innodb_optimize_fulltext_only');
+-----+-----+
| Variable_name          | Value   |
+-----+-----+
| innodb_defragment      | OFF     |
| innodb_optimize_fulltext_only | OFF     |
+-----+-----+

SET SESSION alter_algorithm='INPLACE';
OPTIMIZE TABLE tab;
+-----+-----+-----+
| Table  | Op       | Msg_type | Msg_text           |
+-----+-----+-----+
| db1.tab | optimize | note      | Table does not support optimize, doing recreate + analyze instead |
| db1.tab | optimize | status     | OK                |
+-----+-----+-----+
2 rows in set (0.026 sec)
```

And this succeeds, but the table is not rebuilt:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET GLOBAL innodb_defragment=ON;
SHOW GLOBAL VARIABLES WHERE Variable_name IN('innodb_defragment', 'innodb_optimize_fulltext_only');
+-----+-----+
| Variable_name          | Value   |
+-----+-----+
| innodb_defragment      | ON      |
| innodb_optimize_fulltext_only | OFF     |
+-----+-----+

SET SESSION alter_algorithm='INPLACE';
OPTIMIZE TABLE tab;
+-----+-----+-----+
| Table  | Op       | Msg_type | Msg_text           |
+-----+-----+-----+
| db1.tab | optimize | status    | OK                |
+-----+-----+-----+
1 row in set (0.004 sec)
```

This applies to [OPTIMIZE TABLE](#) for InnoDB tables.

ALTER TABLE ... RENAME TO  
and  
RENAME TABLE ...

InnoDB supports renaming a table with [ALGORITHM](#) set to

INPLACE

This operation only changes the table's metadata, so the table does not have to be rebuilt.

This operation supports the exclusive locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to EXCLUSIVE . When this strategy is used, concurrent DML is **not** permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab RENAME TO old_tab;
Query OK, 0 rows affected (0.011 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
RENAME TABLE tab TO old_tab;
```

This applies to [ALTER TABLE ... RENAME TO](#) and [RENAME TABLE](#) for InnoDB tables.

## Limitations

### Limitations Related to Fulltext Indexes

- If a table has more than one [FULLTEXT](#) index, then it cannot be rebuilt by any [ALTER TABLE](#) operations when [ALGORITHM](#) is set to [INPLACE](#)
- If a table has a [FULLTEXT](#) index, then it cannot be rebuilt by any [ALTER TABLE](#) operations when the [LOCK](#) clause is set to [NONE](#)

### Limitations Related to Spatial Indexes

- If a table has a [SPATIAL](#) index, then it cannot be rebuilt by any [ALTER TABLE](#) operations when the [LOCK](#) clause is set to [NONE](#)

### Limitations Related to Generated (Virtual and Persistent/Stored) Columns

[Generated columns](#) do not currently support online DDL for all of the same operations that are supported for "real" columns.

See [Generated \(Virtual and Persistent/Stored\) Columns: Statement Support](#) for more information on the limitations.

## 4.3.2.19.3 InnoDB Online DDL Operations with the NOCOPY Alter Algorithm

## Contents

1. Supported Operations by Inheritance
2. Column Operations
  1. `ALTER TABLE ... ADD COLUMN`
  2. `ALTER TABLE ... DROP COLUMN`
  3. `ALTER TABLE ... MODIFY COLUMN`
    1. Reordering Columns
    2. Changing the Data Type of a Column
    3. Changing a Column to NULL
    4. Changing a Column to NOT NULL
    5. Adding a New ENUM Option
    6. Adding a New SET Option
    7. Removing System Versioning from a Column
  4. `ALTER TABLE ... ALTER COLUMN`
    1. Setting a Column's Default Value
    2. Removing a Column's Default Value
  5. `ALTER TABLE ... CHANGE COLUMN`
3. Index Operations
  1. `ALTER TABLE ... ADD PRIMARY KEY`
  2. `ALTER TABLE ... DROP PRIMARY KEY`
  3. `ALTER TABLE ... ADD INDEX` and `CREATE INDEX`
    1. Adding a Plain Index
    2. Adding a Fulltext Index
    3. Adding a Spatial Index
  4. `ALTER TABLE ... DROP INDEX` and `DROP INDEX`
  5. `ALTER TABLE ... ADD FOREIGN KEY`
  6. `ALTER TABLE ... DROP FOREIGN KEY`
4. Table Operations
  1. `ALTER TABLE ... AUTO_INCREMENT=...`
  2. `ALTER TABLE ... ROW_FORMAT=...`
  3. `ALTER TABLE ... KEY_BLOCK_SIZE=...`
  4. `ALTER TABLE ... PAGE_COMPRESSED=1` and `ALTER TABLE ... PAGE_COMPRESSION_LEVEL=...`
  5. `ALTER TABLE ... DROP SYSTEM VERSIONING`
  6. `ALTER TABLE ... DROP CONSTRAINT`
  7. `ALTER TABLE ... FORCE`
  8. `ALTER TABLE ... ENGINE=InnoDB`
  9. `OPTIMIZE TABLE ...`
  10. `ALTER TABLE ... RENAME TO` and `RENAME TABLE ...`
5. Limitations
  1. Limitations Related to Generated (Virtual and Persistent/Stored) Columns

## Supported Operations by Inheritance

When the `ALGORITHM` clause is set to

`NOCOPY`

, the supported operations are a superset of the operations that are supported when the `ALGORITHM` clause is set to `INSTANT`

Therefore, when the `ALGORITHM` clause is set to

`NOCOPY`

, some operations are supported by inheritance. See the following additional pages for more information about these supported operations:

- [InnoDB Online DDL Operations with `ALGORITHM=INSTANT`](#)

# Column Operations

## ALTER TABLE ... ADD COLUMN

In MariaDB 10.3.2 and later, InnoDB supports adding columns to a table with `ALGORITHM` set to  
NOCOPY  
in the cases where the operation supports having the `ALGORITHM` clause set to  
INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... ADD COLUMN](#) for more information.

This applies to `ALTER TABLE ... ADD COLUMN` for InnoDB tables.

## ALTER TABLE ... DROP COLUMN

In MariaDB 10.4 and later, InnoDB supports dropping columns from a table with `ALGORITHM` set to  
NOCOPY  
in the cases where the operation supports having the `ALGORITHM` clause set to  
INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... DROP COLUMN](#) for more information.

This applies to `ALTER TABLE ... DROP COLUMN` for InnoDB tables.

## ALTER TABLE ... MODIFY COLUMN

This applies to `ALTER TABLE ... MODIFY COLUMN` for InnoDB tables.

### Reordering Columns

In MariaDB 10.4 and later, InnoDB supports reordering columns within a table with `ALGORITHM` set to  
NOCOPY  
in the cases where the operation supports having the `ALGORITHM` clause set to  
INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Reordering Columns](#) for more information.

### Changing the Data Type of a Column

InnoDB does **not** support modifying a column's data type with `ALGORITHM` set to  
NOCOPY  
in most cases. There are a few exceptions in the cases where the operation supports having the `ALGORITHM` clause set to  
INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Changing the Data Type of a Column](#) for more information.

### Changing a Column to NULL

In MariaDB 10.4.3 and later, InnoDB supports modifying a column to allow `NULL` values with `ALGORITHM` set to  
NOCOPY  
in the cases where the operation supports having the `ALGORITHM` clause set to  
INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Changing a Column to NULL](#) for more information.

### Changing a Column to NOT NULL

InnoDB does **not** support modifying a column to **not** allow `NULL` values with `ALGORITHM` set to  
NOCOPY

For example:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) ROW_FORMAT=REDUNDANT;

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab MODIFY COLUMN c varchar(50) NOT NULL;
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE
```

## Adding a New ENUM Option

InnoDB supports adding a new [ENUM](#) option to a column with [ALGORITHM](#) set to

NOCOPY

in the cases where the operation supports having the [ALGORITHM](#) clause set to

INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Adding a New ENUM Option](#) for more information.

## Adding a New SET Option

InnoDB supports adding a new [SET](#) option to a column with [ALGORITHM](#) set to

NOCOPY

in the cases where the operation supports having the [ALGORITHM](#) clause set to

INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Adding a New SET Option](#) for more information.

## Removing System Versioning from a Column

In [MariaDB 10.3.8](#) and later, InnoDB supports removing [system](#) versioning from a column with [ALGORITHM](#) set to

NOCOPY

in the cases where the operation supports having the [ALGORITHM](#) clause set to

INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Removing System Versioning from a Column](#) for more information.

## ALTER TABLE ... ALTER COLUMN

This applies to [ALTER TABLE ... ALTER COLUMN](#) for InnoDB tables.

### Setting a Column's Default Value

InnoDB supports modifying a column's [DEFAULT](#) value with [ALGORITHM](#) set to

NOCOPY

in the cases where the operation supports having the [ALGORITHM](#) clause set to

INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Setting a Column's Default Value](#) for more information.

### Removing a Column's Default Value

InnoDB supports removing a column's [DEFAULT](#) value with [ALGORITHM](#) set to

NOCOPY

in the cases where the operation supports having the [ALGORITHM](#) clause set to

INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Removing a Column's Default Value](#) for more information.

## ALTER TABLE ... CHANGE COLUMN

InnoDB supports renaming a column with `ALGORITHM` set to

`NOCOPY`

in the cases where the operation supports having the `ALGORITHM` clause set to

`INSTANT`

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... CHANGE COLUMN](#) for more information.

This applies to `ALTER TABLE ... CHANGE COLUMN` for InnoDB tables.

## Index Operations

### ALTER TABLE ... ADD PRIMARY KEY

InnoDB does **not** support adding a primary key to a table with `ALGORITHM` set to

`NOCOPY`

For example:

```
CREATE OR REPLACE TABLE tab (
    a int,
    b varchar(50),
    c varchar(50)
);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD PRIMARY KEY (a);
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to `ALTER TABLE ... ADD PRIMARY KEY` for InnoDB tables.

### ALTER TABLE ... DROP PRIMARY KEY

InnoDB does **not** support dropping a primary key with `ALGORITHM` set to

`NOCOPY`

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab DROP PRIMARY KEY;
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Dropping a primary key is not allowed without also adding a new p
```

This applies to `ALTER TABLE ... DROP PRIMARY KEY` for InnoDB tables.

### ALTER TABLE ... ADD INDEX and CREATE INDEX

This applies to `ALTER TABLE ... ADD INDEX` and `CREATE INDEX` for InnoDB tables.

## Adding a Plain Index

InnoDB supports adding a plain index to a table with `ALGORITHM` set to  
`NOCOPY`

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to  
`NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD INDEX b_index (b);
Query OK, 0 rows affected (0.009 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='NOCOPY';
CREATE INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.009 sec)
```

## Adding a Fulltext Index

InnoDB supports adding a `FULLTEXT` index to a table with `ALGORITHM` set to  
`NOCOPY`

However, there are some limitations, such as:

- Adding a `FULLTEXT` index to a table that does not have a user-defined  
`FTS_DOC_ID` column will require the table to be rebuilt once. When the table is rebuilt, the system adds a hidden  
`FTS_DOC_ID` column. This initial operation will have to be performed with `ALGORITHM` set to  
`INPLACE`. From that point forward, adding additional `FULLTEXT` indexes to the same table will not require the table to be rebuilt, and `ALGORITHM` can be set to  
`NOCOPY`
- Only one `FULLTEXT` index may be added at a time when `ALGORITHM` is set to  
`NOCOPY`

This operation supports a read-only locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to  
`SHARED`. When this strategy is used, read-only concurrent DML is permitted.

For example, this succeeds, but the first operation requires the table to be rebuilt `ALGORITHM` set to  
`INPLACE`, so that the hidden  
`FTS_DOC_ID` column can be added:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.043 sec)

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD FULLTEXT INDEX c_index (c);
Query OK, 0 rows affected (0.017 sec)

```

And this succeeds in the same way as above:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
CREATE FULLTEXT INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.048 sec)

SET SESSION alter_algorithm='NOCOPY';
CREATE FULLTEXT INDEX c_index ON tab (c);
Query OK, 0 rows affected (0.016 sec)

```

But this second command fails, because only one **FULLTEXT** index can be added at a time:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  d varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.041 sec)

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD FULLTEXT INDEX c_index (c), ADD FULLTEXT INDEX d_index (d);
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: InnoDB presently supports one FULLTEXT index creation at a time.

```

## Adding a Spatial Index

InnoDB supports adding a **SPATIAL** index to a table with **ALGORITHM** set to  
NOCOPY

This operation supports a read-only locking strategy. This strategy can be explicitly chosen by setting the **LOCK** clause to  
SHARED  
. When this strategy is used, read-only concurrent DML is permitted.

For example, this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ADD SPATIAL INDEX c_index (c);
Query OK, 0 rows affected (0.005 sec)

```

And this succeeds in the same way as above:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='NOCOPY';
CREATE SPATIAL INDEX c_index ON tab (c);
Query OK, 0 rows affected (0.005 sec)

```

## ALTER TABLE ... DROP INDEX and DROP INDEX

InnoDB supports dropping indexes from a table with [ALGORITHM](#) set to

- NOCOPY
- in the cases where the operation supports having the [ALGORITHM](#) clause set to
- INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... DROP INDEX and DROP INDEX](#) for more information.

This applies to [ALTER TABLE ... DROP INDEX](#) and [DROP INDEX](#) for InnoDB tables.

## ALTER TABLE ... ADD FOREIGN KEY

InnoDB does supports adding foreign key constraints to a table with [ALGORITHM](#) set to

- NOCOPY
- . In order to add a new foreign key constraint to a table with [ALGORITHM](#) set to
- NOCOPY
- , the [foreign\\_key\\_checks](#) system variable needs to be set to
- OFF
- . If it is set to
- ON
- , then
- [ALGORITHM=COPY](#)
- is required.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

- NONE
- . When this strategy is used, all concurrent DML is permitted.

For example, this fails:

```

CREATE OR REPLACE TABLE tab1 (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  d int
);

CREATE OR REPLACE TABLE tab2 (
  a int PRIMARY KEY,
  b varchar(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Adding foreign keys needs foreign_key_checks=OFF. Try ALGORITHM=C

```

But this succeeds:

```

CREATE OR REPLACE TABLE tab1 (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  d int
);

CREATE OR REPLACE TABLE tab2 (
  a int PRIMARY KEY,
  b varchar(50)
);

SET SESSION foreign_key_checks=OFF;
SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
Query OK, 0 rows affected (0.011 sec)

```

This applies to [ALTER TABLE ... ADD FOREIGN KEY](#) for InnoDB tables.

## ALTER TABLE ... DROP FOREIGN KEY

InnoDB supports dropping foreign key constraints from a table with [ALGORITHM](#) set to  
 NOCOPY  
 in the cases where the operation supports having the [ALGORITHM](#) clause set to  
 INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... DROP FOREIGN KEY](#) for more information.

This applies to [ALTER TABLE ... DROP FOREIGN KEY](#) for InnoDB tables.

## Table Operations

### ALTER TABLE ... AUTO\_INCREMENT=...

InnoDB supports changing a table's [AUTO\\_INCREMENT](#) value with [ALGORITHM](#) set to  
 NOCOPY  
 in the cases where the operation supports having the [ALGORITHM](#) clause set to  
 INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... AUTO\\_INCREMENT=...](#) for more information.

This applies to [ALTER TABLE ... AUTO\\_INCREMENT=...](#) for InnoDB tables.

### ALTER TABLE ... ROW\_FORMAT=...

InnoDB does **not** support changing a table's [row format](#) with [ALGORITHM](#) set to  
 NOCOPY

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) ROW_FORMAT=DYNAMIC;

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ROW_FORMAT=COMPRESSED;
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Changing table options requires the table to be rebuilt. Try ALGO

```

This applies to [ALTER TABLE ... ROW\\_FORMAT=...](#) for InnoDB tables.

## ALTER TABLE ... KEY\_BLOCK\_SIZE=...

InnoDB does **not** support changing a table's `KEY_BLOCK_SIZE` with `ALGORITHM` set to  
NOCOPY

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
) ROW_FORMAT=COMPRESSED
KEY_BLOCK_SIZE=4;

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab KEY_BLOCK_SIZE=2;
ERROR 1846 (0A000): ALGORITHM=NOCOPY is not supported. Reason: Changing table options requires the table to be rebuilt. Try ALGO
```

This applies to `KEY_BLOCK_SIZE=...` for InnoDB tables.

## ALTER TABLE ... PAGE\_COMPRESSED=1

and

## ALTER TABLE ... PAGE\_COMPRESSION\_LEVEL=...

In MariaDB 10.3.10 and later, InnoDB supports setting a table's `PAGE_COMPRESSED` value to

1  
with `ALGORITHM` set to  
NOCOPY

in the cases where the operation supports having the `ALGORITHM` clause set to  
INSTANT

InnoDB does **not** support changing a table's `PAGE_COMPRESSED` value from

1  
to  
0  
with `ALGORITHM` set to  
NOCOPY

In these versions, InnoDB also supports changing a table's `PAGE_COMPRESSION_LEVEL` value with `ALGORITHM` set to

NOCOPY  
in the cases where the operation supports having the `ALGORITHM` clause is set to  
INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... PAGE\\_COMPRESSED=1 and ALTER TABLE ... PAGE\\_COMPRESSION\\_LEVEL=...](#) for more information.

This applies to `ALTER TABLE ... PAGE_COMPRESSED=...` and `ALTER TABLE ... PAGE_COMPRESSION_LEVEL=...` for InnoDB tables.

## ALTER TABLE ... DROP SYSTEM VERSIONING

InnoDB does **not** support dropping `system versioning` from a table with `ALGORITHM` set to  
NOCOPY

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) WITH SYSTEM VERSIONING;

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab DROP SYSTEM VERSIONING;
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE

```

This applies to [ALTER TABLE ... DROP SYSTEM VERSIONING](#) for InnoDB tables.

## ALTER TABLE ... DROP CONSTRAINT

In MariaDB 10.3.6 and later, InnoDB supports dropping a [CHECK](#) constraint from a table with [ALGORITHM](#) set to  
NOCOPY  
in the cases where the operation supports having the [ALGORITHM](#) clause set to  
INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... DROP CONSTRAINT](#) for more information.

This applies to [ALTER TABLE ... DROP CONSTRAINT](#) for InnoDB tables.

## ALTER TABLE ... FORCE

InnoDB does **not** support forcing a table rebuild with [ALGORITHM](#) set to  
NOCOPY

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab FORCE;
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE

```

This applies to [ALTER TABLE ... FORCE](#) for InnoDB tables.

## ALTER TABLE ... ENGINE=InnoDB

InnoDB does **not** support forcing a table rebuild with [ALGORITHM](#) set to  
NOCOPY

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='NOCOPY';
ALTER TABLE tab ENGINE=InnoDB;
ERROR 1845 (0A000): ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE

```

This applies to [ALTER TABLE ... ENGINE=InnoDB](#) for InnoDB tables.

## OPTIMIZE TABLE ...

InnoDB does **not** support optimizing a table with with [ALGORITHM](#) set to  
NOCOPY

For example:

```
CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SHOW GLOBAL VARIABLES WHERE Variable_name IN('innodb_defragment', 'innodb_optimize_fulltext_only');
+-----+-----+
| Variable_name      | Value |
+-----+-----+
| innodb_defragment | OFF   |
| innodb_optimize_fulltext_only | OFF   |
+-----+-----+
2 rows in set (0.001 sec)

SET SESSION alter_algorithm='NOCOPY';
OPTIMIZE TABLE tab;
+-----+-----+-----+
| Table  | Op       | Msg_type | Msg_text
+-----+-----+-----+
| db1.tab | optimize | note     | Table does not support optimize, doing recreate + analyze instead
| db1.tab | optimize | error    | ALGORITHM=NOCOPY is not supported for this operation. Try ALGORITHM=INPLACE
| db1.tab | optimize | status   | Operation failed
+-----+-----+-----+
3 rows in set, 1 warning (0.002 sec)
```

This applies to [OPTIMIZE TABLE](#) for InnoDB tables.

ALTER TABLE ... RENAME TO  
and  
RENAME TABLE ...

InnoDB supports renaming a table with [ALGORITHM](#) set to  
NOCOPY  
in the cases where the operation supports having the [ALGORITHM](#) clause set to  
INSTANT

See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: ALTER TABLE ... RENAME TO and RENAME TABLE ...](#) for more information.

This applies to [ALTER TABLE ... RENAME TO](#) and [RENAME TABLE](#) for InnoDB tables.

## Limitations

### Limitations Related to Generated (Virtual and Persistent/Stored) Columns

[Generated columns](#) do not currently support online DDL for all of the same operations that are supported for "real" columns.

See [Generated \(Virtual and Persistent/Stored\) Columns: Statement Support](#) for more information on the limitations.

### 4.3.2.19.4 InnoDB Online DDL Operations with the INSTANT Alter Algorithm

## Contents

- 1. Column Operations
  - 1. [ALTER TABLE ... ADD COLUMN](#)
  - 2. [ALTER TABLE ... DROP COLUMN](#)
  - 3. [ALTER TABLE ... MODIFY COLUMN](#)
    - 1. Reordering Columns
    - 2. Changing the Data Type of a Column
    - 3. Changing a Column to NULL
    - 4. Changing a Column to NOT NULL
    - 5. Adding a New ENUM Option
    - 6. Adding a New SET Option
    - 7. Removing System Versioning from a Column
  - 4. [ALTER TABLE ... ALTER COLUMN](#)
    - 1. Setting a Column's Default Value
    - 2. Removing a Column's Default Value
  - 5. [ALTER TABLE ... CHANGE COLUMN](#)
- 2. Index Operations
  - 1. [ALTER TABLE ... ADD PRIMARY KEY](#)
  - 2. [ALTER TABLE ... DROP PRIMARY KEY](#)
  - 3. [ALTER TABLE ... ADD INDEX and CREATE INDEX](#)
    - 1. Adding a Plain Index
    - 2. Adding a Fulltext Index
    - 3. Adding a Spatial Index
  - 4. [ALTER TABLE ... DROP INDEX and DROP INDEX](#)
  - 5. [ALTER TABLE ... ADD FOREIGN KEY](#)
  - 6. [ALTER TABLE ... DROP FOREIGN KEY](#)
- 3. Table Operations
  - 1. [ALTER TABLE ... AUTO\\_INCREMENT=...](#)
  - 2. [ALTER TABLE ... ROW\\_FORMAT=...](#)
  - 3. [ALTER TABLE ... KEY\\_BLOCK\\_SIZE=...](#)
  - 4. [ALTER TABLE ... PAGE\\_COMPRESSED=1 and ALTER TABLE ... PAGE\\_COMPRESSION\\_LEVEL=...](#)
  - 5. [ALTER TABLE ... DROP SYSTEM VERSIONING](#)
  - 6. [ALTER TABLE ... DROP CONSTRAINT](#)
  - 7. [ALTER TABLE ... FORCE](#)
  - 8. [ALTER TABLE ... ENGINE=InnoDB](#)
  - 9. [OPTIMIZE TABLE ...](#)
  - 10. [ALTER TABLE ... RENAME TO and RENAME TABLE ...](#)
- 4. Limitations
  - 1. Limitations Related to Generated (Virtual and Persistent/Stored) Columns
  - 2. Non-canonical Storage Format Caused by Some Operations
  - 3. Known Bugs
    - 1. [Closed Bugs](#)
    - 2. [Open Bugs](#)

## Column Operations

### ALTER TABLE ... ADD COLUMN

In MariaDB 10.3.2 and later, InnoDB supports adding columns to a table with [ALGORITHM](#) set to

INSTANT

if the new column is the last column in the table. See [MDEV-11369](#) for more information. If the table has a hidden

`FTS_DOC_ID`  
column is present, then this is not supported.

In MariaDB 10.4 and later, InnoDB supports adding columns to a table with `ALGORITHM` set to

`INSTANT`  
, regardless of where in the column list the new column is added.

When this operation is performed with `ALGORITHM` set to

`INSTANT`  
, the tablespace file will have a non-canonical storage format. See [Non-canonical Storage Format Caused by Some Operations](#) for more information.

With the exception of adding an `auto-increment` column, this operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

`NONE`  
. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD COLUMN c varchar(50);
Query OK, 0 rows affected (0.004 sec)
```

And this succeeds in MariaDB 10.4 and later:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD COLUMN c varchar(50) AFTER a;
Query OK, 0 rows affected (0.004 sec)
```

This applies to [ALTER TABLE ... ADD COLUMN](#) for InnoDB tables.

See [Instant ADD COLUMN](#) for InnoDB for more information.

## ALTER TABLE ... DROP COLUMN

In MariaDB 10.4 and later, InnoDB supports dropping columns from a table with `ALGORITHM` set to

`INSTANT`  
. See [MDEV-15562](#) for more information.

When this operation is performed with `ALGORITHM` set to

`INSTANT`  
, the tablespace file will have a non-canonical storage format. See [Non-canonical Storage Format Caused by Some Operations](#) for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

`NONE`  
. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP COLUMN c;
Query OK, 0 rows affected (0.004 sec)
```

This applies to [ALTER TABLE ... DROP COLUMN](#) for InnoDB tables.

## ALTER TABLE ... MODIFY COLUMN

This applies to [ALTER TABLE ... MODIFY COLUMN](#) for InnoDB tables.

### Reordering Columns

In MariaDB 10.4 and later, InnoDB supports reordering columns within a table with [ALGORITHM](#) set to

INSTANT

. See [MDEV-15562](#) for more information.

When this operation is performed with [ALGORITHM](#) set to

INSTANT

, the tablespace file will have a non-canonical storage format. See [Non-canonical Storage Format Caused by Some Operations](#) for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(50) AFTER a;
Query OK, 0 rows affected (0.004 sec)
```

### Changing the Data Type of a Column

InnoDB does **not** support modifying a column's data type with [ALGORITHM](#) set to

INSTANT

in most cases. There are some exceptions:

- InnoDB supports increasing the length of

VARCHAR

columns with [ALGORITHM](#) set to

INSTANT

, unless it would require changing the number of bytes required to represent the column's length. A

VARCHAR

column that is between 0 and 255 bytes in size requires 1 byte to represent its length, while a

VARCHAR

column that is 256 bytes or longer requires 2 bytes to represent its length. This means that the length of a column cannot be increased with [ALGORITHM](#) set to

INSTANT

if the original length was less than 256 bytes, and the new length is 256 bytes or more.

- In MariaDB 10.4.3 and later, InnoDB supports increasing the length of

VARCHAR

columns with [ALGORITHM](#) set to

INSTANT

with no restrictions if the [ROW\\_FORMAT](#) table option is set to [REDUNDANT](#). See [MDEV-15563](#) for more information.

- In MariaDB 10.4.3 and later, InnoDB also supports increasing the length of

VARCHAR

columns with [ALGORITHM](#) set to

INSTANT

in a more limited manner if the [ROW\\_FORMAT](#) table option is set to [COMPACT](#), [DYNAMIC](#), or [COMPRESSED](#). In this scenario, the following limitations apply:

- The length can be increased with [ALGORITHM](#) set to

INSTANT

if the original length of the column is 127 bytes or less, and the new length of the column is 256 bytes or more.

- The length can be increased with [ALGORITHM](#) set to

INSTANT

if the original length of the column is 255 bytes or less, and the new length of the column is still 255 bytes or less.

- The length can be increased with [ALGORITHM](#) set to

INSTANT

if the original length of the column is 256 bytes or more, and the new length of the column is still 256 bytes or more.

- The length can **not** be increased with [ALGORITHM](#) set to

INSTANT  
if the original length was between 128 bytes and 255 bytes, and the new length is 256 bytes or more.  
o See [MDEV-15563](#) for more information.

The supported operations in this category support the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c int;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

But this succeeds because the original length of the column is less than 256 bytes, and the new length is still less than 256 bytes:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
) CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(100);
Query OK, 0 rows affected (0.005 sec)
```

But this fails because the original length of the column is between 128 bytes and 255 bytes, and the new length is greater than 256 bytes:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(255)
) CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(256);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

But this succeeds in [MariaDB 10.4.3](#) and later because the table has

```
ROW_FORMAT=REDUNDANT
:
```

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(200)
) ROW_FORMAT=REDUNDANT;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(300);
Query OK, 0 rows affected (0.004 sec)
```

And this succeeds in [MariaDB 10.4.3](#) and later because the table has

```
ROW_FORMAT=DYNAMIC
```

and the column's original length is 127 bytes or less:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(127)
) ROW_FORMAT=DYNAMIC
CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(300);
Query OK, 0 rows affected (0.003 sec)
```

And this succeeds in [MariaDB 10.4.3](#) and later because the table has

ROW\_FORMAT=COMPRESSED

and the column's original length is 127 bytes or less:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(127)
) ROW_FORMAT=COMPRESSED
CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(300);
Query OK, 0 rows affected (0.003 sec)
```

But this fails even in [MariaDB 10.4.3](#) and later because the table has

ROW\_FORMAT=DYNAMIC

and the column's original length is between 128 bytes and 255 bytes:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(128)
) ROW_FORMAT=DYNAMIC
CHARACTER SET=latin1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(300);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

## Changing a Column to NULL

In [MariaDB 10.4.3](#) and later, InnoDB supports modifying a column to allow **NULL** values with **ALGORITHM** set to

INSTANT

if the **ROW\_FORMAT** table option is set to **REDUNDANT**. See [MDEV-15563](#) for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the **LOCK** clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50) NOT NULL
) ROW_FORMAT=REDUNDANT;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(50) NULL;
Query OK, 0 rows affected (0.004 sec)
```

## Changing a Column to NOT NULL

InnoDB does **not** support modifying a column to **not** allow **NULL** values with **ALGORITHM** set to

INSTANT

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
) ROW_FORMAT=REDUNDANT;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(50) NOT NULL;
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE
```

## Adding a New ENUM Option

InnoDB supports adding a new **ENUM** option to a column with **ALGORITHM** set to

INSTANT

. In order to add a new **ENUM** option with **ALGORITHM** set to

INSTANT

, the following requirements must be met:

- It must be added to the end of the list.
- The storage requirements must not change.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the **LOCK** clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c ENUM('red', 'green')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c ENUM('red', 'green', 'blue');
Query OK, 0 rows affected (0.002 sec)
```

But this fails:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c ENUM('red', 'green')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c ENUM('red', 'blue', 'green');
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY
```

## Adding a New

SET

Option

InnoDB supports adding a new **SET** option to a column with **ALGORITHM** set to

INSTANT

. In order to add a new **SET** option with **ALGORITHM** set to

INSTANT

, the following requirements must be met:

- It must be added to the end of the list.
- The storage requirements must not change.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the **LOCK** clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c SET('red', 'green')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c SET('red', 'green', 'blue');
Query OK, 0 rows affected (0.002 sec)
```

But this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c SET('red', 'green')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c SET('red', 'blue', 'green');
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY

```

## Removing System Versioning from a Column

In MariaDB 10.3.8 and later, InnoDB supports removing system versioning from a column with `ALGORITHM` set to

- INSTANT
  - . In order for this to work, the `system_versioning_alter_history` system variable must be set to `KEEP`
  - . See [MDEV-16330](#) for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

- NONE
  - . When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50) WITH SYSTEM VERSIONING
);

SET SESSION system_versioning_alter_history='KEEP';
SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab MODIFY COLUMN c varchar(50) WITHOUT SYSTEM VERSIONING;
Query OK, 0 rows affected (0.004 sec)

```

## ALTER TABLE ... ALTER COLUMN

This applies to `ALTER TABLE ... ALTER COLUMN` for InnoDB tables.

### Setting a Column's Default Value

InnoDB supports modifying a column's `DEFAULT` value with `ALGORITHM` set to

- INSTANT
  - .

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

- NONE
  - . When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ALTER COLUMN c SET DEFAULT 'No value explicitly provided.';
Query OK, 0 rows affected (0.003 sec)

```

### Removing a Column's Default Value

InnoDB supports removing a column's `DEFAULT` value with `ALGORITHM` set to

- INSTANT
  - .

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

- NONE
  - . When this strategy is used, all concurrent DML is permitted.

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50) DEFAULT 'No value explicitly provided.'
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ALTER COLUMN c DROP DEFAULT;
Query OK, 0 rows affected (0.002 sec)

```

## ALTER TABLE ... CHANGE COLUMN

InnoDB supports renaming a column with [ALGORITHM](#) set to

INSTANT  
, unless the column's data type or attributes changed in addition to the name.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

NONE  
. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab CHANGE COLUMN c str varchar(50);
Query OK, 0 rows affected (0.004 sec)

```

But this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab CHANGE COLUMN c num int;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Cannot change column type INPLACE. Try ALGORITHM=COPY

```

This applies to [ALTER TABLE ... CHANGE COLUMN](#) for InnoDB tables.

## Index Operations

### ALTER TABLE ... ADD PRIMARY KEY

InnoDB does **not** support adding a primary key to a table with [ALGORITHM](#) set to

INSTANT

For example:

```

CREATE OR REPLACE TABLE tab (
  a int,
  b varchar(50),
  c varchar(50)
);

SET SESSION sql_mode='STRICT_TRANS_TABLES';
SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD PRIMARY KEY (a);
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE

```

This applies to [ALTER TABLE ... ADD PRIMARY KEY](#) for InnoDB tables.

## ALTER TABLE ... DROP PRIMARY KEY

InnoDB does **not** support dropping a primary key with [ALGORITHM](#) set to  
INSTANT

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP PRIMARY KEY;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Dropping a primary key is not allowed without also adding a new
```

This applies to [ALTER TABLE ... DROP PRIMARY KEY](#) for InnoDB tables.

## ALTER TABLE ... ADD INDEX and CREATE INDEX

This applies to [ALTER TABLE ... ADD INDEX](#) and [CREATE INDEX](#) for InnoDB tables.

### Adding a Plain Index

InnoDB does **not** support adding a plain index to a table with [ALGORITHM](#) set to  
INSTANT

For example, this fails:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD INDEX b_index (b);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

And this fails:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
CREATE INDEX b_index ON tab (b);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

### Adding a Fulltext Index

InnoDB does **not** support adding a [FULLTEXT](#) index to a table with [ALGORITHM](#) set to  
INSTANT

For example, this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD FULLTEXT INDEX b_index (b);
Query OK, 0 rows affected (0.042 sec)

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD FULLTEXT INDEX c_index (c);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY

```

And this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INPLACE';
CREATE FULLTEXT INDEX b_index ON tab (b);
Query OK, 0 rows affected (0.040 sec)

SET SESSION alter_algorithm='INSTANT';
CREATE FULLTEXT INDEX c_index ON tab (c);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY

```

## Adding a Spatial Index

InnoDB does **not** support adding a **Spatial** index to a table with **ALGORITHM** set to  
INSTANT

For example, this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ADD SPATIAL INDEX c_index (c);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY

```

And this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c GEOMETRY NOT NULL
);

SET SESSION alter_algorithm='INSTANT';
CREATE SPATIAL INDEX c_index ON tab (c);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY

```

ALTER TABLE ... DROP INDEX  
and  
DROP INDEX

InnoDB supports dropping indexes from a table with **ALGORITHM** set to  
INSTANT

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the **LOCK** clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50),
    INDEX b_index (b)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP INDEX b_index;
Query OK, 0 rows affected (0.008 sec)
```

And this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50),
    INDEX b_index (b)
);

SET SESSION alter_algorithm='INSTANT';
DROP INDEX b_index ON tab;
Query OK, 0 rows affected (0.007 sec)
```

This applies to [ALTER TABLE ... DROP INDEX](#) and [DROP INDEX](#) for InnoDB tables.

## ALTER TABLE ... ADD FOREIGN KEY

InnoDB does **not** support adding foreign key constraints to a table with [ALGORITHM](#) set to INSTANT.

For example:

```
CREATE OR REPLACE TABLE tab1 (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50),
    d int
);

CREATE OR REPLACE TABLE tab2 (
    a int PRIMARY KEY,
    b varchar(50)
);

SET SESSION foreign_key_checks=OFF;
SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab1 ADD FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a);
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: ADD INDEX. Try ALGORITHM=NOCOPY
```

This applies to [ALTER TABLE ... ADD FOREIGN KEY](#) for InnoDB tables.

## ALTER TABLE ... DROP FOREIGN KEY

InnoDB supports dropping foreign key constraints from a table with [ALGORITHM](#) set to INSTANT.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to

NONE

. When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab2 (
  a int PRIMARY KEY,
  b varchar(50)
);

CREATE OR REPLACE TABLE tab1 (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  d int,
  FOREIGN KEY tab2_fk (d) REFERENCES tab2 (a)
);
;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab1 DROP FOREIGN KEY tab2_fk;
Query OK, 0 rows affected (0.004 sec)

```

This applies to [ALTER TABLE ... DROP FOREIGN KEY](#) for InnoDB tables.

## Table Operations

### ALTER TABLE ... AUTO\_INCREMENT=...

InnoDB supports changing a table's `AUTO_INCREMENT` value with `ALGORITHM` set to `INSTANT`

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to `NONE`. When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab AUTO_INCREMENT=100;
Query OK, 0 rows affected (0.002 sec)

```

This applies to [ALTER TABLE ... AUTO\\_INCREMENT=...](#) for InnoDB tables.

### ALTER TABLE ... ROW\_FORMAT=...

InnoDB does **not** support changing a table's `row format` with `ALGORITHM` set to `INSTANT`

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) ROW_FORMAT=DYNAMIC;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ROW_FORMAT=COMPRESSED;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Changing table options requires the table to be rebuilt. Try ALG

```

This applies to [ALTER TABLE ... ROW\\_FORMAT=...](#) for InnoDB tables.

`ALTER TABLE ... KEY_BLOCK_SIZE=...`

InnoDB does **not** support changing a table's `KEY_BLOCK_SIZE` with `ALGORITHM` set to  
`INSTANT`

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
) ROW_FORMAT=COMPRESSED
KEY_BLOCK_SIZE=4;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab KEY_BLOCK_SIZE=2;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Changing table options requires the table to be rebuilt. Try ALG
```

This applies to `KEY_BLOCK_SIZE=...` for InnoDB tables.

`ALTER TABLE ... PAGE_COMPRESSED=1`

and

`ALTER TABLE ... PAGE_COMPRESSION_LEVEL=...`

In MariaDB 10.3.10 and later, InnoDB supports setting a table's `PAGE_COMPRESSED` value to

```
1
with ALGORITHM set to
INSTANT
. InnoDB does not support changing a table's PAGE_COMPRESSED value from
1
to
0
with ALGORITHM set to
INSTANT
```

In these versions, InnoDB also supports changing a table's `PAGE_COMPRESSION_LEVEL` value with `ALGORITHM` set to  
`INSTANT`

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the `LOCK` clause to

```
NONE
. When this strategy is used, all concurrent DML is permitted.
```

See [MDEV-16328](#) for more information.

For example, this succeeds:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab PAGE_COMPRESSED=1;
Query OK, 0 rows affected (0.004 sec)
```

And this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) PAGE_COMPRESSED=1
PAGE_COMPRESSION_LEVEL=5;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab PAGE_COMPRESSION_LEVEL=4;
Query OK, 0 rows affected (0.004 sec)

```

But this fails:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) PAGE_COMPRESSED=1;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab PAGE_COMPRESSED=0;
ERROR 1846 (0A000): ALGORITHM=INSTANT is not supported. Reason: Changing table options requires the table to be rebuilt. Try ALG

```

This applies to [ALTER TABLE ... PAGE\\_COMPRESSED=...](#) and [ALTER TABLE ... PAGE\\_COMPRESSION\\_LEVEL=...](#) for InnoDB tables.

## ALTER TABLE ... DROP SYSTEM VERSIONING

InnoDB does **not** support dropping [system versioning](#) from a table with [ALGORITHM](#) set to  
INSTANT

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
) WITH SYSTEM VERSIONING;

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP SYSTEM VERSIONING;
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE

```

This applies to [ALTER TABLE ... DROP SYSTEM VERSIONING](#) for InnoDB tables.

## ALTER TABLE ... DROP CONSTRAINT

In MariaDB 10.3.6 and later, InnoDB supports dropping a [CHECK](#) constraint from a table with [ALGORITHM](#) set to  
INSTANT  
. See [MDEV-16331](#) for more information.

This operation supports the non-locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to  
NONE  
. When this strategy is used, all concurrent DML is permitted.

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50),
  CONSTRAINT b_not_empty CHECK (b != '')
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab DROP CONSTRAINT b_not_empty;
Query OK, 0 rows affected (0.002 sec)

```

This applies to [ALTER TABLE ... DROP CONSTRAINT](#) for InnoDB tables.

## ALTER TABLE ... FORCE

InnoDB does **not** support forcing a table rebuild with [ALGORITHM](#) set to  
INSTANT

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab FORCE;
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to [ALTER TABLE ... FORCE](#) for InnoDB tables.

## ALTER TABLE ... ENGINE=InnoDB

InnoDB does **not** support forcing a table rebuild with [ALGORITHM](#) set to  
INSTANT

For example:

```
CREATE OR REPLACE TABLE tab (
    a int PRIMARY KEY,
    b varchar(50),
    c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab ENGINE=InnoDB;
ERROR 1845 (0A000): ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE
```

This applies to [ALTER TABLE ... ENGINE=InnoDB](#) for InnoDB tables.

## OPTIMIZE TABLE ...

InnoDB does **not** support optimizing a table with with [ALGORITHM](#) set to  
INSTANT

For example:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SHOW GLOBAL VARIABLES WHERE Variable_name IN('innodb_defragment', 'innodb_optimize_fulltext_only');
+-----+-----+
| Variable_name | Value |
+-----+-----+
| innodb_defragment | OFF |
| innodb_optimize_fulltext_only | OFF |
+-----+-----+
2 rows in set (0.001 sec)

SET SESSION alter_algorithm='INSTANT';
OPTIMIZE TABLE tab;
+-----+-----+-----+
| Table | Op      | Msg_type | Msg_text
+-----+-----+-----+
| db1.tab | optimize | note    | Table does not support optimize, doing recreate + analyze instead |
| db1.tab | optimize | error   | ALGORITHM=INSTANT is not supported for this operation. Try ALGORITHM=INPLACE |
| db1.tab | optimize | status  | Operation failed
+-----+-----+-----+
3 rows in set, 1 warning (0.002 sec)

```

This applies to [OPTIMIZE TABLE](#) for InnoDB tables.

ALTER TABLE ... RENAME TO  
and  
RENAME TABLE ...

InnoDB supports renaming a table with [ALGORITHM](#) set to  
INSTANT

This operation supports the exclusive locking strategy. This strategy can be explicitly chosen by setting the [LOCK](#) clause to EXCLUSIVE.  
. When this strategy is used, concurrent DML is **not** permitted.

For example, this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
ALTER TABLE tab RENAME TO old_tab;
Query OK, 0 rows affected (0.008 sec)

```

And this succeeds:

```

CREATE OR REPLACE TABLE tab (
  a int PRIMARY KEY,
  b varchar(50),
  c varchar(50)
);

SET SESSION alter_algorithm='INSTANT';
RENAME TABLE tab TO old_tab;
Query OK, 0 rows affected (0.008 sec)

```

This applies to [ALTER TABLE ... RENAME TO](#) and [RENAME TABLE](#) for InnoDB tables.

## Limitations

## Limitations Related to Generated (Virtual and Persistent/Stored) Columns

Generated columns do not currently support online DDL for all of the same operations that are supported for "real" columns.

See [Generated \(Virtual and Persistent/Stored\) Columns: Statement Support](#) for more information on the limitations.

## Non-canonical Storage Format Caused by Some Operations

Some operations cause a table's tablespace file to use a non-canonical storage format when the

INSTANT

algorithm is used. The affected operations include:

- [Adding a column](#).
- [Dropping a column](#).
- [Reordering columns](#).

These operations require the following non-canonical changes to the storage format:

- A hidden metadata record at the start of the clustered index is used to store each column's **DEFAULT** value. This makes it possible to add new columns that have default values without rebuilding the table.
- A **BLOB** in the hidden metadata record is used to store column mappings. This makes it possible to drop or reorder columns without rebuilding the table. This also makes it possible to add columns to any position or drop columns from any position in the table without rebuilding the table.
- If a column is dropped, old records will contain garbage in that column's former position, and new records will be written with **NULL** values, empty strings, or dummy values.

This non-canonical storage format has the potential to incur some performance or storage overhead for all subsequent DML operations. If you notice some issues like this and you want to normalize a table's storage format to avoid this problem, then you can do so by forcing a table rebuild by executing **ALTER TABLE ... FORCE** with **ALGORITHM** set to

INPLACE

. For example:

```
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab FORCE;
Query OK, 0 rows affected (0.008 sec)
```

However, keep in mind that there are certain scenarios where you may not be able to rebuild the table with **ALGORITHM** set to

INPLACE

. See [InnoDB Online DDL Operations with ALGORITHM=INPLACE: Limitations](#) for more information on those cases. If you hit one of those scenarios, but you still want to rebuild the table, then you would have to do so with **ALGORITHM** set to

COPY

## Known Bugs

There are some known bugs that could lead to issues when an InnoDB DDL operation is performed using the **INSTANT** algorithm. This algorithm will usually be chosen by default if the operation supports the algorithm.

The effect of many of these bugs is that the table seems to *forget* that its tablespace file is in the [non-canonical storage format](#).

If you are concerned that a table may be affected by one of these bugs, then your best option would be to normalize the table structure. This can be done by rebuilding the table. For example:

```
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab FORCE;
Query OK, 0 rows affected (0.008 sec)
```

If you are concerned about these bugs, and you want to perform an operation that supports the **INSTANT** algorithm, but you want to avoid using that algorithm, then you can set the algorithm to **INPLACE** and add the

FORCE

keyword to the **ALTER TABLE** statement:

```
SET SESSION alter_algorithm='INPLACE';
ALTER TABLE tab ADD COLUMN c varchar(50), FORCE;
```

## Closed Bugs

- [MDEV-20066](#) : This bug could cause a table to become corrupt if a column was added instantly. It is fixed in [MariaDB 10.3.18](#) and [MariaDB 10.4.8](#).
- [MDEV-20117](#) : This bug could cause a table to become corrupt if a column was dropped instantly. It is fixed in [MariaDB 10.4.9](#).

## Open Bugs

- [MDEV-18519](#) : This bug could cause a table to become corrupt if a column was added instantly.
- [MDEV-19743](#) : This bug could cause a table to become corrupt during page reorganization if a column was added instantly.

- [MDEV-19783](#) : This bug could cause a table to become corrupt if a column was added instantly.
- [MDEV-20090](#) : This bug could cause a table to become corrupt if columns were added, dropped, or reordered instantly.

## 4.3.2.19.5 Instant ADD COLUMN for InnoDB

MariaDB starting with [10.3.2](#)

Instant [ALTER TABLE ... ADD COLUMN](#) for InnoDB was introduced in [MariaDB 10.3.2](#). The [INSTANT](#) option for the [ALGORITHM](#) clause was introduced in [MariaDB 10.3.7](#).

### Contents

1. [Limitations](#)
2. [Example](#)

Normally, adding a column to a table requires the full table to be rebuilt. The complexity of the operation is proportional to the size of the table, or  $O(n \cdot m)$  where n is the number of rows in the table and m is the number of indexes.

In [MariaDB 10.0](#) and later, the [ALTER TABLE](#) statement supports [online DDL](#) for storage engines that have implemented the relevant online DDL [algorithms](#) and [locking strategies](#).

The [InnoDB](#) storage engine has implemented online DDL for many operations. These online DDL optimizations allow concurrent DML to the table in many cases, even if the table needs to be rebuilt.

See [InnoDB Online DDL Overview](#) for more information about online DDL with InnoDB.

Allowing concurrent DML during the operation does not solve all problems. When a column was added to a table with the older in-place optimization, the resulting table rebuild could still significantly increase the I/O and memory consumption and cause replication lag.

In contrast, with the new instant [ALTER TABLE ... ADD COLUMN](#), all that is needed is an  $O(\log n)$  operation to insert a special hidden record into the table, and an update of the data dictionary. For a large table, instead of taking several hours, the operation would be completed in the blink of an eye. The [ALTER TABLE ... ADD COLUMN](#) operation is only slightly more expensive than a regular [INSERT](#), due to locking constraints.

In the past, some developers may have implemented a kind of "instant add column" in the application by encoding multiple columns in a single [TEXT](#) or [BLOB](#) column. MariaDB [Dynamic Columns](#) was an early example of that. A more recent example is [JSON](#) and related string manipulation functions.

Adding real columns has the following advantages over encoding columns into a single "expandable" column:

- Efficient storage in a native binary format
- Data type safety
- Indexes can be built natively
- Constraints are available: UNIQUE, CHECK, FOREIGN KEY
- DEFAULT values can be specified
- Triggers can be written more easily

With instant [ALTER TABLE ... ADD COLUMN](#), you can enjoy all the benefits of structured storage without the drawback of having to rebuild the table.

Instant [ALTER TABLE ... ADD COLUMN](#) is available for both old and new InnoDB tables. Basically you can just upgrade from MySQL 5.x or MariaDB and start adding columns instantly.

Columns instantly added to a table exist in a separate data structure from the main table definition, similar to how InnoDB separates

[BLOB](#) columns. If the table ever becomes empty, (such as from [TRUNCATE](#) or [DELETE](#) statements), InnoDB incorporates the instantly added columns into the main table definition. See [InnoDB Online DDL Operations with ALGORITHM=INSTANT: Non-canonical Storage Format Caused by Some Operations](#) for more information.

The operation is also crash safe. If the server is killed while executing an instant [ALTER TABLE ... ADD COLUMN](#), when the table is restored InnoDB integrates the new column, flattening the table definition.

## Limitations

- In [MariaDB 10.3](#), instant [ALTER TABLE ... ADD COLUMN](#) only applies when the added columns appear last in the table. The place specifier [LAST](#) is the default. If [AFTER](#) col is specified, then col must be the last column, or the operation will require the table to be rebuilt. In [MariaDB 10.4](#), this restriction has been lifted.
- If the table contains a hidden [FTS\\_DOC\\_ID](#) column due to a [FULLTEXT INDEX](#), then instant [ALTER TABLE ... ADD COLUMN](#) will not be possible.
- InnoDB data files after instant [ALTER TABLE ... ADD COLUMN](#) cannot be imported to older versions of MariaDB or MySQL without first being rebuilt.
- After using Instant [ALTER TABLE ... ADD COLUMN](#), any table-rebuilding operation such as [ALTER TABLE ... FORCE](#) will incorporate instantaneously added columns into the main table body.

- Instant ALTER TABLE ... ADD COLUMN is not available for ROW\_FORMAT=COMPRESSED .
- In MariaDB 10.3 , ALTER TABLE ... DROP COLUMN requires the table to be rebuilt. In MariaDB 10.4 , this restriction has been lifted.

## Example

```

CREATE TABLE t(id INT PRIMARY KEY, u INT UNSIGNED NOT NULL UNIQUE)
ENGINE=InnoDB;

INSERT INTO t(id,u) VALUES(1,1),(2,2),(3,3);

ALTER TABLE t ADD COLUMN
(d DATETIME DEFAULT current_timestamp(),
 p POINT NOT NULL DEFAULT ST_GeomFromText('POINT(0 0)'),
 t TEXT CHARSET utf8 DEFAULT 'The quick brown fox jumps over the lazy dog');

UPDATE t SET t=NULL WHERE id=3;

SELECT id,u,d,ST_AsText(p),t FROM t;

SELECT variable_value FROM information_schema.global_status
WHERE variable_name = 'innodb_instant_alter_column';

```

The above example illustrates that when the added columns are declared NOT NULL, a DEFAULT value must be available, either implied by the data type or set explicitly by the user. The expression need not be constant, but it must not refer to the columns of the table, such as DEFAULT u+1 (a MariaDB extension). The DEFAULT current\_timestamp() would be evaluated at the time of the ALTER TABLE and apply to each row, like it does for non-instant ALTER TABLE. If a subsequent ALTER TABLE changes the DEFAULT value for subsequent INSERT, the values of the columns in existing records will naturally be unaffected.

The design was brainstormed in April by engineers from MariaDB Corporation, Alibaba and Tencent. A prototype was developed by Vin Chen (陈福荣) from the Tencent Game DBA Team.

## 4.3.2.20 Binary Log Group Commit and InnoDB Flushing Performance

### Contents

1. Overview
2. Switching to Old Flushing Behavior
  1. Non-durable Binary Log Settings
  2. Recent Transactions Missing from Backups

MariaDB 10.0 introduced a performance improvement related to [group commit](#) that affects the performance of flushing InnoDB transactions when the [binary log](#) is enabled.

## Overview

In MariaDB 10.0 and above, when both [innodb\\_flush\\_log\\_at\\_trx\\_commit=1](#) (the default) is set and the [binary log](#) is enabled, there is now one less sync to disk inside InnoDB during commit (2 syncs shared between a group of transactions instead of 3).

Durability of commits is not decreased — this is because even if the server crashes before the commit is written to disk by InnoDB, it will be recovered from the binary log at next server startup (and it is guaranteed that sufficient information is synced to disk so that such a recovery is always possible).

## Switching to Old Flushing Behavior

The old behavior, with 3 syncs to disk per (group) commit (and consequently lower performance), can be selected with the new [innodb\\_flush\\_log\\_at\\_trx\\_commit=3](#) option. There is normally no benefit to doing this, however there are a couple of edge cases to be aware of.

## Non-durable Binary Log Settings

If [innodb\\_flush\\_log\\_at\\_trx\\_commit=1](#) is set and the [binary log](#) is enabled, but [sync\\_binlog=0](#) is set, then commits are not guaranteed durable inside InnoDB after commit. This is because if [sync\\_binlog=0](#) is set and if the server crashes, then transactions that were not flushed to the binary log prior to the crash will be missing from the binary log.

In this specific scenario, [innodb\\_flush\\_log\\_at\\_trx\\_commit=3](#) can be set to ensure that transactions will be durable in InnoDB, even if they are not necessarily durable from the perspective of the binary log.

One should be aware that if [sync\\_binlog=0](#) is set, then a crash is nevertheless likely to cause transactions to be missing from the binary log. This will cause the binary log and InnoDB to be inconsistent with each other. This is also likely to cause any [replication slaves](#) to become inconsistent, since transactions are replicated through the [binary log](#) . Thus it is recommended to set [sync\\_binlog=1](#) . With the [group commit](#) improvements introduced in

## Recent Transactions Missing from Backups

Mariabackup and Percona XtraBackup only see transactions that have been flushed to the [redo log](#) . With the [group commit](#) improvements, there may be a small delay (defined by the [binlog\\_commit\\_wait\\_usecs](#) system variable) between when a commit happens and when the commit will be included in a backup.

Note that the backup will still be fully consistent with itself and the [binary log](#) . This problem is normally not an issue in practice. A backup usually takes a long time to complete (relative to the 1 second or so that [binlog\\_commit\\_wait\\_usecs](#) is normally set to), and a backup usually includes a lot of transactions that were committed during the backup. With this in mind, it is not generally noticeable if the backup does not include transactions that were committed during the last 1 second or so of the backup process. It is just mentioned here for completeness.

## 4.3.3 MariaDB ColumnStore

MariaDB ColumnStore is a columnar storage engine that utilizes a massively parallel distributed data architecture. It's a columnar storage system built by porting InfiniDB 4.6.7 to MariaDB, and released under the GPL license.

From [MariaDB 10.5.4](#) , it is available as a storage engine for MariaDB Server. Before then, it is only available as a separate download.

MariaDB ColumnStore is designed for big data scaling to process petabytes of data, linear scalability and exceptional performance with real-time response to analytical queries. It leverages the I/O benefits of columnar storage, compression, just-in-time projection, and horizontal and vertical partitioning to deliver tremendous performance when analyzing large data sets.

Documentation for the latest release of Columnstore is not available on the Knowledge Base. Instead, see:

- [Release Notes](#)
- [Deployment Instructions](#)

### About MariaDB ColumnStore

[About MariaDB ColumnStore](#).

### MariaDB ColumnStore Release Notes

[MariaDB ColumnStore Release Notes](#)

### ColumnStore Getting Started

[Quick summary of steps needed to install MariaDB ColumnStore](#)

### ColumnStore Upgrade Guides

[Documentation on upgrading from prior versions and InfiniDB migration.](#)

### ColumnStore Architecture

[MariaDB ColumnStore Architecture](#)

### Managing ColumnStore

[Managing MariaDB ColumnStore System Environment and Database](#)

### ColumnStore Data Ingestion

[How to load and manipulate data into MariaDB ColumnStore](#)

### ColumnStore SQL Structure and Commands

[SQL syntax supported by MariaDB ColumnStore](#)

### ColumnStore Performance Tuning

[Information relating to configuring and analyzing the ColumnStore system for optimal performance.](#)

### ColumnStore System Variables

[ColumnStore System Variables](#)

### ColumnStore Security Vulnerabilities

[Security vulnerabilities affecting MariaDB ColumnStore](#)

### ColumnStore Troubleshooting

[Articles on troubleshooting tips and techniques](#)

### StorageManager

[Articles on StorageManager and S3 configuration](#)

### Using MariaDB ColumnStore

[Provides details on using third party products and tools with MariaDB ColumnStore](#)



## Building ColumnStore in MariaDB

This is a description of how to build and start a local ColumnStore install...



## Can't create a table starting with a capital letter. All tables are lower case-

Hi, I was playing around with my MariaDB ColumnStore and I noticed the I am...

There are 52 related questions .

## 4.3.4 Aria

### 4.3.4.1 Aria Storage Engine

#### Contents

1. [mysqld Startup Options for Aria](#)
2. [See Also](#)

The **Aria** storage engine is compiled in by default from [MariaDB 5.1](#) and it is required to be 'in use' when mysqld is started.

From [MariaDB 10.4](#), all [system tables](#) are Aria.

Additionally, internal on-disk tables are in the Aria table format instead of the [MyISAM](#) table format. This should speed up some [GROUP BY](#) and [DISTINCT](#) queries because Aria has better caching than MyISAM.

Note: The **Aria** storage engine was previously called **Maria** (see [The Aria Name](#) for details on the rename) and in previous versions of [MariaDB](#) the engine was still called Maria.

The following table options to Aria tables in

[CREATE TABLE](#)

and

[ALTER TABLE](#)

#:

• **TRANSACTIONAL= 0**

|

1  
: If the  
TRANSACTIONAL

table option is set for an Aria table, then the table will be crash-safe. This is implemented by logging any changes to the table to Aria's transaction log, and syncing those writes at the end of the statement. This will marginally slow down writes and updates. However, the benefit is that if the server dies before the statement ends, all non-durable changes will roll back to the state at the beginning of the statement. This also needs up to 6 bytes more for each row and key to store the transaction id (to allow concurrent insert's and selects).

◦

TRANSACTIONAL=1  
is not supported for partitioned tables.

◦ An Aria table's default value for the

TRANSACTIONAL  
table option depends on the table's value for the  
ROW\_FORMAT  
table option. See below for more details.

◦ If the

TRANSACTIONAL  
table option is set for an Aria table, the table does not actually support transactions. See [MDEV-21364](#) for more information. In this context, *transactional* just means *crash-safe*.

• **PAGE\_CHECKSUM= 0**

|

1  
: If index and data should use page checksums for extra safety.

- TABLE\_CHECKSUM= 0
  - |
  - 1 : Same as CHECKSUM in MySQL 5.1
- ROW\_FORMAT=PAGE
  - |
  - FIXED
  - |
  - DYNAMIC : The table's [row format](#) .
    - The default value is PAGE
    - . To emulate MyISAM, set ROW\_FORMAT=FIXED or ROW\_FORMAT=DYNAMIC

The

TRANSACTIONAL  
and  
ROW\_FORMAT  
table options interact as follows:

- If TRANSACTIONAL=1  
is set, then the only supported row format is PAGE
  - . If ROW\_FORMAT is set to some other value, then Aria issues a warning, but still forces the row format to be PAGE
- If TRANSACTIONAL=0  
is set, then the table will be not be crash-safe, and any row format is supported.
- If TRANSACTIONAL  
is not set to any value, then any row format is supported. If ROW\_FORMAT is set, then the table will use that row format. Otherwise, the table will use the default PAGE row format. In this case, if the table uses the PAGE row format, then it will be crash-safe. If it uses some other row format, then it will not be crash-safe.

Some other improvements are:

- - CHECKSUM\_TABLE
    - now ignores values in NULL fields. This makes CHECKSUM\_TABLE faster and fixes some cases where same table definition could give different checksum values depending on [row format](#) . The disadvantage is that the value is now different compared to other MySQL installations. The new checksum calculation is fixed for all table engines that uses the default way to calculate and MyISAM which does the calculation internally. Note: Old MyISAM tables with internal checksum will return the same checksum as before. To fix them to calculate according to new rules you have to do an ALTER TABLE
      - . You can use the old ways to calculate checksums by using the option --old

```
to mysqld or set the system variable '  
@@old  
' to  
1  
when you do  
CHECKSUM TABLE ... EXTENDED;
```

- At startup Aria will check the Aria logs and automatically recover the tables from the last checkpoint if mysqld was not taken down correctly.

## mysqld Startup Options for Aria

For a full list, see [Aria System Variables](#).

In normal operations, the only variables you have to consider are:

- [aria-pagelcache-buffer-size](#)
  - This is where all index and data pages are cached. The bigger this is, the faster Aria will work.
- [aria-block-size](#)
  - The default value 8192, should be ok for most cases. The only problem with a higher value is that it takes longer to find a packed key in the block as one has to search roughly 8192/2 to find each key. We plan to fix this by adding a dictionary at the end of the page to be able to do a binary search within the block before starting a scan. Until this is done and key lookups takes too long time even if you are not hitting disk, then you should consider making this smaller.
  - Possible values to try are
    - 2048
    - ,
    - 4096
    - or
    - 8192
- Note that you can't change this without dumping, deleting old tables and deleting all log files and then restoring your Aria tables. (This is the only option that requires a dump and load)
- [aria-log-purge-type](#)
  - Set this to "[at\\_flush](#)" if you want to keep a copy of the transaction logs (good as an extra backup). The logs will stay around until you execute [FLUSH ENGINE LOGS](#).

## See Also

- [Aria FAQ](#)

### 4.3.4.2 Aria Clients and Utilities

#### 4.3.4.2.1 aria\_chk

##### Contents

1. [Options and Variables](#)
  1. [Global Options](#)
  2. [Main Arguments](#)
  3. [Check Options \(--check is the Default Action for aria\\_chk\):](#)
  4. [Recover \(Repair\) Options \(When Using '--recover' or '--safe-recover'\):](#)
  5. [Other Options](#)
  6. [Variables](#)
2. [Usage](#)

`aria_chk`  
is used to check, repair, optimize, sort and get information about [Aria](#) tables.

With the MariaDB server you can use [CHECK TABLE](#), [REPAIR TABLE](#) and [OPTIMIZE TABLE](#) to do similar things.

##### Note:

`aria_chk`  
should not be used when MariaDB is running. MariaDB assumes that no one is changing the tables it's using!

Usage:

```
aria_chk [OPTIONS] aria_tables[.MAI]
```

Aria table information is stored in 2 files: the

.MAI  
file contains base table information and the index and the  
.MAD  
file contains the data.  
aria\_chk  
takes one or more  
.MAI  
files as arguments.

The following groups are read from the my.cnf files:

- [maria\_chk]
- [aria\_chk]

## Options and Variables

### Global Options

The following options to handle option files may be given as the first argument:

| Option                      | Description                                      |
|-----------------------------|--------------------------------------------------|
| --<br>print-defaults        | Print the program argument list and exit.        |
| --<br>no-defaults           | Don't read default options from any option file. |
| --<br>defaults-file=#       | Only read default options from the given file #. |
| --<br>defaults-extra-file=# | Read this file after the global files are read.  |

### Main Arguments

| Option | Description |
|--------|-------------|
|--------|-------------|

|                                |                                                                                                       |
|--------------------------------|-------------------------------------------------------------------------------------------------------|
| -<br>#<br>,                    | Output debug log. Often this is 'd:o,filename'.                                                       |
| --<br>debug=...                |                                                                                                       |
| -H<br>,                        | Display this help and exit.                                                                           |
| --<br>HELP                     |                                                                                                       |
| -?<br>,                        | Display this help and exit.                                                                           |
| --<br>help                     |                                                                                                       |
| --<br>datadir=path             | Path for control file (and logs if<br>--logdir<br>not used).                                          |
| --<br>ignore-<br>control-file  | Don't open the control file. Only use this if you are sure the tables are not used by another program |
| --<br>logdir=path              | Path for log files.                                                                                   |
| --<br>require-<br>control-file | Abort if we can't find/read the maria_log_control file                                                |
| -s<br>,                        | Only print errors. One can use two -s to make aria_chk very silent.                                   |
| --<br>silent                   |                                                                                                       |

|                       |                                                                                                                                                                    |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -t<br>,               | Path for temporary files. Multiple paths can be specified, separated by colon (:) on Unix or semicolon (;) on Windows. They will be used in a round-robin fashion. |
| --<br><br>tmpdir=path |                                                                                                                                                                    |
| -v<br>,               | Print more information. This can be used with<br>--description<br>and<br>--check<br>. Use many -v for more verbosity.                                              |
| --<br><br>verbose     |                                                                                                                                                                    |
| -V<br>,               | Print version and exit.                                                                                                                                            |
| --<br><br>version     |                                                                                                                                                                    |
| -w<br>,               | Wait if table is locked.                                                                                                                                           |
| --<br><br>wait        |                                                                                                                                                                    |

### Check Options (--check is the Default Action for aria\_chk):

| Option                 | Description                                                                                                                                                  |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -c<br>,                | Check table for errors.                                                                                                                                      |
| --<br><br>check        |                                                                                                                                                              |
| -e<br>,                | Check the table VERY thoroughly. Only use this in extreme cases as aria_chk should normally be able to find out if the table is ok even without this switch. |
| --<br><br>extend-check |                                                                                                                                                              |

|                              |                                                                                                                          |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| -F<br>,                      | Check only tables that haven't been closed properly.                                                                     |
| --<br>fast                   |                                                                                                                          |
| -C<br>,                      | Check only tables that have changed since last check.                                                                    |
| --<br>check-only-<br>changed |                                                                                                                          |
| -f<br>,                      | Restart with '<br>-r<br>' if there are any errors in the table. States will be updated as with '<br>--update-state<br>'. |
| --<br>force                  |                                                                                                                          |
| -i<br>,                      | Print statistics information about table that is checked.                                                                |
| --<br>information            |                                                                                                                          |
| -m<br>,                      | Faster than extend-check, and finds 99.99% of all errors. Should be good enough for most cases.                          |
| --<br>medium-check           |                                                                                                                          |

|         |                                                                                                                                                                                                                                                                                                                                                                     |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -U<br>, | Mark tables as crashed if any errors were found and clean if check didn't find any errors but table was marked as 'not clean' before. This allows one to get rid of warnings like 'table not properly closed'. If table was updated, update also the timestamp for when the check was made. This option is on by default! Use<br>--skip-update-state<br>to disable. |
| -T<br>, | Don't mark table as checked.                                                                                                                                                                                                                                                                                                                                        |

### Recover (Repair) Options (When Using '--recover' or '--safe-recover'):

| Option                           | Description                                                                                                                                                       |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -B<br>,                          |                                                                                                                                                                   |
| --<br>backup                     | Make a backup of the .MAD file as 'filename-time.BAK'.                                                                                                            |
| --<br>correct-<br>checksum       | Correct checksum information for table.                                                                                                                           |
| -D<br>,                          |                                                                                                                                                                   |
| --<br>data-file-<br>length=<br># | Max length of data file (when recreating data file when it's full).                                                                                               |
| -e<br>,                          |                                                                                                                                                                   |
| --<br>extend-check               | Try to recover every possible row from the data file. Normally this will also find a lot of garbage rows; Don't use this option if you are not totally desperate. |

|         |                                                  |                                                                                                                                 |
|---------|--------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
|         | <code>-f</code><br>,--<br><b>force</b>           | Overwrite old temporary files.                                                                                                  |
|         | <code>-k</code><br>,--<br><b>keys-used=</b><br># | Tell MARIA to update only some specific keys. # is a bit mask of which keys to use. This can be used to get faster inserts.     |
| length= | --<br><b>max-record-</b><br>#                    | Skip rows bigger than this if aria_chk can't allocate memory to hold it.                                                        |
|         | <code>-r</code><br>,--<br><b>recover</b>         | Can fix almost anything except unique keys that aren't unique.                                                                  |
|         | <code>-n</code><br>,--<br><b>sort-recover</b>    | Forces recovering with sorting even if the temporary file would be very big.                                                    |
| recover | <code>-p</code><br>,--<br><b>parallel-</b>       | Uses the same technique as '-r' and '-n', but creates all the keys in parallel, in different threads.                           |
|         | <code>-o</code><br>,--<br><b>safe-recover</b>    | Uses old recovery method; Slower than '-r' but can handle a couple of cases where '-r' reports that it can't fix the data file. |

|                                  |                                                                                                                                                                                                                                              |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| --<br>transaction-<br>log        | Log repair command to transaction log. This is needed if one wants to use the maria_read_log to repeat the repair.                                                                                                                           |
| --<br>character-<br>sets-dir=... | Directory where character sets are.                                                                                                                                                                                                          |
| --<br>set-<br>collation=name     | Change the collation used by the index.                                                                                                                                                                                                      |
| -q<br>,<br>--<br>quick           | Faster repair by not modifying the data file. One can give a second '<br>-q<br>' to force aria_chk to modify the original datafile in case of duplicate keys. NOTE: Tables where the data file is corrupted can't be fixed with this option. |
| -u<br>,<br>--<br>unpack          | Unpack file packed with <a href="#">aria_pack</a> .                                                                                                                                                                                          |

## Other Options

| Option                       | Description                                                                                                                                                                             |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -a<br>,<br>--<br>analyze     | Analyze distribution of keys. Will make some joins in MariaDB faster. You can check the calculated distribution by using '<br>--description --verbose table_name<br>'                   |
| --<br>stats_method=name      | Specifies how index statistics collection code should treat NULLs. Possible values of name are "nulls_unequal" (default for 4.1/5.0), "nulls_equal" (emulate 4.0), and "nulls_ignored". |
| -d<br>,<br>--<br>description | Prints some information about table.                                                                                                                                                    |

|                                  |                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -A<br>,                          |                                                                                                                                                                                                                                                                                                                                      |
| --set-<br>auto-increment[=value] | Force auto_increment to start at this or higher value If no value is given, then sets the next auto_increment value to the highest used value for the auto key + 1.                                                                                                                                                                  |
| -S<br>,                          |                                                                                                                                                                                                                                                                                                                                      |
| --sort-index                     | Sort index blocks. This speeds up 'read-next' in applications.                                                                                                                                                                                                                                                                       |
| -R<br>,                          |                                                                                                                                                                                                                                                                                                                                      |
| --sort-<br>records=              | Sort records according to an index. This makes your data much more localized and may speed up things (It may be VERY slow to do a sort the first time!).                                                                                                                                                                             |
| #                                |                                                                                                                                                                                                                                                                                                                                      |
| -b<br>,                          |                                                                                                                                                                                                                                                                                                                                      |
| --block-search=                  | Find a record, a block at given offset belongs to.                                                                                                                                                                                                                                                                                   |
| #                                |                                                                                                                                                                                                                                                                                                                                      |
| -z<br>,                          |                                                                                                                                                                                                                                                                                                                                      |
| --zerofill                       | Remove transaction id's from the data and index files and fills empty space in the data and index files with zeroes. Zerofilling makes it possible to move the table from one system to another without the server having to do an automatic zerofill. It also allows one to compress the tables better if one want to archive them. |
| --zerofill-keep-lsn              | Like<br>--zerofill<br>but does not zero out LSN of data/index pages.                                                                                                                                                                                                                                                                 |

## Variables

| Option           | Description                                   |
|------------------|-----------------------------------------------|
| page_buffer_size | Size of page buffer. Used by<br>--safe-repair |

|                   |                                                                                          |
|-------------------|------------------------------------------------------------------------------------------|
| read_buffer_size  | Read buffer size for sequential reads during scanning                                    |
| write_buffer_size | Write buffer size for sequential writes during repair of fixed size or dynamic size rows |
| sort_buffer_size  | Size of sort buffer. Used by<br>--recover                                                |
| sort_key_blocks   | Internal buffer for sorting keys; Don't touch :)                                         |

## Usage

One main usage of

`aria_chk`

is when you want to do a fast check of all Aria tables in your system. This is faster than doing it in MariaDB as you can allocate all free memory to the buffers.

Assuming you have a bit more than 2G free memory.

The following commands, run in the MariaDB data directory, check all your tables and repairs only those that have an error:

```
aria_chk --check --sort_order --force --sort_buffer_size=1G */*.MAI
```

If you want to optimize all your tables: (The

`--zerofill`

is used here to fill up empty space with

`\0`

which can speed up compressed backups).

```
aria_chk --analyze --sort-index --page_buffer_size=1G --zerofill */*.MAI
```

In case you have a serious problem and have to use

`--safe-recover`

:

```
aria_chk --safe-recover --zerofill --page_buffer_size=2G */*.MAI
```

## 4.3.4.2.2 aria\_pack

### Contents

1. [Options](#)
2. [Unpacking](#)
3. [Example](#)
4. [See Also](#)

`aria_pack` is a tool for compressing [Aria](#) tables. The resulting table are read-only, and usually about 40% to 70% smaller.

`aria_pack` is run as follows

```
aria_pack [options] file_name [file_name2...]
```

The file name is the .MAI index file. The extension can be omitted, although keeping it permits wildcards, such as

```
aria_pack *.MAI
```

to compress all the files.

`aria_pack` compresses each column separately, and, when the resulting data is read, only the individual rows and columns required need to be decompressed, allowing for quicker reading.

Once a table has been packed, use `aria_chk -rq` (the quick and recover options) to rebuild its indexes.

## Options

The following variables can be set while passed as commandline options to aria\_pack, or set in the [ariapack] section in your [my.cnf](#) file.

| Option                    | Description                                                                                      |
|---------------------------|--------------------------------------------------------------------------------------------------|
| -b, --backup              | Make a backup of the table as table_name.OLD.                                                    |
| --character-sets-dir=name | Directory where character sets are.                                                              |
| -h, --datadir             | Path for control file (and logs if<br>--logdir<br>not used). From <a href="#">MariaDB 10.5.3</a> |
| -#, --debug[=name]        | Output debug log. Often this is 'd:t:o,filename'.                                                |
| -?, --help                | Display help and exit.                                                                           |
| -f, --force               | Force packing of table even if it gets bigger or if tempfile exists.                             |
| --ignore-control-file     | Ignore the control file. From <a href="#">MariaDB 10.5.3</a> .                                   |
| -j, --join=name           | Join all given tables into 'new_table_name'. All tables MUST have identical layouts.             |
| --require-control-file    | Abort if cannot find control file. From <a href="#">MariaDB 10.5.3</a> .                         |
| -s, --silent              | Only write output when an error occurs.                                                          |
| -t, --test                | Don't pack table, only test packing it.                                                          |
| -T, --tmpdir=name         | Use temporary directory to store temporary table.                                                |
| -v, --verbose             | Write info about progress and packing result. Use many -v for more verbosity!                    |
| -V, --version             | Output version information and exit.                                                             |
| -w, --wait                | Wait and retry if table is in use.                                                               |

## Unpacking

To unpack a table compressed with aria\_pack, use the [aria\\_chk -u](#) option.

## Example

```
> aria_pack /my/data/test/posts
Compressing /my/data/test/posts.MAD: (1690 records)
- Calculating statistics
- Compressing file
37.71%
> aria_chk -rq --ignore-control-file /my/data/test/posts
- check record delete-chain
- recovering (with keycache) Aria-table '/my/data/test/posts'
Data records: 1690
State updated
```

## See Also

- [FLUSH TABLES FOR EXPORT](#)
- [myisamchk](#)

### 4.3.4.2.3 aria\_read\_log

[aria\\_read\\_log](#) is a tool for displaying and applying log records from an [Aria](#) transaction log.

Note: Aria is compiled without -DIDENTICAL\_PAGES\_AFTER\_RECOVERY which means that the table files are not byte-to-byte identical to files created during normal execution. This should be ok, except for test scripts that try to compare files before and after recovery.

Usage:

```
aria_read_log OPTIONS
```

You need to use one of

-d  
or  
-a  
.

# Options

The following variables can be set while passed as commandline options to aria\_read\_log, or set in the

[aria\_read\_log]  
section in your [my.cnf](#) file.

| Option                       | Description                                                                                                                   |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| -a, --apply                  | Apply log to tables: modifies tables! you should make a backup first! Displays a lot of information if not run with --silent. |
| --character-sets-dir=name    | Directory where character sets are.                                                                                           |
| -c, --check                  | if --display-only, check if record is fully readable (for debugging).                                                         |
| -?, --help                   | Display help and exit.                                                                                                        |
| -d, --display-only           | Display brief info read from records' header.                                                                                 |
| -e, --end-lsn=#              | Stop applying at this lsn. If end-lsn is used, UNDO:s will not be applied                                                     |
| -h, --aria-log-dir-path=name | Path to the directory where to store transactional log                                                                        |
| -P, --page-buffer-size=#     | The size of the buffer used for index blocks for Aria tables.                                                                 |
| -l, --print-log-control-file | Print the content of the aria_log_control_file. From <a href="#">MariaDB 10.4.1</a> .                                         |
| -o, --start-from-lsn=#       | Start reading log from this lsn.                                                                                              |
| -C, --start-from-checkpoint  | Start applying from last checkpoint.                                                                                          |
| -s, --silent                 | Print less information during apply/undo phase.                                                                               |
| -T, --tables-to-redo=name    | List of comma-separated tables that we should apply REDO on. Use this if you only want to recover some tables.                |
| -t, --tmpdir=name            | Path for temporary files. Multiple paths can be specified, separated by colon (:)                                             |
| --translog-buffer-size=#     | The size of the buffer used for transaction log for Aria tables.                                                              |
| -u, --undo                   | Apply UNDO records to tables. (disable with --disable-undo) (Defaults to on; use --skip-undo to disable.)                     |
| -v, --verbose                | Print more information during apply/undo phase.                                                                               |
| -V, --version                | Print version and exit.                                                                                                       |

## 4.3.4.2.4 aria\_s3\_copy

MariaDB starting with [10.5](#)

The [S3 storage engine](#) has been available since [MariaDB 10.5.4](#).

`aria_s3_copy`  
is a tool for copying an [Aria](#) table to and from [S3](#).

The Aria table must be non transactional and have [ROW\\_FORMAT=PAGE](#).

For

`aria_s3_copy`

to work reliably, the table should not be changed by the MariaDB server during the copy, and one should have first performed [FLUSH TABLES](#) to ensure that the table is properly closed.

Example of properly created Aria table:

```
create table test1 (a int) transactional=0 row_format=PAGE engine=aria;
```

Note that [ALTER TABLE table\\_name ENGINE=S3](#) will work for any kind of table. This internally converts the table to an Aria table and then moves it to S3 storage.

## Main Arguments

| Option                   | Description                 |
|--------------------------|-----------------------------|
| -?, --help               | Display this help and exit. |
| -k, --s3-access-key=name | AWS access key ID           |
| -r, --s3-region=name     | AWS region                  |
| -K, --s3-secret-key=name | AWS secret access key ID    |

|                                |                                                                                                  |
|--------------------------------|--------------------------------------------------------------------------------------------------|
| -b, --s3-bucket=name           | AWS prefix for tables                                                                            |
| -h, --s3-host-name=name        | Host name to S3 provider                                                                         |
| -c, --compress                 | Use compression                                                                                  |
| -o, --op=name                  | Operation to execute. One of 'from_s3', 'to_s3' or 'delete_from_s3'                              |
| -d, --database=name            | Database for copied table (second prefix). If not given, the directory of the table file is used |
| -B, --s3-block-size=#          | Block size for data/index blocks in s3                                                           |
| -L, --s3-protocol-version=name | Protocol used to communication with S3. One of "Amazon" or "Original".                           |
| -f, --force                    | Force copy even if target exists                                                                 |
| -v, --verbose                  | Write more information                                                                           |
| -V, --version                  | Print version and exit.                                                                          |
| #, --debug[=name]              | Output debug log. Often this is 'd:to,filename'.                                                 |
| --s3-debug                     | Output debug log from marias3 to stdout                                                          |

## Typical Configuration in a my.cnf File

```
[aria_s3_copy]
s3-bucket=mariadb
s3-access-key=xxxx
s3-secret-key=xxx
s3-region=eu-north-1
#s3-host-name=s3.amazonaws.com
#s3-protocol-version=Amazon
verbose=1
op=to
```

## Example Usage

The following code will copy an existing Aria table named

```
test1
to S3. If the
--database
option is not given, then the directory name where the table files exist will be used as the database.
```

```
shell> aria_s3_copy --force --op=to --database=foo --compress --verbose --s3_block_size=4M test1
Delete of aria table: foo.test1
Delete of index information foo/test1/index
Delete of data information foo/test1/data
Delete of base information and frm
Copying frm file test1.frm
Copying aria table: foo.test1 to s3
Creating aria table information foo/test1/aria
Copying index information foo/test1/index
.
Copying data information foo/test1/data
```

When using

```
--verbose
,
aria_s3_copy
will write a dot for each #/79 part of the file copied.
```

## See Also

[Using the S3 storage engine](#) . This pages has examples of .my.cnf entries for using

```
aria_s3_copy
```

## 4.3.4.3 Aria FAQ

This FAQ provides information on the [Aria](#) storage engine.

The **Aria** storage engine was previously known as **Maria** , (see, the [Aria Name](#) ). In current releases of **MariaDB** , you can refer to the engine as Maria or Aria. As this will change in future releases, please update references in your scripts and automation to use the correct name.

## Contents

1. [What is Aria?](#)
2. [Why is the engine called Aria?](#)
3. [What's the goal for the current version?](#)
4. [What's the goal for the next version?](#)
5. [What is the ultimate goal of Aria?](#)
6. [What are the design goals in Aria?](#)
7. [Where can I find documentation and help about Aria?](#)
8. [Who develops Aria?](#)
9. [What is the release policy/schedule of Aria?
  1. \[Extended commitment for Beta 1.5\]\(#\)](#)
10. [How does Aria 1.5 Compare to MyISAM?](#)
11. [Advantages of Aria compared to MyISAM](#)
12. [Differences between Aria and MyISAM](#)
13. [Disadvantages of Aria compared to MyISAM](#)
14. [Differences between MariaDB 5.1 release and the normal MySQL-5.1 release?](#)
15. [Why do you use the TRANSACTIONAL keyword now when Aria is not yet transactional?](#)
16. [What are the known problems with the MySQL-5.1-Maria release?](#)
17. [What is going to change in later Aria main releases?](#)
18. [How can I create a MyISAM-like \(non-transactional\) table in Aria?](#)
19. [What are the advantages/disadvantages of the new PAGE format compared to the old MyISAM-like row formats \(DYNAMIC and FIXED\)](#)
20. [What's the proper way to copy a Aria table from one place to another?](#)
21. [When is it safe to remove old log files?](#)

## What is Aria?

Aria is a storage engine for MySQL ® and MariaDB. It was originally developed with the goal of becoming the default transactional **and** non-transactional storage engine for MariaDB and MySQL.

It has been in development since 2007 and was first announced on Monty's [blog](#) . The same core MySQL engineers who developed the MySQL server and the [MyISAM](#) , [MERGE](#) , and [MEMORY](#) storage engines are also working on Aria.

## Why is the engine called Aria?

Originally, the storage engine was called **Maria** , after Monty's younger daughter. Monty named MySQL after his first child, **My** and his second child **Max** gave his name to MaxDB and the MySQL-Max distributions.

In practice, having both *MariaDB* the database server and *Maria* the storage engine with such similar names proved confusing. To mitigate this, the decision was made to change the name. A Rename Maria contest was held during the first half of 2010 and names were submitted from around the world. Monty picked the name **Aria** from a short list of finalist. Chris Tooley, who suggested it, received the prize of a Linux-powered [System 76 Meerkat NetTop](#) from Monty Program.

For more information, see the [Aria Name](#) .

## What's the goal for the current version?

The current version of Aria is 1.5. The goal of this release is to develop a crash-safe alternative to MyISAM. That is, when MariaDB restarts after a crash, Aria recovers all tables to the state as of the start of a statement or at the start of the last

LOCK TABLES  
statement.

The current goal is to keep the code stable and fix all bugs.

# What's the goal for the next version?

The next version of Aria is 2.0. The goal for this release is to develop a fully transactional storage engine with at least all the major features of InnoDB.

Currently, Aria 2.0 is on hold as its developers are focusing on improving MariaDB. However, they are interested in working with interested customers and partners to add more features to Aria and eventually release 2.0.

These are some of the goals for Aria 2.0:

- ACID compliant
- Commit/Rollback
- Concurrent updates/deletes
- Row locking
- Group commit (Already in [MariaDB 5.2](#) )
- Faster lookup in index pages (Page directory)

Beginning in Aria 2.5, the plan is to focus on improving performance.

# What is the ultimate goal of Aria?

Long term, we have the following goals for Aria:

- To create a new, ACID and Multi-Version Concurrency Control (MVCC), transactional storage engine that can function as both the default non-transactional and transactional storage engine for MariaDB and MySQL ® .
- To be a MyISAM replacement. This is possible because Aria can also be run in non-transactional mode, supports the same row formats as MyISAM, and supports or will support all major features of MyISAM.
- To be the default non-transactional engine in MariaDB (instead of MyISAM).

# What are the design goals in Aria?

- Multi-Version Concurrency Control (MVCC) and ACID storage engine.
- Optionally non-transactional tables that should be 'as fast and as compact' as MyISAM tables.
- Be able to use Aria for internal temporary tables in MariaDB (instead of MyISAM).
- All indexes should have equal speed (clustered index is not on our current road map for Aria. If you need clustered index, you should use XtraDB).
- Allow 'any' length transactions to work (Having long running transactions will cause more log space to be used).
- Allow log shipping; that is, you can do incremental backups of Aria tables just by copying the Aria logs.
- Allow copying of Aria tables between different Aria servers (under some well-defined constraints).
- Better blob handling (than is currently offered in MyISAM, at a minimum).
- No memory copying or extra memory used for blobs on insert/update.
- Blobs allocated in big sequential blocks - Less fragmentation over time.
- Blobs are stored so that Aria can easily be extended to have access to any part of a blob with a single fetch in the future.
- Efficient storage on disk (that is, low row data overhead, low page data overhead and little lost space on pages). Note: There is still some more work to succeed with this goal. The disk layout is fine, but we need more in-memory caches to ensure that we get a higher fill factor on the pages.
- Small footprint, to make MariaDB + Aria suitable for desktop and embedded applications.
- Flexible memory allocation and scalable algorithms to utilize large amounts of memory efficiently, when it is available.

# Where can I find documentation and help about Aria?

Documentation is available at [Aria](#) and related topics. The project is maintained on [GitHub](#) .

If you want to know what happens or be part of developing Aria, you can subscribe to the [maria-developers](#) , [maria-docs](#) , or [maria-discuss](#) groups on Launchpad.

To report and check bugs in Aria, see [Reporting Bugs](#) .

You can usually find some of the Maria developers on our Zulip instance at <https://mariadb.zulipchat.com> or on the IRC channel #maria at <https://libera.chat/> .

# Who develops Aria?

The Core Team who develop Aria are:

## Technical lead

- Michael "Monty" Widenius - Creator of MySQL and MyISAM

## Core Developers (in alphabetical order)

- Guilhem Bichot - Replication expert, on line backup for MyISAM, etc.
- Kristian Nielsen - MySQL build tools, NDB, MySQL server
- Oleksandr Byelkin - Query cache, sub-queries, views.
- Sergei Golubchik - Server Architect, Full text search, keys for MyISAM-Merge, Plugin architecture, etc.

All except Guilhem Bichot are working for [MariaDB Corporation Ab](#) .

# What is the release policy/schedule of Aria?

Aria follows the same [release criteria](#) as for [MariaDB](#). Some clarifications, unique for the Aria storage engine:

- Aria index and data file formats should be backwards and forwards compatible to ensure easy upgrades and downgrades.
- The log file format should also be compatible, but we don't make any guarantees yet. In some cases when upgrading, you must remove the old `aria_log.%`  
and  
`maria_log.%`  
files before restarting MariaDB. (So far, this has only occurred in the upgrade from [MariaDB 5.1](#) and [MariaDB 5.2](#)).

## Extended commitment for Beta 1.5

- Aria is now feature complete according to specification.

## How does Aria 1.5 Compare to MyISAM?

Aria 1.0 was basically a crash-safe non-transactional version of MyISAM. Aria 1.5 added more concurrency (multiple inserter) and some optimizations.

Aria supports all aspects of MyISAM, except as noted below. This includes external and internal check/repair/compressing of rows, different row formats, different index compress formats,

`aria_chk`

etc. After a normal shutdown you can copy Aria files between servers.

## Advantages of Aria compared to MyISAM

- Data and indexes are crash safe.
- On a crash, changes will be rolled back to state of the start of a statement or a last `LOCK TABLES`  
statement.
- Aria can replay almost everything from the log. (Including

`CREATE`

,

`DROP`

,

`RENAME`

,

`TRUNCATE`

tables). Therefore, you make a backup of Aria by just copying the log. The things that can't be replayed (yet) are:

- Batch

`INSERT`

into an empty table (This includes  
`LOAD DATA INFILE`

,

`SELECT... INSERT`  
and

`INSERT`

(many rows)).

- 

`ALTER TABLE`

. Note that  
.frm

tables are NOT recreated!

- 

`LOAD INDEX`

can skip index blocks for unwanted indexes.

- Supports all MyISAM

`ROW`

formats and new

`PAGE`

format where data is stored in pages. (default size is 8K).

- Multiple concurrent inserters into the same table.

- When using

`PAGE`

format (default) row data is cached by page cache.

- Aria has unit tests of most parts.

- Supports both crash-safe (soon to be transactional) and not transactional tables. (Non-transactional tables are not logged and rows uses less space):

- CREATE TABLE foo (...) TRANSACTIONAL=0|1 ENGINE=Aria
- PAGE  
is the only crash-safe/transactional row format.
- PAGE  
format should give a notable speed improvement on systems which have bad data caching. (For example Windows).
- From MariaDB 10.5, max key length is 2000 bytes, compared to 1000 bytes in MyISAM.

## Differences between Aria and MyISAM

- Aria uses BIG (1G by default) log files.
- Aria has a log control file (
 

```
aria_log_control
```

 ) and log files (
 

```
aria_log.%
```

 ). The log files can be automatically purged when not needed or purged on demand (after backup).
- Aria uses 8K pages by default (MyISAM uses 1K). This makes Aria a bit faster when using keys of fixed size, but slower when using variable-length packed keys (until we add a directory to index pages).

## Disadvantages of Aria compared to MyISAM

- Aria doesn't support
 

```
INSERT DELAYED
```
- Aria does not support multiple key caches.
- Storage of very small rows (< 25 bytes) are not efficient for
 

```
PAGE
```

 format.
- MERGE  
tables don't support Aria (should be very easy to add later).
- Aria data pages in block format have an overhead of 10 bytes/page and 5 bytes/row. Transaction and multiple concurrent-writer support will use an extra overhead of 7 bytes for new rows, 14 bytes for deleted rows and 0 bytes for old compacted rows.
- No external locking (MyISAM has external locking, but this is a rarely used feature).
- Aria has one page size for both index and data (defined when Aria is used the first time). MyISAM supports different page sizes per index.
- Small overhead (15 bytes) per index page.
- Aria doesn't support MySQL internal RAID (disabled in MyISAM too, it's a deprecated feature).
- Minimum data file size for PAGE format is 16K (with 8K pages).
- Aria doesn't support indexes on virtual fields.

## Differences between MariaDB 5.1 release and the normal MySQL-5.1 release?

See:

- [Aria storage engine](#)
- [MariaDB versus MySQL](#)

## Why do you use the TRANSACTIONAL keyword now when Aria is not yet transactional?

In the current development phase Aria tables created with

```
TRANSACTIONAL=1
are crash safe and atomic but not transactional because changes in Aria tables can't be rolled back with the
ROLLBACK
command. As we planned to make Aria tables fully transactional, we decided it was better to use the
TRANSACTIONAL
keyword from the start so that applications don't need to be changed later.
```

## What are the known problems with the MySQL-5.1-Maria release?

- See
 

```
KNOWN_BUGS.txt
```

 for open/design bugs.

- See jira.mariadb.org for newly reported bugs. Please report anything you can't find here!
- If there is a bug in the Aria recovery code or in the code that generates the logs, or if the logs become corrupted, then mysqld may fail to start because Aria can't execute the logs at start up.
- Query cache and concurrent insert using page row format have a bug, please disable query cache while using page row format and [MDEV-6817](#) isn't complete

If Aria doesn't start or you have an unrecoverable table (shouldn't happen):

- Remove the `aria_log.%` files from the data directory.

- Restart `mysqld` and run

[CHECK TABLE](#)

,

[REPAIR TABLE](#)

or

`mysqlcheck`

on your Aria tables.

Alternatively,

- Remove logs and run

`aria_chk`

on your `*.MAI` files.

## What is going to change in later Aria main releases?

The

`LOCK TABLES`  
statement will not start a crash-safe segment. You should use

`BEGIN`

and

`COMMIT`

instead.

To make things future safe, you could do this:

```
BEGIN;
LOCK TABLES ....
UNLOCK TABLES;
COMMIT;
```

And later you can just remove the

`LOCK TABLES`  
and  
`UNLOCK TABLES`  
statements.

## How can I create a MyISAM-like (non-transactional) table in Aria?

Example:

```
CREATE TABLE t1 (a int) ROW_FORMAT=FIXED TRANSACTIONAL=0 PAGE_CHECKSUM=0;
CREATE TABLE t2 (a int) ROW_FORMAT=DYNAMIC TRANSACTIONAL=0 PAGE_CHECKSUM=0;
SHOW CREATE TABLE t1;
SHOW CREATE TABLE t2;
```

Note that the rows are not cached in the page cache for

FIXED

or

DYNAMIC

format. If you want to have the data cached (something MyISAM doesn't support) you should use

ROW\_FORMAT=PAGE

:

```
CREATE TABLE t3 (a int) ROW_FORMAT=PAGE TRANSACTIONAL=0 PAGE_CHECKSUM=0;
SHOW CREATE TABLE t3;
```

You can use

PAGE\_CHECKSUM=1

also for non-transactional tables; This puts a page checksums on all index pages. It also puts a checksum on data pages if you use  
ROW\_FORMAT=PAGE

You may still have a speed difference (may be slightly positive or negative) between MyISAM and Aria because of different page sizes. You can change the page size for MariaDB with

```
--aria-block-size=\n#, where\n\\# is 1024, 2048, 4096, 8192, 16384 or 32768.
```

Note that if you change the page size you have to dump all your old tables into text (with

mysqldump

) and remove the old Aria log and files:

```
# rm datadir/aria_log*
```

## What are the advantages/disadvantages of the new PAGE format compared to the old MyISAM-like row formats ( DYNAMIC and FIXED )

The MyISAM-like

DYNAMIC

and

FIXED

format are extremely simple and have very little space overhead, so it's hard to beat them for when it comes to simple scanning of unmodified data. The

DYNAMIC

format does however get notably worse over time if you update the row a lot in a manner that increases the size of the row.

The advantages of the

PAGE

format (compared to

DYNAMIC

or

FIXED

) for non-transactional tables are:

- It's cached by the Page Cache, which gives better random performance (as it uses less system calls).
- Does not fragment as easily as the

DYNAMIC

format during

UPDATE

statements. The maximum number of fragments are very low.

- Code can easily be extended to only read the accessed columns (for example to skip reading blobs).
- Faster updates (compared to

DYNAMIC

).

The disadvantages are:

- Slight storage overhead (should only be notable for very small row sizes)
- Slower full table scan time.
- When using

```
row_format=PAGE
, (the default), Aria first writes the row, then the keys, at which point the check for duplicate keys happens. This makes
PAGE
format slower than
DYNAMIC
(or MyISAM) if there is a lot of duplicated keys because of the overhead of writing and removing the row. If this is a problem, you can use
row_format=DYNAMIC
to get same behavior as MyISAM.
```

## What's the proper way to copy a Aria table from one place to another?

An Aria table consists of 3 files:

```
XXX.frm : The definition for the table, used by MySQL.
XXX.MYI : Aria internal information about the structure of the data and index and data for all indexes.
XXX.MAD : The data.
```

It's safe to copy all the Aria files to another directory or MariaDB instance if any of the following holds:

- If you shutdown the MariaDB Server properly with

```
mysqladmin shutdown
```

, so that there is nothing for Aria to recover when it starts.

or

- If you have run a

```
FLUSH TABLES
```

statement and not accessed the table using SQL from that time until the tables have been copied.

In addition, you must adhere the following rule for transactional tables:

You can't copy the table to a location within the same MariaDB server if the new table has existed before and the new table is still active in the Aria recovery log (that is, Aria may need to access the old data during recovery). If you are unsure whether the old name existed, run

```
aria_chk --zerofill
```

on the table before you use it.

After copying a transactional table and before you use the table, we recommend that you run the command:

```
$ aria_chk --zerofill table_name
```

This will overwrite all references to the logs (LSN), all transactional references (TRN) and all unused space with 0. It also marks the table as 'movable'. An additional benefit of zerofill is that the Aria files will compress better. No real data is ever removed as part of zerofill.

Aria will automatically notice if you have copied a table from another system and do 'zerofill' for the first access of the table if it was not marked as 'movable'. The reason for using

```
aria_chk --zerofill
```

is that you avoid a delay in the MariaDB server for the first access of the table.

Note that this automatic detection doesn't work if you copy tables within the same MariaDB server!

## When is it safe to remove old log files?

If you want to remove the Aria log files (

```
aria_log.%
) with
rm
or delete, then you must first shut down MariaDB cleanly (for example, with
```

```
mysqladmin shutdown
```

) before deleting the old files.

The same rules apply when upgrading MariaDB; When upgrading, first take down MariaDB in a clean way and then upgrade. This will allow you to remove the old log files if there are incompatible problems between releases.

Don't remove the

`aria_log_control`

file! This is not a log file, but a file that contains information about the Aria setup (current transaction id, unique id, next log file number etc.).

If you do, Aria will generate a new

`aria_log_control`

file at startup and will regard all old Aria files as files moved from another system. This means that they have to be 'zerofilled' before they can be used. This will happen automatically at next access of the Aria files, which can take some time if the files are big.

If this happens, you will see things like this in your mysqld.err file:

```
[Note] Zerofilling moved table: '.\database\xxxx'
```

As part of zerofilling no vital data is removed.

## 4.3.4.4 Aria Storage Formats

### Contents

1. [Fixed-length](#)
2. [Dynamic](#)
3. [Page](#)
4. [Transactional](#)

The [Aria](#) storage engine supports three different table storage formats.

These are FIXED, DYNAMIC and PAGE, and they can be set with the ROW FORMAT option in the [CREATE TABLE](#) statement. PAGE is the default format, while FIXED and DYNAMIC are essentially the same as the [MyISAM formats](#).

The [SHOW TABLE STATUS](#) statement can be used to see the storage format used by a table.

### Fixed-length

Fixed-length (or static) tables contain records of a fixed-length. Each column is the same length for all records, regardless of the actual contents. It is the default format if a table has no BLOB, TEXT, VARCHAR or VARBINARY fields, and no ROW FORMAT is provided. You can also specify a fixed table with ROW\_FORMAT=FIXED in the table definition.

Tables containing BLOB or TEXT fields cannot be FIXED, as by design these are both dynamic fields.

Fixed-length tables have a number of characteristics

- fast, since MariaDB will always know where a record begins
- easy to cache
- take up more space than dynamic tables, as the maximum amount of storage space will be allocated to each record.
- reconstructing after a crash is uncomplicated due to the fixed positions
- no fragmentation or need to re-organize, unless records have been deleted and you want to free the space up.

### Dynamic

Dynamic tables contain records of a variable length. It is the default format if a table has any BLOB, TEXT, VARCHAR or VARBINARY fields, and no ROW FORMAT is provided. You can also specify a DYNAMIC table with ROW\_FORMAT=DYNAMIC in the table definition.

Dynamic tables have a number of characteristics

- Each row contains a header indicating the length of the row.
- Rows tend to become fragmented easily. UPDATING a record to be longer will likely ensure it is stored in different places on the disk.
- All string columns with a length of four or more are dynamic.
- They require much less space than fixed-length tables.
- Restoring after a crash is more complicated than with FIXED tables.

### Page

Page format is the default format for Aria tables, and is the only format that can be used if TRANSACTIONAL=1.

Page tables have a number of characteristics:

- It's cached by the page cache, which gives better random performance as it uses fewer system calls.
- Does not fragment as easily as the DYNAMIC format during UPDATES. The maximum number of fragments are very low.
- Updates more quickly than dynamic tables.

- Has a slight storage overhead, mainly notable on very small rows
- Slower to perform a full table scan
- Slower if there are multiple duplicated keys, as Aria will first write a row, then keys, and only then check for duplicates

## Transactional

See [Aria Storage Engine](#) for the impact of the TRANSACTIONAL option on the row format.

### 4.3.4.5 Aria Status Variables

#### Contents

1. [Aria\\_pagecache\\_blocks\\_not\\_flushed](#)
2. [Aria\\_pagecache\\_blocks\\_unused](#)
3. [Aria\\_pagecache\\_blocks\\_used](#)
4. [Aria\\_pagecache\\_read\\_requests](#)
5. [Aria\\_pagecache\\_reads](#)
6. [Aria\\_pagecache\\_write\\_requests](#)
7. [Aria\\_pagecache\\_writes](#)
8. [Aria\\_transaction\\_log\\_syncs](#)

This page documents status variables related to the [Aria storage engine](#). See [Server Status Variables](#) for a complete list of status variables that can be viewed with `SHOW STATUS`.

See also the [Full list of MariaDB options, system and status variables](#).

#### Aria\_pagecache\_blocks\_not\_flushed

- **Description:** The number of dirty blocks in the Aria page cache. The global value can be flushed by

[FLUSH STATUS](#)

- **Scope:** Global

- **Data Type:**

numeric

---

#### Aria\_pagecache\_blocks\_unused

- **Description:** Free blocks in the Aria page cache. The global value can be flushed by

[FLUSH STATUS](#)

- **Scope:** Global

- **Data Type:**

numeric

---

#### Aria\_pagecache\_blocks\_used

- **Description:** Blocks used in the Aria page cache. The global value can be flushed by

[FLUSH STATUS](#)

- **Scope:** Global

- **Data Type:**

numeric

#### Aria\_pagecache\_read\_requests

- **Description:** The number of requests to read something from the Aria page cache.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### Aria\_pagecache\_reads

- **Description:** The number of Aria page cache read requests that caused a block to be read from the disk.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### Aria\_pagecache\_write\_requests

- **Description:** The number of requests to write a block to the Aria page cache.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### Aria\_pagecache\_writes

- **Description:** The number of blocks written to disk from the Aria page cache.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

#### Aria\_transaction\_log\_syncs

- **Description:** The number of Aria log fsyncs.
  - **Scope:** Global
  - **Data Type:**  
    numeric
- 

### 4.3.4.6 Aria System Variables

## Contents

1. [aria\\_block\\_size](#)
2. [aria\\_checkpoint\\_interval](#)
3. [aria\\_checkpoint\\_log\\_activity](#)
4. [aria\\_encrypt\\_tables](#)
5. [aria\\_force\\_start\\_after\\_recovery\\_failures](#)
6. [aria\\_group\\_commit](#)
7. [aria\\_group\\_commit\\_interval](#)
8. [aria\\_log\\_file\\_size](#)
9. [aria\\_log\\_purge\\_type](#)
10. [aria\\_max\\_sort\\_file\\_size](#)
11. [aria\\_page\\_checksum](#)
12. [aria\\_pagedcache\\_age\\_threshold](#)
13. [aria\\_pagedcache\\_buffer\\_size](#)
14. [aria\\_pagedcache\\_division\\_limit](#)
15. [aria\\_pagedcache\\_file\\_hash\\_size](#)
16. [aria\\_recover](#)
17. [aria\\_recover\\_options](#)
18. [aria\\_repair\\_threads](#)
19. [aria\\_sort\\_buffer\\_size](#)
20. [aria\\_stats\\_method](#)
21. [aria\\_sync\\_log\\_dir](#)
22. [aria\\_used\\_for\\_temp\\_tables](#)
23. [deadlock\\_search\\_depth\\_long](#)
24. [deadlock\\_search\\_depth\\_short](#)
25. [deadlock\\_timeout\\_long](#)
26. [deadlock\\_timeout\\_short](#)

This page documents system variables related to the [Aria storage engine](#). For options that are not system variables, see [Aria Options](#).

See [Server System Variables](#) for a complete list of system variables and instructions on setting system variables.

Also see the [Full list of MariaDB options, system and status variables](#).

### aria\_block\_size

- **Description:** Block size to be used for Aria index pages. Changing this requires dumping, deleting old tables and deleting all log files, and then restoring your Aria tables. If key lookups take too long (and one has to search roughly 8192/2 by default to find each key), can be made smaller, e.g.

2048  
or  
4096

- **Commandline:**

--aria-block-size=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**

numeric

- **Default Value:**

8192

### aria\_checkpoint\_interval

- **Description:** Interval in seconds between automatic checkpoints. 0 means 'no automatic checkpoints' which makes sense only for testing.
- **Commandline:**

--aria-checkpoint-interval=#

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**

numeric

- **Default Value:**

- **Range:**

```
0
to
4294967295
```

---

### `aria_checkpoint_log_activity`

- **Description:** Number of bytes that the transaction log has to grow between checkpoints before a new checkpoint is written to the log.
- **Commandline:**

```
aria-checkpoint-log-activity=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
numeric
```

- **Default Value:**

```
1048576
```

- **Range**

```
0
to
4294967295
```

---

### `aria_encrypt_tables`

- **Description:** Enables automatic encryption of all user-created Aria tables that have the

`ROW_FORMAT`

table option set to

`PAGE`

. See [Data at Rest Encryption](#) and [Enabling Encryption for User-created Tables](#) .

- **Commandline:**

```
aria-encrypt-tables={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
boolean
```

- **Default Value:**

```
OFF
```

- **Introduced:** [MariaDB 10.1.3](#)

---

### `aria_force_start_after_recovery_failures`

- **Description:** Number of consecutive log recovery failures after which logs will be automatically deleted to cure the problem; 0 (the default) disables the feature.

- **Commandline:**

```
--aria-force-start-after-recovery-failures=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

0

---

## aria\_group\_commit

- **Description:** Specifies Aria [group commit mode](#).

- **Commandline:**

--aria\_group\_commit="value"

- **Alias:**

maria\_group\_commit

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

- **Valid values:**

◦

none

- *Group commit is disabled.*

◦

hard

- *Wait the number of microseconds specified by aria\_group\_commit\_interval before actually doing the commit. If the interval is 0 then just check if any other threads have requested a commit during the time this commit was preparing (just before sync() file) and send their data to disk also before sync().*

◦

soft

- *The service thread will wait the specified time and then sync() to the log. If the interval is 0 then it won't wait for any commits (this is dangerous and should generally not be used in production)*

- **Default Value:**

none

---

## aria\_group\_commit\_interval

- **Description:** Interval between [Aria group commits](#) in microseconds (1/1000000 second) for other threads to come and do a commit in "hard" mode and sync()/commit at all in "soft" mode. Option only has effect if [aria\\_group\\_commit](#) is used.

- **Commandline:**

--aria\_group\_commit\_interval=#

- **Alias:**

maria\_group\_commit\_interval

- **Scope:** Global

- **Dynamic:** No

- **Type:** numeric

- **Valid Values:**

- **Default Value:**

0

(no waiting)

- **Range:**

0-4294967295

---

## aria\_log\_file\_size

- **Description:** Limit for Aria transaction log size

- **Commandline:**

```
--aria-log-file-size=#
```

- **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    numeric
  - **Default Value:**  
    1073741824
- 

### aria\_log\_purge\_type

- **Description:** Specifies how the Aria transactional log will be purged. Set to  
    at\_flush  
    to keep a copy of the transaction logs (good as an extra backup). The logs will stay until the next **FLUSH LOGS** ;

- **Commandline:**  
    --aria-log-purge-type=name

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
    enumeration

- **Default Value:**  
    immediate

- **Valid Values:**  
    immediate  
    ,  
    external  
    ,  
    at\_flush
- 

### aria\_max\_sort\_file\_size

- **Description:** Don't use the fast sort index method to created index if the temporary file would get bigger than this.
- **Commandline:**  
    --aria-max-sort-file-size=#

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
    numeric

- **Default Value:**  
    9223372036853727232

- **Range:**  
    0  
    to  
    9223372036854775807
- 

### aria\_page\_checksum

- **Description:** Determines whether index and data should use page checksums for extra safety. Can be overridden per table with **PAGE\_CHECKSUM** clause in **CREATE TABLE** .

- **Commandline:**  
    --aria-page-checksum=#

- **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
- **Default Value:**  
ON
- 

### aria\_pagecache\_age\_threshold

- **Description:** This characterizes the number of hits a hot block has to be untouched until it is considered aged enough to be downgraded to a warm block. This specifies the percentage ratio of that number of hits to the total number of blocks in the page cache.
  - **Commandline:**  
--aria-pagecache-age-threshold=#
- **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
numeric
- **Default Value:**  
300
- **Range:**  
100  
to  
9999900
- 

### aria\_pagecache\_buffer\_size

- **Description:** The size of the buffer used for index and data blocks for Aria tables. This can include explicit Aria tables, system tables, and temporary tables. Increase this to get better handling and measure by looking at [aria-status-variables/#aria\\_pagecache\\_reads](#) (should be small) vs [aria-status-variables/#aria\\_pagecache\\_read\\_requests](#) .
  - **Commandline:**  
--aria-pagecache-buffer-size=#
- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
numeric
- **Default Value:**  
134217720  
(128MB)
- **Range:**  
131072  
(128KB) upwards
- 

### aria\_pagecache\_division\_limit

- **Description:** The minimum percentage of warm blocks in the key cache.
  - **Commandline:**  
--aria-pagecache-division-limit=#
- **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
numeric
- **Default Value:**

100

- **Range:**

1  
to  
100

---

## aria\_pagecache\_file\_hash\_size

- **Description:** Number of hash buckets for open and changed files. If you have many Aria files open you should increase this for faster flushing of changes. A good value is probably 1/10th of the number of possible open Aria files.

- **Commandline:**

--aria-pagecache-file-hash-size=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

512

- **Range:**

128  
to  
16384

- **Introduced:** MariaDB 10.0.13
- 

## aria\_recover

- **Description:**

aria\_recover  
has been renamed to  
aria\_recover\_options  
in MariaDB 10.2.0 . See [aria\\_recover\\_options](#) for the description.

---

## aria\_recover\_options

- **Description:** Specifies how corrupted tables should be automatically repaired. More than one option can be specified, for example

FORCE, BACKUP

.

NORMAL  
: Normal automatic repair, the default until MariaDB 10.2.3

o

OFF  
: Autorecovery is disabled, the equivalent of not using the option

o

QUICK  
: Does not check rows in the table if there are no delete blocks.

o

FORCE  
: Runs the recovery even if it determines that more than one row from the data file will be lost.

o

BACKUP  
: Keeps a backup of the data files.

- **Commandline:**

--aria-recover-options[=#]

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**  
enumeration

- **Default Value:**

- BACKUP, QUICK  
( $\geq$  MariaDB 10.2.4 )
- NORMAL  
( $\leq$  MariaDB 10.2.3 )

- **Valid Values:**

- NORMAL
- ,
- BACKUP
- ,
- FORCE
- ,
- QUICK
- ,
- OFF

- **Introduced:** MariaDB 10.2.0
- 

### aria\_repair\_threads

- **Description:** Number of threads to use when repairing Aria tables. The value of 1 disables parallel repair. Increasing from the default will usually result in faster repair, but will use more CPU and memory.

- **Commandline:**

- aria-repair-threads=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

- numeric

- **Default Value:**

- 1

---

### aria\_sort\_buffer\_size

- **Description:** The buffer that is allocated when sorting the index when doing a [REPAIR](#) or when creating indexes with [CREATE INDEX](#) or [ALTER TABLE](#).

- **Commandline:**

- aria-sort-buffer-size=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

- numeric

- **Default Value:**

- 268434432

- (from MariaDB 10.0.3 ),

- 134217728

- (before MariaDB 10.0.3 )

---

### aria\_stats\_method

- **Description:** Determines how NULLs are treated for Aria index statistics purposes. If set to

- nulls\_equal

- , all NULL index values are treated as a single group. This is usually fine, but if you have large numbers of NULLs the average group size

is slanted higher, and the optimizer may miss using the index for ref accesses when it would be useful. If set to

`nulls_unequal`

, the default, the opposite approach is taken, with each NULL forming its own group of one. Conversely, the average group size is slanted lower, and the optimizer may use the index for ref accesses when not suitable. Setting to

`nulls_ignored`

ignores NULLs altogether from index group calculations. Statistics need to be recalculated after this method is changed. See also [Index Statistics](#) , [myisam\\_stats\\_method](#) and [innodb\\_stats\\_method](#) .

- **Commandline:**

```
--aria-stats-method=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

`numeric`

- **Default Value:**

`nulls_unequal`

- **Valid Values:**

`nulls_equal`

,

`nulls_unequal`

,

`nulls_ignored`

## `aria_sync_log_dir`

- **Description:** Controls syncing directory after log file growth and new file creation.

- **Commandline:**

```
--aria-sync-log-dir=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

`enumeration`

- **Default Value:**

`NEWFILE`

- **Valid Values:**

`NEWFILE`

,

`NEVER`

,

`ALWAYS`

## `aria_used_for_temp_tables`

- **Description:** Readonly variable indicating whether the [Aria](#) storage engine is used for temporary tables. If set to

`ON`

, the default, the Aria storage engine is used. If set to

`OFF`

, MariaDB reverts to using [MyISAM](#) for on-disk temporary tables. The [MEMORY](#) storage engine is used for temporary tables regardless of this variable's setting where appropriate. The default can be changed by not using the

`--with-aria-tmp-tables`

option when building MariaDB.

- **Commandline:** No

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

`boolean`

- **Default Value:**

`ON`

---

## deadlock\_search\_depth\_long

- **Description:** Long search depth for the [two-step deadlock detection](#). Only used by the [Aria](#) storage engine.
- **Commandline:**

```
--deadlock-search-depth-long=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    15
```

- **Range:**

```
    0
```

```
    to
```

```
    33
```

---

## deadlock\_search\_depth\_short

- **Description:** Short search depth for the [two-step deadlock detection](#). Only used by the [Aria](#) storage engine.
- **Commandline:**

```
--deadlock-search-depth-short=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    4
```

- **Range:**

```
    0
```

```
    to
```

```
    32
```

---

## deadlock\_timeout\_long

- **Description:** Long timeout in microseconds for the [two-step deadlock detection](#). Only used by the [Aria](#) storage engine.
- **Commandline:**

```
--deadlock-timeout-long=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    50000000
```

- **Range:**

```
    0
```

```
    to
```

```
    4294967295
```

```
deadlock_timeout_short
```

- **Description:** Short timeout in microseconds for the [two-step deadlock detection](#). Only used by the [Aria storage engine](#).

- **Commandline:**

```
--deadlock-timeout-short=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
numeric
```

- **Default Value:**

```
10000
```

- **Range:**

```
0
```

```
to
```

```
4294967295
```

## 4.3.4.7 Aria Group Commit

Since [MariaDB 5.2](#), the [Aria storage engine](#) has included a feature to group commits to speed up concurrent threads doing many inserts into the same or different Aria tables.

By default, group commit for Aria is turned off. It is controlled by the [aria\\_group\\_commit](#) and [aria\\_group\\_commit\\_interval](#) system variables.

Information on setting server variables can be found on the [Server System Variables](#) page.

## Terminology

- A

```
commit  
is  
flush of logs  
followed by a sync.
```

- 

```
sent to disk  
means written to disk but not sync()ed,
```

- 

```
flushed  
mean sent to disk and synced().
```

- 

```
LSN  
means log serial number. It's refers to the position in the transaction log.
```

## Non Group commit logic (aria\_group\_commit="none")

The thread which first started the

```
commit
```

is performing the actual flush of logs. Other threads set the new goal (LSN) of the next pass (if it is maximum) and wait for the pass end or just wait for the pass end.

The effect of this is that a flush (write of logs + sync) will save all data for all threads/transactions that have been waiting since the last flush.

## If hard group commit is enabled (aria\_group\_commit="hard")

### If hard commit and aria\_group\_commit\_interval=0

The first thread sends all changed buffers to disk. This is repeated as long as there are new LSNs added. The process can not loop forever because we have a limited number of threads and they will wait for the data to be synced.

Pseudo code:

```

do
    send changed buffers to disk
    while new_goal
    sync

```

## If hard commit and aria\_group\_commit\_interval > 0

If less than rate microseconds has passed since the last sync, then after buffers have been sent to disk, wait until rate microseconds has passed since last sync, do sync and return. This ensures that if we call sync infrequently we don't do any waits.

## If soft group commit is enabled (aria\_group\_commit="soft")

Note that soft group commit should only be used if you can afford to lose a few rows if your machine shuts down hard (as in the case of a power failure).

Works like in

```
non group commit'
```

but the thread doesn't do any real sync(). If aria\_group\_commit\_interval is not zero, the sync() will be performed by a service thread with the given rate when needed (new LSN appears). If aria\_group\_commit\_interval is zero, there will be no sync() calls.

## Code

The code for this can be found in storage/maria/ma\_loghandler.c::translog\_flush()

### 4.3.4.8 Benchmarking Aria

We have not yet had time to benchmark [Aria](#) properly. Here follows some things that have been discussed on the [maria-discuss](#) email list.

## Aria used for internal temporary tables

By default Aria (instead of MyISAM) is used for the internal temporary tables when MEMORY tables overflows to disk or MEMORY tables can't be used (for example when you are using temporary results with BLOB's). In most cases Aria should give you better performance than using MyISAM, but this is not always the case.

```

CREATE TABLE `t1` (`id` int(11) DEFAULT NULL, `tea` text)
ENGINE=MyISAM DEFAULT CHARSET=latin1;
insert t1 select rand()*2e8, repeat(rand(), rand()*64) from t1;

```

Repeat the last row until you get 2097152 rows.

The queries tested

```

Q1: SELECT id, tea from t1 group by left(id,1) order by null;
Q2: SELECT id, count(*), tea from t1 group by left(id,1) order by null;
Q3: SELECT id, tea from t1 group by left(id,2) order by null;
Q4: SELECT id, count(*), tea from t1 group by left(id,2) order by null;
Q5: SELECT id, tea from t1 group by id % 100 order by null;
Q6: SELECT id, count(*), tea from t1 group by id % 100 order by null;

```

Results (times in seconds, lower is better):

| Test | Aria 8K page size | Aria 2K page size | MyISAM |
|------|-------------------|-------------------|--------|
| Q1   | 3.08              | 2.41              | 2.17   |
| Q2   | 6.24              | 5.21              | 12.89  |
| Q3   | 4.87              | 4.05              | 4.04   |
| Q4   | 8.20              | 7.04              | 15.14  |
| Q5   | 7.10              | 6.37              | 6.28   |
| Q6   | 10.38             | 9.09              | 17.00  |

The good news is that for common group by queries that is using summary functions there is a close to 50 % speedup of using Aria for internal temporary tables.

Note that queries Q1,Q3 and Q5 are not typical queries as there is no sum functions involved. In this case rows are just written to the tmp tables and there is no updates. As soon as there are summary functions and updates the new row format in Aria gives a close to 50 % speedup.

The above table also shows how the page size (determined by the [aria\\_block\\_size](#) system variable) affects the performance. The reason for the difference is that there is more data to move back/from the page cache for inserting of keys. (When reading data we are normally not copying pages). The bigger page size however allows longer keys and fewer index levels so for bigger data sets the different should be smaller. It's possible to in the future

optimize Aria to not copy pages from the page cache also for index writes and then this difference should disappear.

The default page size for Aria is 8K.

If you want to run MariaDB with MyISAM for temporary tables, don't use the configure option '--with-aria-tmp-tables' when building MariaDB.

## 4.3.4.9 Aria Two-step Deadlock Detection

### Description

The [Aria storage engine](#) can automatically detect and deal with deadlocks (see the [Wikipedia deadlocks article](#) ).

This feature is controlled by four configuration variables, two that control the search depth and two that control the timeout.

- `deadlock_search_depth_long`
- `deadlock_search_depth_short`
- `deadlock_timeout_long`
- `deadlock_timeout_short`

### How it Works

If Aria is ever unable to obtain a lock, we might have a deadlock. There are two primary ways for detecting if a deadlock has actually occurred. First is to search a wait-for graph (see the [wait-for graph on Wikipedia](#)) and the second is to just wait and let the deadlock exhibit itself. Aria Two-step Deadlock Detection does a combination of both.

First, if the lock request cannot be granted immediately, we do a short search of the wait-for graph with a small search depth as configured by the `deadlock_search_depth_short`

variable. We have a depth limit because the graph can (theoretically) be arbitrarily big and we don't want to recursively search the graph arbitrarily deep. This initial, short search is very fast and most deadlocks will be detected right away. If no deadlock cycles are found with the short search the system waits for the amount of time configured in

`deadlock_timeout_short`

to see if the lock conflicts will be removed and the lock can be granted. Assuming this did not happen and the lock request still waits, the system then moves on to step two, which is a repeat of the process but this time searching deeper using the

`deadlock_search_depth_long`

. If no deadlock has been detected, it waits

`deadlock_timeout_long`

and times out.

When a deadlock is detected the system uses a weighting algorithm to determine which thread in the deadlock should be killed and then kills it.

## 4.3.4.10 Aria Encryption Overview

### Contents

1. [Basic Configuration](#)
2. [Determining Whether a Table is Encrypted](#)
3. [Encryption and the Aria Log](#)

MariaDB can encrypt data in tables that use the [Aria storage engine](#). This includes both user-created tables and internal on-disk temporary tables that use the Aria storage engine. This ensures that your Aria data is only accessible through MariaDB.

For encryption with the InnoDB and XtraDB storage engines, see [Encrypting Data for InnoDB/XtraDB](#).

### Basic Configuration

In order to enable encryption for tables using the [Aria storage engine](#), there are a couple server system variables that you need to set and configure. Most users will want to set

`aria_encrypt_tables`

and

`encrypt_tmp_disk_tables`

Users of data-at-rest encryption will also need to have a [key management and encryption plugin](#) configured. Some examples are [File Key Management Plugin](#) and [AWS Key Management Plugin](#).

```
[mariadb]
...
# File Key Management
plugin_load_add = file_key_management
file_key_management_filename = /etc/mysql/encryption/keyfile.enc
file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
file_key_management_encryption_algorithm = AES_CTR

# Aria Encryption
aria_encrypt_tables=ON
encrypt_tmp_disk_tables=ON
```

## Determining Whether a Table is Encrypted

The InnoDB storage engine has the [information\\_schema.INNODB\\_TABLESPACES\\_ENCRYPTION table](#) that can be used to get information about which tables are encrypted. Aria does not currently have anything like that (see [MDEV-17324](#) about that).

To determine whether an Aria table is encrypted, you currently have to search the data file for some plain text that you know is in the data.

For example, let's say that we have the following table:

```
SELECT * FROM db1.aria_tab LIMIT 1;
+----+----+
| id | str |
+----+----+
| 1  | str1 |
+----+----+
1 row in set (0.00 sec)
```

Then, we could search the data file that belongs to

```
db1.aria_tab
for
str1
using a command-line tool, such as strings :
```

```
$ sudo strings /var/lib/mysql/db1/aria_tab.MAD | grep "str1"
str1
```

If you can find the plain text of the string, then you know that the table is not encrypted.

## Encryption and the Aria Log

Only Aria tables are currently encrypted. The [Aria log](#) is not yet encrypted. See [MDEV-8587](#) about that.

### 4.3.4.11 The Aria Name

#### Contents

1. [Backstory](#)
2. [Renaming Maria \(the engine\)](#)
3. [See Also](#)

The [Aria](#) storage engine used to be called Maria. This page gives the history and background of how and why this name was changed to Aria.

## Backstory

When starting what became the MariaDB project, Monty and the initial developers only planned to work on a next generation MyISAM storage engine replacement. This storage engine would be crash safe and eventually support transactions. Monty named the storage engine, and the project, after his daughter, Maria.

Work began in earnest on the Maria storage engine but the plans quickly expanded and morphed and soon the developers were not just working on a storage engine, but on a complete branch of the MySQL database. Since the project was already called Maria, it made sense to call the whole database server MariaDB.

## Renaming Maria (the engine)

So now there was the database, MariaDB, and the storage engine, Maria. To end the confusion this caused, the decision was made to rename the storage engine.

Monty's first suggestion was to name it *Lucy*, after his dog, but few who heard it liked that idea. So the decision was made that the next best thing was for the community to suggest and vote on names.

This was done by running a contest in 2009 through the end of May 2010. After that the best names were voted on by the community and Monty picked and [announced the winner](#) (Aria) at OSCON 2010 in Portland.

The winning entry was submitted by Chris Tooley. He received a Linux-powered [System 76](#) Meerkat NetTop as his prize for suggesting the winning name from Monty Program.

## See Also

- [Why is the Software Called MariaDB?](#)

### 4.3.5 Archive

The `ARCHIVE` storage engine is a storage engine that uses gzip to compress rows. It is mainly used for storing large amounts of data, without indexes, with only a very small footprint.

A table using the `ARCHIVE` storage engine is stored in two files on disk. There's a table definition file with an extension of `.frm`, and a data file with the extension `.ARZ`. At times during optimization, a `.ARN` file will appear.

New rows are inserted into a compression buffer and are flushed to disk when needed. `SELECT`s cause a flush. Sometimes, rows created by multi-row inserts are not visible until the statement is complete.

`ARCHIVE` allows a maximum of one key. The key must be on an `AUTO_INCREMENT` column, and can be a `PRIMARY KEY` or a non-unique key. However, it has a limitation: it is not possible to insert a value which is lower than the next `AUTO_INCREMENT` value.

#### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Characteristics](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing `INSTALL SONAME` or `INSTALL PLUGIN`. For example:

```
INSTALL SONAME 'ha_archive';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the `--plugin-load` or the `--plugin-load-add` options. This can be specified as a command-line argument to `mysqld` or it can be specified in a relevant server option group in an option file. For example:

```
[mariadb]
...
plugin_load_add = ha_archive
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing `UNINSTALL SONAME` or `UNINSTALL PLUGIN`. For example:

```
UNINSTALL SONAME 'ha_archive';
```

If you installed the plugin by providing the `--plugin-load` or the `--plugin-load-add` options in a relevant server option group in an option file, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Characteristics

- Supports `INSERT` and `SELECT`, but not `DELETE`, `UPDATE` or `REPLACE`.
- Data is compressed with zlib as it is inserted, making it very small.
- Data is slow the select, as it needs to be uncompressed, and, besides the `query cache`, there is no cache.
- Supports `AUTO_INCREMENT` (since MariaDB/MySQL 5.1.6), which can be a unique or a non-unique index.
- Since MariaDB/MySQL 5.1.6, selects scan past BLOB columns unless they are specifically requested, making these queries much more efficient.
- Does not support `spatial` data types.
- Does not support `transactions`.
- Does not support foreign keys.
- Does not support `virtual columns`.
- No storage limit.
- Supports row locking.

- Supports [table discovery](#), and the server can access ARCHIVE tables even if the corresponding .frm file is missing.
- [OPTIMIZE TABLE](#) and [REPAIR TABLE](#) can be used to compress the table in its entirety, resulting in slightly better compression.
- With MariaDB, it is possible to upgrade from the MySQL 5.0 format without having to dump the tables.
- [INSERT DELAYED](#) is supported.
- Running many SELECTs during the insertions can deteriorate the compression, unless only multi-rows INSERTs and [INSERT DELAYED](#) are used.

No items found.

There are [3 related questions](#).

## 4.3.6 BLACKHOLE

The

BLACKHOLE

storage engine accepts data but does not store it and always returns an empty result.

A table using the

BLACKHOLE

storage engine consists of a single .frm table format file, but no associated data or index files.

This storage engine can be useful, for example, if you want to run complex filtering rules on a slave without incurring any overhead on a master. The master can run a

BLACKHOLE

storage engine, with the data replicated to the slave for processing.

### Contents

- [1. Installing the Plugin](#)
- [2. Uninstalling the Plugin](#)
- [3. Using the BLACKHOLE Storage Engine](#)
  - [1. Using with DML](#)
  - [2. Using with Replication](#)
  - [3. Using with Triggers](#)
  - [4. Using with Foreign Keys](#)
  - [5. Using with Virtual Columns](#)
  - [6. Using with AUTO\\_INCREMENT](#)
- [4. Limits](#)
- [5. Examples](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing [INSTALL SONAME](#) or [INSTALL PLUGIN](#). For example:

```
INSTALL SONAME 'ha_blackhole';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the [--plugin-load](#) or the [--plugin-load-add](#) options. This can be specified as a command-line argument to [mysqld](#) or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = ha_blackhole
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing [UNINSTALL SONAME](#) or [UNINSTALL PLUGIN](#). For example:

```
UNINSTALL SONAME 'ha_blackhole';
```

If you installed the plugin by providing the [--plugin-load](#) or the [--plugin-load-add](#) options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Using the BLACKHOLE Storage Engine

### Using with DML

[INSERT](#), [UPDATE](#), and [DELETE](#) statements all work with the BLACKHOLE storage engine. However, no data changes are actually applied.

## Using with Replication

If the binary log is enabled, all SQL statements will be logged as usual, and replicated to any slave servers. However, since rows are not stored, it is important to use statement-based rather than the row or mixed format, as [UPDATE](#) and [DELETE](#) statements are neither logged nor replicated. See [Binary Log Formats](#).

## Using with Triggers

Some [triggers](#) work with the BLACKHOLE storage engine.

BEFORE triggers for [INSERT](#) statements are still activated.

Triggers for [UPDATE](#) and [DELETE](#) statements are **not** activated.

Triggers with the FOR EACH ROW clause do not apply, since the tables have no rows.

## Using with Foreign Keys

Foreign keys are not supported. If you convert an [InnoDB](#) table to BLACKHOLE, then the foreign keys will disappear. If you convert the same table back to InnoDB, then you will have to recreate them.

## Using with Virtual Columns

If you convert an [InnoDB](#) table which contains [virtual columns](#) to BLACKHOLE, then it produces an error.

## Using with AUTO\_INCREMENT

Because a BLACKHOLE table does not store data, it will not maintain the [AUTO\\_INCREMENT](#) value. If you are replicating to a table that can handle [AUTO\\_INCREMENT](#) columns, and are not explicitly setting the primary key auto-increment value in the [INSERT](#) query, or using the [SET INSERT\\_ID](#) statement, inserts will fail on the slave due to duplicate keys.

## Limits

The maximum key size is:

- 3500 bytes (>= [MariaDB 10.1.48](#), [MariaDB 10.2.35](#), [MariaDB 10.3.26](#), [MariaDB 10.4.16](#) and [MariaDB 10.5.7](#))
- 1000 bytes (<= [MariaDB 10.1.47](#), [MariaDB 10.2.34](#), [MariaDB 10.3.25](#), [MariaDB 10.4.15](#) and [MariaDB 10.5.6](#)).

## Examples

```
CREATE TABLE table_name (
    id int unsigned primary key not null,
    v varchar(30)
) ENGINE=BLACKHOLE;

INSERT INTO table_name VALUES (1, 'bob'),(2, 'jane');

SELECT * FROM table_name;
Empty set (0.001 sec)
```

## 4.3.7 CONNECT

Note: You can download a [PDF version of the CONNECT documentation](#) (1.7.0003).

| Connect Version   | Introduced                                                                                                                            | Maturity |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------|----------|
| Connect 1.07.0002 | <a href="#">MariaDB 10.5.9</a> , <a href="#">MariaDB 10.4.18</a> , <a href="#">MariaDB 10.3.28</a> , <a href="#">MariaDB 10.2.36</a>  | Stable   |
| Connect 1.07.0001 | <a href="#">MariaDB 10.4.12</a> , <a href="#">MariaDB 10.3.22</a> , <a href="#">MariaDB 10.2.31</a> , <a href="#">MariaDB 10.1.44</a> | Stable   |
| Connect 1.06.0010 | <a href="#">MariaDB 10.4.8</a> , <a href="#">MariaDB 10.3.18</a> , <a href="#">MariaDB 10.2.27</a>                                    | Stable   |
| Connect 1.06.0007 | <a href="#">MariaDB 10.3.6</a> , <a href="#">MariaDB 10.2.14</a> , <a href="#">MariaDB 10.1.33</a>                                    | Stable   |
| Connect 1.06.0005 | <a href="#">MariaDB 10.3.3</a> , <a href="#">MariaDB 10.2.10</a> , <a href="#">MariaDB 10.1.29</a>                                    | Stable   |
| Connect 1.06.0004 | <a href="#">MariaDB 10.3.2</a> , <a href="#">MariaDB 10.2.9</a> , <a href="#">MariaDB 10.1.28</a>                                     | Stable   |
| Connect 1.06.0001 | <a href="#">MariaDB 10.3.1</a> , <a href="#">MariaDB 10.2.8</a> , <a href="#">MariaDB 10.1.24</a>                                     | Beta     |
| Connect 1.05.0003 | <a href="#">MariaDB 10.3.0</a> , <a href="#">MariaDB 10.2.5</a> , <a href="#">MariaDB 10.1.22</a>                                     | Stable   |
| Connect 1.05.0001 | <a href="#">MariaDB 10.2.4</a> , <a href="#">MariaDB 10.1.21</a>                                                                      | Stable   |
| Connect 1.04.0008 | <a href="#">MariaDB 10.2.2</a> , <a href="#">MariaDB 10.1.17</a>                                                                      | Stable   |
| Connect 1.04.0006 | <a href="#">MariaDB 10.2.0</a> , <a href="#">MariaDB 10.1.13</a> ,                                                                    | Stable   |
| Connect 1.04.0005 | <a href="#">MariaDB 10.1.10</a>                                                                                                       | Beta     |
| Connect 1.04.0003 | <a href="#">MariaDB 10.1.9</a>                                                                                                        | Beta     |

The CONNECT storage engine enables MariaDB to access external local or remote data (MED). This is done by defining tables based on different data types, in particular files in various formats, data extracted from other DBMS or products (such as Excel or MongoDB) via ODBC or JDBC, or data retrieved from the environment (for example DIR, WMI, and MAC tables)

This storage engine supports table partitioning, MariaDB virtual columns and permits defining *special* columns such as ROWID, FILEID, and SERVID.

No precise definition of maturity exists. Because CONNECT handles many table types, each type has a different maturity depending on whether it is old and well-tested, less well-tested or newly implemented. This will be indicated for all data types.



## Introduction to the CONNECT Engine

[Reasons behind the CONNECT storage engine.](#)



## Installing the CONNECT Storage Engine

[Installing the CONNECT storage engine.](#)



## Creating and Dropping CONNECT Tables

[Creating and dropping CONNECT tables.](#)



## CONNECT Data Types

[Data types supported by CONNECT.](#)



## Current Status of the CONNECT Handler

[The current CONNECT handler is a stable release.](#)

# CONNECT Table Types



## CONNECT Table Types Overview

[CONNECT can handle many table formats.](#)



## Inward and Outward Tables

[The two broad categories of CONNECT tables.](#)



## CONNECT Table Types - Data Files

[CONNECT plain DOS or UNIX data files.](#)



## CONNECT Zipped File Tables

[When the table file or files are compressed in one or several zip files.](#)



## CONNECT DOS and FIX Table Types

[CONNECT tables based on text files](#)



## CONNECT DBF Table Type

[CONNECT dBASE III or IV tables.](#)



## CONNECT BIN Table Type

[CONNECT binary files in which each row is a logical record of fixed length](#)



## CONNECT VEC Table Type

*CONNECT binary files organized in vectors*



## CONNECT CSV and FMT Table Types

*Variable length CONNECT data files.*



## CONNECT - NoSQL Table Types

*Based on files that do not match the relational format but often represent hierarchical data.*



## CONNECT - Files Retrieved Using Rest Queries

*JSON, XML and CSV data files can be retrieved as results from REST queries.*



## CONNECT JSON Table Type

*JSON (JavaScript Object Notation) is a widely-used lightweight data-interchange format.*



## CONNECT XML Table Type

*CONNECT XML files*



## CONNECT INI Table Type

*CONNECT INI Windows configuration or initialization files.*



## CONNECT - External Table Types

*Access tables belonging to the current or another server.*



## CONNECT ODBC Table Type: Accessing Tables From Another DBMS

*CONNECT Table Types - ODBC Table Type: Accessing Tables from other DBMS*



## CONNECT JDBC Table Type: Accessing Tables from Another DBMS

*Using JDBC to access other tables.*



## CONNECT MONGO Table Type: Accessing Collections from MongoDB

*Used to directly access MongoDB collections as tables.*



## CONNECT MYSQL Table Type: Accessing MySQL/MariaDB Tables

*Accessing a MySQL or MariaDB table or view*



## CONNECT PROXY Table Type

*Tables that access and read the data of another table or view*



## CONNECT XCOL Table Type

*Based on another table/view, used when object tables have a column that contains a list of values*



## CONNECT OCCUR Table Type

*Extension to the PROXY type when referring to a table/view having several c...*



## CONNECT PIVOT Table Type

*Transform the result of another table into another table along "pivot" and "fact" columns.*



## CONNECT TBL Table Type: Table List

*Define a table as a list of tables of any engine and type.*



## CONNECT - Using the TBL and MYSQL Table Types Together

*Used together, the TBL and MYSQL types lift all the limitations of the FEDERATED and MERGE engines*



## CONNECT Table Types - Special "Virtual" Tables

*VIR, WMI and MAC special table types*



## CONNECT Table Types - VIR

*VIR virtual type for CONNECT*



## CONNECT Table Types - OEM: Implemented in an External LIB

*CONNECT OEM table types are implemented in an external library.*



## CONNECT Table Types - Catalog Tables

*Catalog tables return information about another table or data source*



## Adding DataFlex 3.1c .dat Files As An External Table Type With CONNECT

*I'm using MariaDB's CONNECT engine to access / utilize a set of Visual FoxP...*



## **CONNECT engine windows**

*with mariadb 10.07 installed on windows, how to setup engines, particulary ...*



## **creating pivot table fails**

*I tried to create a pivot table based on an existing table "test1" and get ...*



## **limit of number of columns**

*I have a table of 6MB with 50 values in the pivot-values leading to 50 colu...*

## Other CONNECT Articles



### **CONNECT - Security**

*CONNECT requires the FILE privilege for "outward" tables*



### **CONNECT - OEM Table Example**

*Example showing how an OEM table can be implemented.*

## Using CONNECT



### **Using CONNECT - General Information**

*Using CONNECT - General Information.*



### **Using CONNECT - Virtual and Special Columns**

*Virtual and special columns example usage*



### **Using CONNECT - Importing File Data Into MariaDB Tables**

*Directly using external (file) data has many advantages*



### **Using CONNECT - Exporting Data From MariaDB**

*Exporting data from MariaDB with CONNECT*



### **Using CONNECT - Indexing**

*Indexing with the CONNECT handler*



### **Using CONNECT - Condition Pushdown**

*Using CONNECT - Condition Pushdown.*



### **USING CONNECT - Offline Documentation**

*CONNECT Plugin User Manual.*



### **Using CONNECT - Partitioning and Sharding**

*Partitioning and Sharding with CONNECT*

## Other CONNECT Articles



### **CONNECT - Making the GetRest Library**

*Compiling the function calling the cpprestsdk package separately that will be loaded by CONNECT.*



### **CONNECT - Adding the REST Feature as a Library Called by an OEM Table**

*How the REST feature can be added as a library called by an OEM table.*



### **CONNECT - Compiling JSON UDFs in a Separate Library**

*There are situations when you may need to have JSON UDFs in a separate library.*



### **CONNECT System Variables**

*System variables related to the CONNECT storage engine.*



### **JSON Sample Files**

*expense.json sample file*

There are 20 related questions .

## 4.3.7.1 Introduction to the CONNECT Engine

CONNECT is not just a new "YASE" (Yet another Storage Engine) that provides another way to store data with additional features. It brings a new dimension to MariaDB, already one of the best products to deal with traditional database transactional applications, further into the world of business intelligence and data analysis, including NoSQL facilities. Indeed, BI is the set of techniques and tools for the transformation of raw data into meaningful

and useful information. And where is this data?

"It's amazing in an age where relational databases reign supreme when it comes to managing data that so much information still exists outside RDBMS engines in the form of flat files and other such constructs. In most enterprises, data is passed back and forth between disparate systems in a fashion and speed that would rival the busiest expressways in the world, with much of this data existing in common, delimited files. Target systems intercept these source files and then typically proceed to load them via ETL (extract, transform, load) processes into databases that then utilize the information for business intelligence, transactional functions, or other standard operations. ETL tasks and data movement jobs can consume quite a bit of time and resources, especially if large volumes of data are present that require loading into a database. This being the case, many DBAs welcome alternative means of accessing and managing data that exists in file format."

This has been written by Robin Schumacher. [1]

What he describes is known as MED (Management of External Data) enabling the handling of data not stored in a DBMS database as if it were stored in tables. An ISO standard exists that describes one way to implement and use MED in SQL by defining foreign tables for which an external FDW (Foreign Data Wrapper) has been developed in C.

However, since this was written, a new source of data was developed as the "cloud". Data are existing worldwide and, in particular, can be obtained in JSON or XML format in answer to REST queries. From [Connect 1.06.0010](#), it is possible to create JSON, XML or CSV tables based on data retrieved from such REST queries.

MED as described above is a rather complex way to achieve this goal and MariaDB does not support the ISO SQL/MED standard. But, to cover the need, possibly in transactional but mostly in decision support applications, the CONNECT storage engine supports MED in a much simpler way.

The main features of CONNECT are:

1. No need for additional SQL language extensions.
2. Embedded wrappers for many external data types (files, data sources, virtual).
3. NoSQL query facilities for [JSON](#), [XML](#), HTML files and using JSON UDFs.
4. NoSQL data obtained from REST queries (requires `cpprestsdk`).
5. NoSQL new data type [MONGO](#) accessing MongoDB collections as relational tables.
6. Read/Write access to external files of most commonly used formats.
7. Direct access to most external data sources via ODBC, JDBC and MySQL or MongoDB API.
8. Only used columns are retrieved from external scan.
9. Push-down WHERE clauses when appropriate.
10. Support of special and virtual columns.
11. Parallel execution of multi-table tables (currently unavailable).
12. Supports partitioning by sub-files or by sub-tables (enabling table sharding).
13. Support of MRR for SELECT, UPDATE and DELETE.
14. Provides remote, block, dynamic and virtual indexing.
15. Can execute complex queries on remote servers.
16. Provides an API that allows writing additional FDW in C++.

With CONNECT, MariaDB has one of the most advanced implementations of MED and NoSQL, without the need for complex additions to the SQL syntax (foreign tables are "normal" tables using the CONNECT engine).

Giving MariaDB easy and natural access to external data enables the use of all of its powerful functions and SQL-handling abilities for developing business intelligence applications.

With version 1.07 of CONNECT, retrieving data from REST queries is available in all binary distributed version of MariaDB, and, from 1.07.002, CONNECT allows workspaces greater than 4GB.

---

1. ↑ Robin Schumacher is Vice President Products at DataStax and former Director of Product Management at MySQL. He has over 13 years of database experience in DB2, MySQL, Oracle, SQL Server and other database engines.

## 4.3.7.2 Installing the CONNECT Storage Engine

The

CONNECT

storage engine enables MariaDB to access external local or remote data (MED). This is done by defining tables based on different data types, in particular files in various formats, data extracted from other DBMS or products (such as Excel or MongoDB) via ODBC or JDBC, or data retrieved from the environment (for example DIR, WMI, and MAC tables)

This storage engine supports table partitioning, MariaDB virtual columns and permits defining special columns such as ROWID, FILEID, and SERVID.

The storage engine must be installed before it can be used.

## Contents

- 1. [Installing the Plugin's Package](#)
  - 1. [Installing on Linux](#)
    - 1. [Installing with a Package Manager](#)
      - 1. [Installing with yum/dnf](#)
      - 2. [Installing with apt-get](#)
      - 3. [Installing with zypper](#)
  - 2. [Installing the Plugin](#)
  - 3. [Uninstalling the Plugin](#)
  - 4. [Installing Dependencies](#)
    - 1. [Installing unixODBC](#)
  - 5. [See Also](#)

## Installing the Plugin's Package

The

CONNECT  
storage engine's shared library is included in MariaDB packages as the  
`ha_connect.so`  
or  
`ha_connect.so`  
shared library on systems where it can be built.

## Installing on Linux

The

CONNECT  
storage engine is included in [binary tarballs](#) on Linux.

### Installing with a Package Manager

The

CONNECT  
storage engine can also be installed via a package manager on Linux. In order to do so, your system needs to be configured to install from one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the [MariaDB Package Repository setup script](#).

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the [MariaDB Repository Configuration Tool](#).

#### Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant [RPM package](#) from MariaDB's repository using

`yum`

or

`dnf`

. Starting with RHEL 8 and Fedora 22,

`yum`

has been replaced by

`dnf`

, which is the next major version of

`yum`

. However,

`yum`

commands still work on many systems that use

`dnf`

. For example:

```
sudo yum install MariaDB-connect-engine
```

#### Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant [DEB package](#) from MariaDB's repository using

```
apt-get
```

. For example:

```
sudo apt-get install mariadb-plugin-connect
```

Installing with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant [RPM package](#) from MariaDB's repository using

```
zypper
```

. For example:

```
sudo zypper install MariaDB-connect-engine
```

## Installing the Plugin

Once the shared library is in place, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'ha_connect';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = ha_connect
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'ha_connect';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Installing Dependencies

The

CONNECT

storage engine has some external dependencies.

## Installing unixODBC

The

CONNECT

storage engine requires an ODBC library. On Unix-like systems, that usually means installing [unixODBC](#). On some systems, this is installed as the

unixODBC

package. For example:

```
sudo yum install unixODBC
```

On other systems, this is installed as the

libodbc1

package. For example:

```
sudo apt-get install libodbc1
```

If you do not have the ODBC library installed, then you may get an error about a missing library when you attempt to install the plugin. For example:

```
INSTALL SONAME 'ha_connect';
ERROR 1126 (HY000): Can't open shared library '/home/ian/MariaDB_Downloads/10.1.17/lib/plugin/ha_connect.so'
(errno: 2, libodbc.so.1: cannot open shared object file: No such file or directory)
```

## See Also

- [PLUGIN OVERVIEW](#)

### 4.3.7.3 Creating and Dropping CONNECT Tables

#### Contents

1. [Table Options](#)
2. [Column Options](#)
3. [Index Options](#)

Create Table statements for “CONNECT” tables are standard MariaDB create statements specifying

engine=CONNECT

. There are a few additional table and column options specific to CONNECT.

## Table Options

| Table Option   | Type    | Description                                                                                                                                                                                                                                                            |
|----------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AVG_ROW_LENGTH | Integer | Can be specified to help CONNECT estimate the size of a variable record table length.                                                                                                                                                                                  |
| BLOCK_SIZE     | Integer | The number of rows each block of a <a href="#">FIX</a> , <a href="#">BIN</a> , <a href="#">DBF</a> , or <a href="#">VEC</a> table contains. For an <a href="#">ODBC</a> table this is the RowSet size option. For a <a href="#">JDBC</a> table this is the fetch size. |

|              |         |                                                                                                                                                                                                                              |
|--------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CATFUNC      | String  | The catalog function used by a <a href="#">catalog table</a> .                                                                                                                                                               |
| COLIST       | String  | The column list of <a href="#">OCCUR</a> tables or \$project of <a href="#">MONGO</a> tables.                                                                                                                                |
| COMPRESS     | Number  | <p>1<br/>or<br/>2</p> <p>if the data file is g-zip compressed. Defaults to 0. Before CONNECT 1.05.0001, this was boolean, and true if the data file is compressed.</p>                                                       |
| CONNECTION   | String  | Specifies the connection of an <a href="#">ODBC</a> , <a href="#">JDBC</a> or <a href="#">MYSQL</a> table.                                                                                                                   |
| DATA_CHARSET | String  | The character set used in the external file or data source.                                                                                                                                                                  |
| DBNAME       | String  | The target database for <a href="#">ODBC</a> , <a href="#">JDBC</a> , <a href="#">MYSQL</a> , <a href="#">catalog</a> , and <a href="#">PROXY</a> based tables. The database concept is sometimes known as a <i>schema</i> . |
| ENGINE       | String  | Must be specified as<br>CONNECT<br>.                                                                                                                                                                                         |
| ENDING       | Integer | End of line length. Defaults to<br>1<br>for Unix/Linux and<br>2<br>for Windows.                                                                                                                                              |
| FILE_NAME    | String  | The file (path) name for all table types based on files. Can be absolute or relative to the current data directory. If not specified, this is an <a href="#">Inward table</a> and a default value is used.                   |
| FILTER       | String  | To filter an external table. Currently MONGO tables only.                                                                                                                                                                    |
| HEADER       | Integer | Applies to <a href="#">CSV</a> , <a href="#">VEC</a> , and <a href="#">HTML</a> files. Its meaning depends on the table type.                                                                                                |
| HTTP         | String  | The HTTP of the client of REST queries. From <a href="#">Connect 1.06.0010</a> .                                                                                                                                             |
| HUGE         | Boolean | To specify that a table file can be larger than 2GB. For a <a href="#">MYSQL</a> table, prevents the result set from being stored in memory.                                                                                 |
| LRECL        | Integer | The file record size (often calculated by default).                                                                                                                                                                          |

|             |         |                                                                                                                                                                                                                                                                                                                              |
|-------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MAPPED      | Boolean | Specifies whether <i>file mapping</i> is used to handle the table file.                                                                                                                                                                                                                                                      |
| MODULE      | String  | The (path) name of the DLL or shared lib implementing the access of a non-standard ( <a href="#">OEM</a> ) table type.                                                                                                                                                                                                       |
| MULTIPLE    | Integer | Used to specify multiple file tables.                                                                                                                                                                                                                                                                                        |
| OPTION_LIST | String  | Used to specify all other options not yet directly defined.                                                                                                                                                                                                                                                                  |
| QCHAR       | String  | Specifies the character used for quoting some fields of a <a href="#">CSV</a> table or the identifiers of an <a href="#">ODBC / JDBC</a> tables.                                                                                                                                                                             |
| QUOTED      | Integer | The level of quoting used in <a href="#">CSV</a> table files.                                                                                                                                                                                                                                                                |
| READONLY    | Boolean | True if the data file must not be modified or erased.                                                                                                                                                                                                                                                                        |
| SEP_CHAR    | String  | Specifies the field separator character of a <a href="#">CSV</a> or <a href="#">XCOL</a> table. Also, used to specify the Jpath separator for <a href="#">JSON tables</a> .                                                                                                                                                  |
| SEPINDEX    | Boolean | When true, indexes are saved in separate files.                                                                                                                                                                                                                                                                              |
| SPLIT       | Boolean | True for a <a href="#">VEC</a> table when all columns are in separate files.                                                                                                                                                                                                                                                 |
| SRCDEF      | String  | The source definition of a table retrieved via <a href="#">ODBC</a> , <a href="#">JDBC</a> or the MySQL API or used by a <a href="#">PIVOT</a> table.                                                                                                                                                                        |
| SUBTYPE     | String  | The subtype of an <a href="#">OEM</a> table type.                                                                                                                                                                                                                                                                            |
| TABLE_LIST  | String  | The comma separated list of TBL table sub-tables.                                                                                                                                                                                                                                                                            |
| TABLE_TYPE  | String  | The external table <code>type : DOS , FIX , BIN , CSV , FMT , XML , JSON ,INI , DBF , VEC , ODBC , JDBC , MYSQL , TBL , PROXY , XCOL , OCCUR , PIVOT , ZIP , VIR , DIR , WMI , MAC , and OEM</code> . Defaults to <a href="#">DOS</a> , <a href="#">MYSQL</a> , or <a href="#">PROXY</a> depending on what options are used. |
| TABNAME     | String  | The target table or node for <a href="#">ODBC</a> , <a href="#">JDBC</a> , <a href="#">MYSQL</a> , <a href="#">PROXY</a> , or <a href="#">catalog tables</a> ; or the top node name for <a href="#">XML</a> tables.                                                                                                          |

|            |         |                                                                                                                                |
|------------|---------|--------------------------------------------------------------------------------------------------------------------------------|
| URI        | String  | The URI of a REST request.. From <a href="#">Connect 1.06.0010</a> .                                                           |
| XFILE_NAME | String  | The file (path) base name for table index files. Can be absolute or relative to the data directory. Defaults to the file name. |
| ZIPPED     | Boolean | True if the table file(s) is/are zipped in one or several zip files.                                                           |

All integers in the above table are unsigned big integers.

Because CONNECT handles many table types; many table type specific options are not in the above list and must be entered using the `OPTION_LIST` option. The syntax to use is:

```
... option_list='opname1=opvalue1,opname2=opvalue2...'
```

Be aware that until Connect 1.5.5, no blanks should be inserted before or after the '

```
=  
' and '  
,  
' characters. The option name is all that is between the start of the string or the last '  
,  
' character and the next '  
=  
' character, and the option value is all that is between this '  
=  
' character and the next '  
,  
' or end of string. For instance:
```

```
option_list='name=TABLE,coltype=HTML,attribute=border=1;cellpadding=5,headattr=bgcolor=yellow';
```

This defines four options,

```
name  
'  
coltype  
'  
attribute  
, and '  
headattr  
'; with values '  
TABLE  
'  
HTML  
'  
border=1;cellpadding=5  
, and '  
bgcolor=yellow  
' respectively. The only restriction is that values cannot contain commas, but they can contain equal signs.
```

## Column Options

| Column Option | Type   | Description                                             |
|---------------|--------|---------------------------------------------------------|
| DATE_FORMAT   | String | The format indicating how a date is stored in the file. |
| DISTRIB       | Enum   | "scattered", "clustered", "sorted" (ascending).         |

|                           |         |                                                            |
|---------------------------|---------|------------------------------------------------------------|
| <code>FIELD_FORMAT</code> | String  | The column format for some table types.                    |
| <code>FIELD_LENGTH</code> | Integer | Set the internal field length for DATE columns.            |
| <code>FLAG</code>         | Integer | An integer value whose meaning depends on the table type.  |
| <code>JPATH</code>        | String  | The Json path of JSON table columns.                       |
| <code>MAX_DIST</code>     | Integer | Maximum number of distinct values in this column.          |
| <code>SPECIAL</code>      | String  | The name of the SPECIAL column that set this column value. |
| <code>XPATH</code>        | String  | The XML path of XML table columns.                         |

- The

`MAX_DIST`  
and  
`DISTRIB`

column options are used for block indexing.

- All integers in the above table are unsigned big integers.
- `JPATH` and `XPATH` were added to make `CREATE TABLE` statements more readable, but they do the same thing as `FIELD_FORMAT` and any of them can be used with the same result.

## Index Options

| Index Option        | Type    | Description                 |
|---------------------|---------|-----------------------------|
| <code>DYNAM</code>  | Boolean | Set the index as “dynamic”. |
| <code>MAPPED</code> | Boolean | Use index file mapping.     |

**Note 1:** Creating a CONNECT table based on file does not erase or create the file if the file name is specified in the `CREATE TABLE` statement (“`outward`” table). If the file does not exist, it will be populated by subsequent `INSERT` or `LOAD` commands or by the “AS select statement” of the `CREATE TABLE` command. Unlike the CSV engine, CONNECT easily permits the creation of tables based on already existing files, for instance files made by other applications. However, if the file name is not specified, a file with a name defaulting to

`tablename.tablename`  
will be created in the data directory (“`inward`” table).

**Note 2:** Dropping a CONNECT table is done with a standard `DROP` statement. For `outward tables`, this drops only the CONNECT table definition but does not erase the corresponding data file and index files. Use

`DELETE`  
or  
`TRUNCATE`

to do so. This is contrary to data and index files of `inward tables` are erased on `DROP` like for other MariaDB engines.

## 4.3.7.4 CONNECT Data Types

## Contents

1. [TYPE\\_STRING](#)
2. [TYPE\\_INT](#)
3. [TYPE\\_SHORT](#)
4. [TYPE\\_TINY](#)
5. [TYPE\\_BIGINT](#)
6. [TYPE\\_DOUBLE](#)
7. [TYPE\\_DECIM](#)
8. [DATE Data type](#)
  1. [Date Format in Text Tables](#)
  2. [Usage Notes](#)
  3. [Handling dates that are out of the range of supported CONNECT dates](#)
9. [NULL handling](#)
10. [Unsigned numeric types](#)
11. [Data type conversion](#)

Many data types make no or little sense when applied to plain files. This why [CONNECT](#) supports only a restricted set of data types. However, ODBC, JDBC or MYSQL source tables may contain data types not supported by CONNECT. In this case, CONNECT makes an automatic conversion to a similar supported type when it is possible.

The data types currently supported by CONNECT are:

| Type name   | Description            | Used for                                                                                                                  |
|-------------|------------------------|---------------------------------------------------------------------------------------------------------------------------|
| TYPE_STRING | Zero ended string      | <a href="#">char</a> , <a href="#">varchar</a> , <a href="#">text</a>                                                     |
| TYPE_INT    | 4 bytes integer        | <a href="#">int</a> , <a href="#">mediumint</a> , <a href="#">integer</a>                                                 |
| TYPE_SHORT  | 2 bytes integer        | <a href="#">smallint</a>                                                                                                  |
| TYPE_TINY   | 1 byte integer         | <a href="#">tinyint</a>                                                                                                   |
| TYPE_BIGINT | 8 bytes integer        | <a href="#">bigint</a> , <a href="#">longlong</a>                                                                         |
| TYPE_DOUBLE | 8 bytes floating point | <a href="#">double</a> , <a href="#">float</a> , <a href="#">real</a>                                                     |
| TYPE_DECIM  | Numeric value          | <a href="#">decimal</a> , <a href="#">numeric</a> , <a href="#">number</a>                                                |
| TYPE_DATE   | 4 bytes integer        | <a href="#">date</a> , <a href="#">datetime</a> , <a href="#">time</a> , <a href="#">timestamp</a> , <a href="#">year</a> |

## TYPE\_STRING

This type corresponds to what is generally known as [CHAR](#) or [VARCHAR](#) by database users, or as strings by programmers. Columns containing characters have a maximum length but the character string is of fixed or variable length depending on the file format.

The [DATA\\_CHARSET](#) option must be used to specify the character set used in the data source or file. Note that, unlike usually with MariaDB, when a multi-byte character set is used, the column size represents the number of bytes the column value can contain, not the number of characters.

## TYPE\_INT

The ] [INTEGER](#) type contains signed integer numeric 4-byte values (the *int* of the C language) ranging from

-2,147,483,648  
to  
2,147,483,647  
for signed type and  
0  
to

4,294,967,295  
for unsigned type.

## TYPE\_SHORT

The SHORT data type contains signed [integer numeric 2-byte](#) values (the *short integer* of the C language) ranging from

-32,768  
to  
32,767  
for signed type and  
0  
to  
65,535  
for unsigned type.

## TYPE\_TINY

The TINY data type contains [integer numeric 1-byte](#) values (the *char* of the C language) ranging from

-128  
to  
127  
for signed type and  
0  
to  
255  
for unsigned type. For some table types, TYPE\_TINY is used to represent Boolean values (0 is false, anything else is true).

## TYPE\_BIGINT

The [BIGINT](#) data type contains signed integer 8-byte values (the *long long* of the C language) ranging from

-9,223,372,036,854,775,808  
to  
9,223,372,036,854,775,807  
for signed type and from  
0  
to  
18,446,744,073,709,551,615  
for unsigned type.

Inside tables, the coding of all integer values depends on the table type. In tables represented by text files, the number is written in characters, while in tables represented by binary files (

BIN  
or  
VEC

) the number is directly stored in the binary representation corresponding to the platform.

The *length* (or *precision*) specification corresponds to the length of the table field in which the value is stored for text files only. It is used to set the output field length for all table types.

## TYPE\_DOUBLE

The DOUBLE data type corresponds to the C language [double](#) type, a floating-point double precision value coded with 8 bytes. Like for integers, the internal coding in tables depends on the table type, characters for text files, and platform binary representation for binary files.

The *length* specification corresponds to the length of the table field in which the value is stored for text files only. The *scale* (was *precision*) is the number of decimal digits written into text files. For binary table types (BIN and VEC) this does not apply. The *length* and *scale* specifications are used to set the output field length and number of decimals for all types of tables.

## TYPE\_DECIM

The DECIMAL data type corresponds to what MariaDB or ODBC data sources call NUMBER, NUMERIC, or [DECIMAL](#) : a numeric value with a maximum number of digits (the *precision*) some of them eventually being decimal digits (the *scale*). The internal coding in CONNECT is a character representation of the number. For instance:

```
colname decimal(14,6)
```

This defines a column *colname* as a number having a *precision* of 14 and a *scale* of 6. Supposing it is populated by:

```
insert into xxx values (-2658.74);
```

The internal representation of it will be the character string

```
-2658.740000
```

. The way it is stored in a file table depends on the table type. The *length* field specification corresponds to the length of the table field in which the value is stored and is calculated by CONNECT from the *precision* and the *scale* values. This length is *precision* plus 1 if *scale* is not 0 (for the decimal point) plus 1 if this column is not unsigned (for the eventual minus sign). In fix formatted tables the number is right justified in the field of width *length*, for variable formatted tables, such as CSV, the field is the representing character string.

Because this type is mainly used by CONNECT to handle numeric or decimal fields of ODBC, JDBC and MySQL table types, CONNECT does not provide decimal calculations or comparison by itself. This is why decimal columns of CONNECT tables cannot be indexed.

## DATE Data type

Internally, date/time values are stored by CONNECT as a signed 4-byte integer. The value 0 corresponds to 01 January 1970 12:00:00 am coordinated universal time ([UTC](#)). All other date/time values are represented by the number of seconds elapsed since or before midnight (00:00:00), 1 January 1970, to that date/time value. Date/time values before midnight 1 January 1970 are represented by a negative number of seconds.

CONNECT handles dates from **13 December 1901, 20:45:52 to 18 January 2038, 19:14:07**.

Although date and time information can be represented in both CHAR and INTEGER data types, the DATE data type has special associated properties. For each DATE value, CONNECT can store all or only some of the following information: century, year, month, day, hour, minute, and second.

## Date Format in Text Tables

Internally, date/time values are handled as a signed 4-byte integer. But in text tables (type DOS, FIX, CSV, FMT, and DBF) dates are most of the time stored as a formatted character string (although they also can be stored as a numeric string representing their internal value). Because there are infinite ways to format a date, the format to use for decoding dates, as well as the field length in the file, must be associated to date columns (except when they are stored as the internal numeric value).

Note that this associated format is used only to describe the way the temporal value is stored internally. This format is used both for output to decode the date in a SELECT statement as well as for input to encode the date in INSERT or UPDATE statements. However, what is kept in this value depends on the data type used in the column definition (all the MariaDB temporal values can be specified). When creating a table, the format is associated to a date column using the DATE\_FORMAT option in the column definition, for instance:

```
create table birthday (
  Name varchar(17),
  Bday date field_length=10 date_format='MM/DD/YYYY',
  Btime time field_length=8 date_format='hh:mm tt')
engine=CONNECT table_type=CSV;

insert into birthday values ('Charlie','2012-11-12','15:30:00');

select * from birthday;
```

The SELECT query returns:

| Name    | Bday       | Btime    |
|---------|------------|----------|
| Charlie | 2012-11-12 | 15:30:00 |

The values of the INSERT statement must be specified using the standard MariaDB syntax and these values are displayed as MariaDB temporal values. Sure enough, the column formats apply only to the way these values are represented inside the CSV files. Here, the inserted record will be:

```
Charlie,11/12/2012,03:30 PM
```

**Note:** The field\_length option exists because the MariaDB syntax does not allow specifying the field length between parentheses for temporal column types. If not specified, the field length is calculated from the date format (sometimes as a max value) or made equal to the default length value if there is no date format. In the above example it could have been removed as the calculated values are the ones specified. However, if the table type would have been DOS or FIX, these values could be adjusted to fit the actual field length within the file.

A CONNECT format string consists of a series of elements that represent a particular piece of information and define its format. The elements will be recognized in the order they appear in the format string. Date and time format elements will be replaced by the actual date and time as they appear in the source string. They are defined by the following groups of characters:

| Element | Description                                                                     |
|---------|---------------------------------------------------------------------------------|
| YY      | The last two digits of the year (that is, 1996 would be coded as "96").         |
| YYYY    | The full year (that is, 1996 could be entered as "96" but displayed as "1996"). |
| MM      | The one or two-digit month number.                                              |
| MMM     | The three-character month abbreviation.                                         |

|      |                                                                     |
|------|---------------------------------------------------------------------|
| MMMM | The full month name.                                                |
| DD   | The one or two-digit month day.                                     |
| DDD  | The three-character weekday abbreviation.                           |
| DDDD | The full weekday name.                                              |
| hh   | The one or two-digit hour in 12-hour or 24-hour format.             |
| mm   | The one or two-digit minute.                                        |
| ss   | The one or two-digit second.                                        |
| t    | The one-letter AM/PM abbreviation (that is, AM is entered as "A").  |
| tt   | The two-letter AM/PM abbreviation (that is, AM is entered as "AM"). |

## Usage Notes

- To match the source string, you can add body text to the format string, enclosing it in single quotes or double quotes if it would be ambiguous. Punctuation marks do not need to be quoted.
- The hour information is regarded as 12-hour format if a “t” or “tt” element follows the “hh” element in the format or as 24-hour format otherwise.
- The “MM”, “DD”, “hh”, “mm”, “ss” elements can be specified with one or two letters (e.g. “MM” or “M”) making no difference on input, but placing a leading zero to one-digit values on output [1] for two-letter elements.
- If the format contains elements DDD or DDDD, the day of week name is skipped on input and ignored to calculate the internal date value. On output, the correct day of week name is generated and displayed.
- Temporal values are always stored as numeric in [BIN](#) and [VEC](#) tables.

## Handling dates that are out of the range of supported CONNECT dates

If you want to make a table containing, for instance, historical dates not being convertible into CONNECT dates, make your column CHAR or VARCHAR and store the dates in the MariaDB format. All date functions applied to these strings will convert them to MariaDB dates and will work as if they were real dates. Of course they must be inserted and will be displayed using the MariaDB format.

## NULL handling

CONNECT handles [null values](#) for data sources able to produce nulls. Currently this concerns mainly the [ODBC](#), [JDBC](#), [MONGO](#), [MYSQL](#), [XML](#), [JSON](#) and [INI](#) table types. For INI, [JSON](#), MONGO or XML types, null values are returned when the key is missing in the section (INI) or when the corresponding node does not exist in a row (XML, JSON, MONGO).

For other file tables, the issue is to define what a null value is. In a numeric column, 0 can sometimes be a valid value but, in some other cases, it can make no sense. The same for character columns; is a blank field a valid value or not?

A special case is DATE columns with a DATE\_FORMAT specified. Any value not matching the format can be regarded as NULL.

CONNECT leaves the decision to you. When declaring a column in the [CREATE TABLE](#) statement, if it is declared NOT NULL, blank or zero values will be considered as valid values. Otherwise they will be considered as NULL values. In all cases, nulls are replaced on insert or update by pseudo null values, a zero-length character string for text types or a zero value for numeric types. Once converted to pseudo null values, they will be recognized as NULL only for columns declared as nullable.

For instance:

```
create table t1 (a int, b char(10)) engine=connect;
insert into t1 values (0,'zero'),(1,'one'),(2,'two'),(null,'???');
select * from t1 where a is null;
```

The select query replies:

| a    | b    |
|------|------|
| NULL | zero |
| NULL | ???  |

Sure enough, the value 0 entered on the first row is regarded as NULL for a nullable column. However, if we execute the query:

```
select * from t1 where a = 0;
```

This will return no line because a NULL is not equal to 0 in an SQL where clause.

Now let us see what happens with not null columns:

```
create table t1 (a int not null, b char(10) not null) engine=connect;
insert into t1 values (0,'zero'),(1,'one'),(2,'two'),(null,'???');
```

The insert statement will produce a warning saying:

| Level   | Code | Message                   |
|---------|------|---------------------------|
| Warning | 1048 | Column 'a' cannot be null |

It is replaced by a pseudo null

0

on the fourth row. Let us see the result:

```
select * from t1 where a is null;
select * from t1 where a = 0;
```

The first query returns no rows, 0 are valid values and not NULL. The second query replies:

| a | b    |
|---|------|
| 0 | zero |
| 0 | ???  |

It shows that the NULL inserted value was replaced by a valid 0 value.

## Unsigned numeric types

They are supported by CONNECT since version 1.01.0010 for fixed numeric types (TINY, SHORT, INTEGER, and BITINT).

## Data type conversion

CONNECT is able to convert data from one type to another in most cases. These conversions are done without warning even when this leads to truncation or loss of precision. This is true, in particular, for tables of type ODBC, JDBC, MYSQL and PROXY (via MySQL) because the source table may contain some data types not supported by CONNECT. They are converted when possible to CONNECT types.

When converted, MariaDB types are converted as:

| MariaDB Types            | CONNECT Type              | Remark                                                                              |
|--------------------------|---------------------------|-------------------------------------------------------------------------------------|
| integer , medium integer | TYPE_INT                  | 4 byte integer                                                                      |
| small integer            | TYPE_SHORT                | 2 byte integer                                                                      |
| tiny integer             | TYPE_TINY                 | 1 byte integer                                                                      |
| char , varchar           | TYPE_STRING               | Same length                                                                         |
| double , float , real    | TYPE_DOUBLE               | 8 byte floating point                                                               |
| decimal , numeric        | TYPE_DECIM                | Length depends on precision and scale                                               |
| all date related types   | TYPE_DATE                 | Date format can be set accordingly                                                  |
| bigint , longlong        | TYPE_BIGINT               | 8 byte integer                                                                      |
| enum , set               | TYPE_STRING               | Numeric value not accessible                                                        |
| All text types           | TYPE_STRING<br>TYPE_ERROR | Depending on the value of the <code>connect_type_conv</code> system variable value. |
| Other types              | TYPE_ERROR                | Not supported, no conversion provided.                                              |

For `ENUM`, the length of the column is the length of the longest value of the enumeration. For `SET` the length is enough to contain all the set values concatenated with comma separator.

In the case of `TEXT` columns, the handling depends on the values given to the `connect_type_conv` and `connect_conv_size` system variables.

Note: `BLOB` is currently not converted by default until a `TYPE_BIN` type is added to CONNECT. However, the `FORCE` option (from Connect 1.06.006) can be specified for blob columns containing text and the `SKIP` option also applies to ODBC BLOB columns.

ODBC SQL types are converted as:

| SQL Types             | Connect Type | Remark                                                                                                                                                   |
|-----------------------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQL_CHAR, SQL_VARCHAR | TYPE_STRING  |                                                                                                                                                          |
| SQL_LONGVARCHAR       | TYPE_STRING  | len = min(abs(len), connect_conv_size)<br>If the column is generated by discovery (columns not specified) its length is <code>connect_conv_size</code> . |

|                                              |             |                                                                                                                                                          |
|----------------------------------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQL_NUMERIC, SQL_DECIMAL                     | TYPE_DECIM  |                                                                                                                                                          |
| SQL_INTEGER                                  | TYPE_INT    |                                                                                                                                                          |
| SQL_SMALLINT                                 | TYPE_SHORT  |                                                                                                                                                          |
| SQL_TINYINT, SQL_BIT                         | TYPE_TINY   |                                                                                                                                                          |
| SQL_FLOAT, SQL_REAL, SQL_DOUBLE              | TYPE_DOUBLE |                                                                                                                                                          |
| SQL_DATETIME                                 | TYPE_DATE   | len = 10                                                                                                                                                 |
| SQL_INTERVAL                                 | TYPE_STRING | len = 8 + ((scale) ? (scale+1) : 0)                                                                                                                      |
| SQL_TIMESTAMP                                | TYPE_DATE   | len = 19 + ((scale) ? (scale +1) : 0)                                                                                                                    |
| SQL_BIGINT                                   | TYPE_BIGINT |                                                                                                                                                          |
| SQL_GUID                                     | TYPE_STRING | <br>len=36                                                                                                                                               |
| SQL_BINARY, SQL_VARBINARY, SQL_LONGVARBINARY | TYPE_STRING | len = min(abs(len), connect_conv_size)<br>Only if the value of <a href="#">connect_type_conv</a> is force<br>. The column should use the binary charset. |
| Other types                                  | TYPE_ERROR  | <i>Not supported.</i>                                                                                                                                    |

JDBC SQL types are converted as:

| JDBC Types                    | Connect Type              | Remark                                                                                                                                                     |
|-------------------------------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| (N)CHAR, (N)VARCHAR           | TYPE_STRING               |                                                                                                                                                            |
| LONG(N)VARCHAR                | TYPE_STRING               | len = min(abs(len), connect_conv_size)<br>If the column is generated by discovery (columns not specified), its length is <a href="#">connect_conv_size</a> |
| NUMERIC, DECIMAL, VARBINARY   | TYPE_DECIM                |                                                                                                                                                            |
| INTEGER                       | TYPE_INT                  |                                                                                                                                                            |
| SMALLINT                      | TYPE_SHORT                |                                                                                                                                                            |
| TINYINT, BIT                  | TYPE_TINY                 |                                                                                                                                                            |
| FLOAT, REAL, DOUBLE           | TYPE_DOUBLE               |                                                                                                                                                            |
| DATE                          | TYPE_DATE                 | len = 10                                                                                                                                                   |
| TIME                          | TYPE_DATE                 | len = 8 + ((scale) ? (scale+1) : 0)                                                                                                                        |
| TIMESTAMP                     | TYPE_DATE                 | len = 19 + ((scale) ? (scale +1) : 0)                                                                                                                      |
| BIGINT                        | TYPE_BIGINT               |                                                                                                                                                            |
| UUID (specific to PostgreSQL) | TYPE_STRING<br>TYPE_ERROR | len=36<br>If <a href="#">connect_type_conv=NO</a>                                                                                                          |
| Other types                   | TYPE_ERROR                | <i>Not supported.</i>                                                                                                                                      |

Note: The [connect\\_type\\_conv](#) SKIP option also applies to ODBC and JDBC tables.

1. ↑ Here input and output are used to specify respectively decoding the date to get its numeric value from the data file and encoding a date to write it in the table file. Input is performed within [SELECT](#) queries; output is performed in [UPDATE](#) or [INSERT](#) queries.

## 4.3.7.5 Current Status of the CONNECT Handler

The current CONNECT handler is a GA (stable) release. It was written starting both from an aborted project written for MySQL in 2004 and from the “DBCONNECT” program. It was tested on all the examples described in this document, and is distributed with a set of 53 test cases. Here is a not limited list of future developments:

1. Adding more table types.
2. Make more tests files (53 are already made)
3. Adding more data types, in particular unsigned ones (done for unsigned).
4. Supporting indexing on nullable and decimal columns.
5. Adding more optimize tools (block indexing, dynamic indexing, etc.) (done)
6. Supporting MRR (done)
7. Supporting partitioning (done)
8. Getting NOSQL data from the Net as answers from REST queries (done)

No programs are bug free, especially new ones. Please [report all bugs](#) or documentation errors using the means provided by MariaDB.

## 4.3.7.6 CONNECT Table Types

The main feature of [CONNECT](#) is to give MariaDB the ability to handle tables from many sources, native files, other DBMS's tables, or special “virtual” tables. Moreover, for all tables physically represented by data files, CONNECT recognizes many different file formats, described below but not limited in the future to this list, because more can be easily added to it on demand ([OEM tables](#) ).

Note: You can download a [PDF version of the CONNECT documentation \(1.7.0003\)](#).



### CONNECT Table Types Overview

*CONNECT can handle many table formats.*



### Inward and Outward Tables

*The two broad categories of CONNECT tables.*



### CONNECT Table Types - Data Files

*CONNECT plain DOS or UNIX data files.*



### CONNECT Zipped File Tables

*When the table file or files are compressed in one or several zip files.*



### CONNECT DOS and FIX Table Types

*CONNECT tables based on text files*



### CONNECT DBF Table Type

*CONNECT dBASE III or IV tables.*



### CONNECT BIN Table Type

*CONNECT binary files in which each row is a logical record of fixed length*



### CONNECT VEC Table Type

*CONNECT binary files organized in vectors*



### CONNECT CSV and FMT Table Types

*Variable length CONNECT data files.*



### CONNECT - NoSQL Table Types

*Based on files that do not match the relational format but often represent hierarchical data.*



### CONNECT - Files Retrieved Using Rest Queries

*JSON, XML and CSV data files can be retrieved as results from REST queries.*



### CONNECT JSON Table Type

*JSON (JavaScript Object Notation) is a widely-used lightweight data-interchange format.*



### CONNECT XML Table Type

*CONNECT XML files*



## CONNECTINI Table Type

*CONNECTINI Windows configuration or initialization files.*



## CONNECT - External Table Types

**1** *Access tables belonging to the current or another server.*



## CONNECT ODBC Table Type: Accessing Tables From Another DBMS

**3** *CONNECT Table Types - ODBC Table Type: Accessing Tables from other DBMS*



## CONNECT JDBC Table Type: Accessing Tables from Another DBMS

*Using JDBC to access other tables.*



## CONNECT MONGO Table Type: Accessing Collections from MongoDB

**1** *Used to directly access MongoDB collections as tables.*



## CONNECT MYSQL Table Type: Accessing MySQL/MariaDB Tables

**5** *Accessing a MySQL or MariaDB table or view*



## CONNECT PROXY Table Type

*Tables that access and read the data of another table or view*



## CONNECT XCOL Table Type

*Based on another table/view, used when object tables have a column that contains a list of values*



## CONNECT OCCUR Table Type

*Extension to the PROXY type when referring to a table/view having several c...*



## CONNECT PIVOT Table Type

**2** *Transform the result of another table into another table along "pivot" and "fact" columns.*



## CONNECT TBL Table Type: Table List

**2** *Define a table as a list of tables of any engine and type.*



## CONNECT - Using the TBL and MYSQL Table Types Together

*Used together, the TBL and MYSQL types lift all the limitations of the FEDERATED and MERGE engines*



## CONNECT Table Types - Special "Virtual" Tables

*VIR, WMI and MAC special table types*



## CONNECT Table Types - VIR

*VIR virtual type for CONNECT*



## CONNECT Table Types - OEM: Implemented in an External LIB

*CONNECT OEM table types are implemented in an external library.*



## CONNECT Table Types - Catalog Tables

*Catalog tables return information about another table or data source*

There are 4 related questions .

## 4.3.7.6.1 CONNECT Table Types Overview

CONNECT can handle very many table formats; it is indeed one of its main features. The Type option specifies the type and format of the table. The Type options available values and their descriptions are listed in the following table:

| Type    | Description                                                                                                                                                                                                                                                                                                                          |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BIN     | Binary file with numeric values in platform representation, also with columns at fixed offset within records and fixed record length.                                                                                                                                                                                                |
| BSON    | (Temporary) JSON table handled by the new JSON handling.                                                                                                                                                                                                                                                                             |
| CSV *\$ | "Comma Separated Values" file in which each variable length record contains column values separated by a specific character (defaulting to the comma)                                                                                                                                                                                |
| DBF *   | File having the dBASE format.                                                                                                                                                                                                                                                                                                        |
| DOS     | The table is contained in one or several files. The file format can be refined by some other options of the command or more often using a specific type as many of those described below. Otherwise, it is a flat text file where columns are placed at a fixed offset within each record, the last column being of variable length. |

|                 |                                                                                                                                                                                                                                 |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>DIR</b>      | Virtual table that returns a file list like the Unix<br>ls<br>or DOS<br>dir<br>command.                                                                                                                                         |
| <b>FIX</b>      | Text file arranged like DOS but with fixed length records.                                                                                                                                                                      |
| <b>FMT</b>      | File in which each record contains the column values in a non-standard format (the same for each record) This format is specified in the column definition.                                                                     |
| <b>INI</b>      | File having the format of the initialization or configuration files used by many applications.                                                                                                                                  |
| <b>JDBC *</b>   | Table accessed via a JDBC driver.                                                                                                                                                                                               |
| <b>JSON *\$</b> | File having the JSON format.                                                                                                                                                                                                    |
| <b>MAC</b>      | Virtual table returning information about the machine and network cards (Windows only).                                                                                                                                         |
| <b>MONGO *</b>  | Table accessed via the MongoDB C Driver API.                                                                                                                                                                                    |
| <b>MYSQL *</b>  | Table accessed using the MySQL API like the FEDERATED engine.                                                                                                                                                                   |
| <b>OCCUR *</b>  | A table based on another table existing on the current server, several columns of the object table containing values that can be grouped in only one column.                                                                    |
| <b>ODBC *</b>   | Table extracted from an application accessible via ODBC or unixODBC. For example from another DBMS or from an Excel spreadsheet.                                                                                                |
| <b>OEM *</b>    | Table of any other formats not directly handled by CONNECT but whose access is implemented by an external FDW (foreign data wrapper) written in C++ (as a DLL or Shared Library).                                               |
| <b>PIVOT *</b>  | Used to "pivot" the display of an existing table or view.                                                                                                                                                                       |
| <b>PROXY *</b>  | A table based on another table existing on the current server.                                                                                                                                                                  |
| <b>TBL *</b>    | Accessing a collection of tables as one table (like the MERGE engine does for MyISAM tables)                                                                                                                                    |
| <b>VEC</b>      | Binary file organized in vectors, in which column values are grouped consecutively, either split in separate files or in a unique file.                                                                                         |
| <b>VIR*</b>     | Virtual table containing only special and virtual columns.                                                                                                                                                                      |
| <b>WMI *</b>    | Windows Management Instrumentation table displaying information coming from a WMI provider. This type enables to get in tabular format all sorts of information about the machine hardware and operating system (Windows only). |
| <b>XCOL *</b>   | A table based on another table existing on the current server with one of its columns containing comma separated values.                                                                                                        |
| <b>XML *\$</b>  | File having the XML or HTML format.                                                                                                                                                                                             |
| <b>ZIP</b>      | Table giving information about the contents of a zip file.                                                                                                                                                                      |

## Catalog Tables

For all table types marked with a '\*' in the table above, CONNECT is able to analyze the data source to retrieve the column definition. This can be used to define a "catalog" table that displays the column description of the source, or to create a table without specifying the column definition that will be automatically constructed by CONNECT when creating the table.

When marked with a '\$' the file can be the result returned by a REST query.

### 4.3.7.6.2 Inward and Outward Tables

#### Contents

- 1. [Outward Tables](#)
  - 1. [Altering Outward Tables](#)
- 2. [Inward Tables](#)
  - 1. [Altering Inward Tables](#)

There are two broad categories of file-based CONNECT tables. Inward and Outward. They are described below.

## Outward Tables

Tables are "outward" when their file name is specified in the CREATE TABLE statement using the `file_name` option.

Firstly, remember that CONNECT implements MED (Management of External Data). This means that the "true" CONNECT tables – "outward tables" – are based on data that belongs to files that can be produced by other applications or data imported from another DBMS.

Therefore, their data is "precious" and should not be modified except by specific commands such as `INSERT`, `UPDATE`, or `DELETE`. For other

commands such as [CREATE](#), [DROP](#), or [ALTER](#) their data is never modified or erased.

Outward tables can be created on existing files or external tables. When they are dropped, only the local description is dropped, the file or external table is not dropped or erased. Also, [DROP TABLE](#) does not erase the indexes.

[ALTER TABLE](#) produces the following warning, as a reminder:

```
Warning (Code 1105): This is an outward table, table data were not modified.
```

If the specified file does not exist, it is created when data is inserted into the table. If a [SELECT](#) is issued before the file is created, the following error is produced:

```
Warning (Code 1105): Open(rb) error 2 on <file_path>: No such file or directory
```

## Altering Outward Tables

When an [ALTER TABLE](#) is issued, it just modifies the table definition accordingly without changing the data. [ALTER](#) can be used safely to, for instance, modify options such as [MAPPED](#), [HUGE](#) or [READONLY](#) but with extreme care when modifying column definitions or order options because some column options such as [FLAG](#) should also be modified or may become wrong.

Changing the table type with [ALTER](#) often makes no sense. But many suspicious alterations can be acceptable if they are just meant to correct an existing wrong definition.

Translating a CONNECT table to another engine is fine but the opposite is forbidden when the target CONNECT table is not table based or when its data file exists (because when the target table data cannot be changed and if the source table is dropped, the table data would be lost). However, it can be done to create a new file-based tables when its file does not exist or is void.

Creating or dropping indexes is accepted because it does not modify the table data. However, it is often unsafe to do it with an [ALTER TABLE](#) statement that does other modifications.

Of course, all changes are acceptable for empty tables.

**Note:** Using outward tables requires the [FILE](#) privilege.

## Inward Tables

A special type of file-based CONNECT tables are "inward" tables. They are file-based tables whose file name is not specified in the [CREATE TABLE](#) statement (no `file_name` option).

Their file will be located in the current database directory and their name will default to `tablename.type` where `tablename` is the table name and `type` is the table type folded to lower case. When they are created without using a

```
CREATE TABLE ... SELECT ...
statement, an empty file is made at create time and they can be populated by further inserts.
```

They behave like tables of other storage engines and, unlike outward CONNECT tables, they are erased when the table is dropped. Of course they should not be read-only to be usable. Even though their utility is limited, they can be used for testing purposes or when the user does not have the [FILE](#) privilege.

## Altering Inward Tables

One thing to know, because CONNECT builds indexes in a specific way, is that all index modifications are done using an "in-place" algorithm – meaning not using a temporary table. This is why, when indexing is specified in an [ALTER TABLE](#) statement containing other changes that cannot be done "in-place", the statement cannot be executed and raises an error.

Converting an inward table to an outward table, using an [ALTER TABLE](#) statement specifying a new file name and/or a new table type, is restricted the same way it is when converting a table from another engine to an outward table. However there are no restrictions to convert another engine table to a CONNECT inward table.

## 4.3.7.6.3 CONNECT Table Types - Data Files

### Contents

1. [Multiple File Tables](#)
2. [Record Format](#)
3. [File Mapping](#)
4. [Big File Tables](#)
5. [Compressed File Tables](#)
6. [Relational Formatted Tables](#)
7. [NoSQL Table Types](#)

Most of the tables processed by CONNECT are just plain DOS or UNIX data files, logically regarded as tables thanks to the description given when creating the table. This description comes from the

[CREATE TABLE](#)

statement. Depending on the application, these tables can already exist as data files, used as is by CONNECT, or can have been physically

made by CONNECT as the result of a

```
CREATE TABLE ... SELECT ...
and/or INSERT statement(s).
```

The file *path/name* is given by the

```
FILE_NAME
option. If it is a relative path/name, it will be relative to the database directory, the one containing the table
.FR
file.
```

Unless specified, the maturity of file table types is stable.

## Multiple File Tables

A **multiple** file table is one that is physically contained in several files of the same type instead of just one. These files are processed sequentially during the process of a query and the result is the same as if all the table files were merged into one. This is great to process files coming from different sources (such as cash register log files) or made at different time periods (such as bank monthly reports) regarded as one table. Note that the operations on such files are restricted to sequential Select and Update; and that VEC multiple tables are not supported by CONNECT. The file list depends on the setting of the **multiple** option of the

```
CREATE TABLE
statement for that table.
```

Multiple tables are specified by the option MULTIPLE= *n* , which can take four values:

|   |                                                                                                                                                                    |
|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | Not a multiple table (the default). This can be used in an <a href="#">ALTER TABLE</a> statement.                                                                  |
| 1 | The table is made from files located in the same directory. The FILE_NAME option is a pattern such as<br>'cash*.log'<br>that all the table file path/names verify. |
| 2 | The FILE_NAME gives the name of a file that contains the path/names of all the table files. This file can be made using a DIR table.                               |
| 3 | Like multiple=1 but also including eligible files from the directory sub-folders.                                                                                  |

The

```
FILEID
special column, described here , allows query pruning by filtering the file list or doing some grouping on the files that make a multiple table.
```

**Note:** Multiple was not initially implemented for XML tables. This restriction was removed in version 1.02.

## Record Format

This characteristic applies to table files handled by the operating system input/output functions. It is **fixed** for table types **FIX** , **BIN** , **DBF** and **VEC** , and it is variable for **DOS** , **VCT**, **FMT** and some **JSON** tables.

For fixed tables, most I/O operations are done by block of BLOCK\_SIZE rows. This diminishes the number of I/O's and enables block indexing.

Starting with CONNECT version 1.6.6, the BLOCK\_SIZE option can also be specified for variable tables. Then, a file similar to the block indexing file is created by CONNECT that gives the size in bytes of each block of BLOCK\_SIZE rows. This enables the use of block I/Os and block indexing to variable tables. It also enables CONNECT to return the exact row number for info commands

## File Mapping

For file-based tables of reasonable size, processing time can be greatly enhanced under Windows(TM) and some flavors of UNIX or Linux by using the technique of "file mapping", in which a file is processed as if it were entirely in memory. Mapping is specified when creating the table by the use of the

```
MAPPED=YES
option. This does not apply to tables not handled by system I/O functions (
```

```
XML
```

and

```
INI
```

).

## Big File Tables

Because all files are handled by the standard input/output functions of the operating system, their size is limited to 2GB, the maximum size handled by standard functions. For some table types, CONNECT can deal with files that are larger than 2GB, or prone to become larger than this limit. These are the **FIX** , **BIN** and **VEC** types. To tell connect to use input/output functions dealing with big files, specify the option

```
huge=1
or
```

huge=YES

for that table. Note however that CONNECT cannot randomly access tables having more than 2G records.

## Compressed File Tables

CONNECT can make and process some tables whose data file is compressed. The only supported compression format is the zlib format. Zip and zlib formats are supported differently. The table types that can be compressed are [DOS](#), [FIX](#), [BIN](#), [CSV](#) and [FMT](#). This can save some disk space at the cost of a somewhat longer processing time.

Some restrictions apply to compressed tables:

- Compressed tables are not indexable.
- Update and partial delete are not supported.

Use the numeric compress option to specify a compressed table:

1. Not compressed
2. Compressed in zlib format.
3. Made of compressed blocks of block\_size records (enabling block indexing)

## Relational Formatted Tables

These are based on files whose records represent one table row. Only the column representation within each record can differ. The following relational formatted tables are supported:

- [DOS and FIX Table Types](#)
- [DBF Table Type](#)
- [BIN Table Type](#)
- [VEC Table Type](#)
- [CSV and FMT Table Types](#)

## NoSQL Table Types

These are based on files that do not match the relational format but often represent hierarchical data. CONNECT can handle JSON, INI-CFG, XML and some HTML files..

The way it is done is different from what PostgreSQL does. In addition to including in a table some column values of a specific data format (JSON, XML) to be handled by specific functions, CONNECT can directly use JSON, XML or INI files that can be produced by other applications and this is the table definition that describes where and how the contained information must be retrieved.

This is also different from what MariaDB does with [dynamic columns](#), which is close to what MySQL and PostgreSQL do with the JSON column type.

The following NoSQL types are supported:

- [XML Table Type](#)
- [JSON Table Type](#)
- [INI Table Type](#)

## 4.3.7.6.4 CONNECT Zipped File Tables

MariaDB starting with [10.2.4](#)

Connect can work on table files that are compressed in one or several zip files.

The specific options used when creating tables based on zip files are:

| Table Option | Type    | Description                                                                                                                                                                                                                                               |
|--------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ZIPPED       | Boolean | Required to be set as true.                                                                                                                                                                                                                               |
| ENTRY*       | String  | The optional name or pattern of the zip entry or entries to be used with the table. If not specified, all entries or only the first one will be used depending on the mulentries option setting.                                                          |
| MULENTRIES*  | Boolean | True if several entries are part of the table. If not specified, it defaults to false if the entry option is not specified. If the entry option is specified, it defaults to true if the entry name contains wildcard characters or false if it does not. |
| APPEND*      | Boolean | Used when creating new zipped tables (see below)                                                                                                                                                                                                          |
| LOAD*        | String  | Used when creating new zipped tables (see below)                                                                                                                                                                                                          |

Options marked with a '\*' must be specified in the option list.

Examples of use:

Example 1: Single CSV File Included in a Single ZIP File

Let's suppose you have a CSV file from which you would create a table by:

```
create table emp
... optional column definition
engine=connect table_type=CSV file_name='E:/Data/employee.csv'
sep_char=';' header=1;
```

If the CSV file is included in a ZIP file, the CREATE TABLE becomes:

```
create table empzip
... optional column definition
engine=connect table_type=CSV file_name='E:/Data/employee.zip'
sep_char=';' header=1 zipped=1 option_list='Entry=emp.csv';
```

The `file_name` option is the name of the zip file. The `entry` option is the name of the entry inside the zip file. If there is only one entry file inside the zip file, this option can be omitted.

## Example 2: Several CSV Files Included in a Single ZIP File

If the table is made from several files such as `emp01.csv`, `emp02.csv`, etc., the standard create table would be:

```
create table empmul (
... required column definition
) engine=connect table_type=CSV file_name='E:/Data/emp*.csv'
sep_char=';' header=1 multiple=1;
```

But if these files are all zipped inside a unique zip file, it becomes:

```
create table empzmul
... required column definition
engine=connect table_type=CSV file_name='E:/Data/emp.zip'
sep_char=';' header=1 zipped=1 option_list='Entry=emp*.csv';
```

Here the `entry` option is the pattern that the files inside the zip file must match. If all entry files are ok, the `entry` option can be omitted but the Boolean option `mulentries` must be specified as true.

## Example 3: Single CSV File included in Multiple ZIP Files (Without considering subfolders)

If the table is created on several zip files, it is specified as for all other multiple tables:

```
create table zempmul (
... required column definition
) engine=connect table_type=CSV file_name='E:/Data/emp*.zip'
sep_char=';' header=1 multiple=1 zipped=yes
option_list='Entry=employee.csv';
```

Here again the `entry` option is used to restrict the entry file(s) to be used inside the zip files and can be omitted if all are ok.

The column descriptions can be retrieved by the discovery process for table types allowing it. It cannot be done for multiple tables or multiple entries.

A catalog table can be created by adding `catfunc=columns`. This can be used to show the column definitions of multiple tables. `Multiple` must be set to false and the column definitions will be the ones of the first table or entry.

This first implementation has some restrictions:

1. Zipped tables are read-only. `UPDATE` and `DELETE` are not supported. However, `INSERT` is supported in a specific way when making tables.
2. The inside files are decompressed into memory. Memory problems may arise with huge files.
3. Only file types that can be handled from memory are eligible for this. This includes `DOS`, `FIX`, `BIN`, `CSV`, `FMT`, `DBF`, `JSON`, and `XML` table types, as well as types based on these such as `XCOL`, `OCCUR` and `PIVOT`.

Optimization by indexing or block indexing is possible for table types supporting it. However, it applies to the uncompressed table. This means that the whole table is always uncompressed.

Partitioning is also supported. See how to do it in the section about partitioning.

## Creating New Zipped Tables

Tables can be created to access already existing zip files. However, is it also possible to make the zip file from an existing file or table. Two ways are available to make the zip file:

### Insert Method

`insert` can be used to make the table file for table types based on records (this excludes DBF, XML and JSON when `pretty` is not 0). However, the current

implementation of the used package (minizip) does not support adding to an already existing zip entry. This means that when executing an insert statement the inserted records are not added but replace the existing ones. CONNECT protects existing data by not allowing such inserts, Therefore, only three ways are available to do so:

1. Using only one insert statement to make the whole table. This is possible only for small tables and is principally useful when making tests.
2. Making the table from the data of another table. This can be done by executing an "insert into table select \* from another\_table" or by specifying "as select \* from another\_table" in the create table statement.
3. Making the table from a file whose format enables to use the "load data infile" statement.

To add a new entry in an existing zip file, specify "append=YES" in the option list. When inserting several entries, use ALTER to specify the required options, for instance:

```
create table znumul (
Chiffre int(3) not null,
Lettre char(16) not null
engine=CONNECT table_type=CSV
file_name='C:/Data/FMT/mnum.zip' header=1 lrecl=20 zipped=1
option_list='Entry=Num1';
insert into znumul select * from num1;
alter table znumul option_list='Entry=Num2,Append=YES';
insert into znumul select * from num2;
alter table znumul option_list='Entry=Num3,Append=YES';
insert into znumul select * from num3;
alter table znumul option_list='Entry=Num*,Append=YES';
select * from znumul;
```

The last ALTER is needed to display all the entries.

## File Zipping Method

This method enables to make the zip file from another file when creating the table. It applies to all table types including DBF, XML and JSON. It is specified in the create table statement with the load option. For example:

```
create table XSERVZIP (
NUMERO varchar(4) not null,
LIEU varchar(15) not null,
CHEF varchar(5) not null,
FONCTION varchar(12) not null,
NOM varchar(21) not null
engine=CONNECT table_type=XML file_name='E:/Xml/perso.zip' zipped=1
option_list='entry=services,load=E:/Xml/serv2.xml';
```

When executing this statement, the *serv2.xml* file will be zipped as */perso.zip*. The entry name can be specified or defaults to the source file name.

If the column descriptions are specified, the table can be used later to read from the zipped table, but they are not used when creating the zip file. Thus, a fake column (there must be one) can be specified and another table created to read the zip file. This one can take advantage of the discovery process to avoid providing the columns description for table types allowing it. For instance:

```
create table mkzq (whatever int)
engine=connect table_type=DBF zipped=1
file_name='C:/Data/EAUX/dbf/CQUART.ZIP'
option_list='Load=C:/Data/EAUX/dbf/CQUART.DBF';

create table zquart
engine=connect table_type=DBF zipped=1
file_name='C:/Data/EAUX/dbf/CQUART.ZIP';
```

It is also possible to create a multi-entries table from several files:

```
CREATE TABLE znewcities (
_id char(5) NOT NULL,
city char(16) NOT NULL,
lat double(18,6) NOT NULL `FIELD_FORMAT`='loc:[0]',
lng double(18,6) NOT NULL `FIELD_FORMAT`='loc:[1]',
pop int(6) NOT NULL,
state char(2) NOT NULL
) ENGINE=CONNECT TABLE_TYPE=JSON FILE_NAME='E:/Json/newcities.zip' ZIPPED=1 LRECL=1000 OPTION_LIST='Load=E:/Json/city_*.json,mul',
```

Here the files to load are specified with wildcard characters and the *mulentries* options must be specified. However, the *entry* option must not be specified, entry names will be made from the file names. Provide a fake column description if the files have different column layout, but specific tables will have to be created to read each of them.

## ZIP Table Type

A ZIP table type is also available. It is not meant to read the inside files but to display information about the zip file contents. For instance:

```
create table xzipinfo2 (
entry varchar(256)not null,
cmpsize bigint not null flag=1,
uncsize bigint not null flag=2,
method int not null flag=3,
date datetime not null flag=4)
engine=connect table_type=ZIP file_name='E:/Data/Json/cities.zip';
```

This will display the name, compressed size, uncompressed size, and compress method of all entries inside the zip file. Column names are irrelevant; these are flag values that mean what information to retrieve.

It is possible to retrieve this information from several zip files by specifying the multiple option:

```
create table TestZip1 (
entry varchar(260)not null,
cmpsize bigint not null flag=1,
uncsize bigint not null flag=2,
method int not null flag=3,
date datetime not null flag=4,
zipname varchar(256) special='FILEID')
engine=connect table_type=ZIP multiple=1
file_name='C:/Data/Ziptest/CCAM06300_DBF_PART*.zip';
```

Here we added the special column zipname to get the name of the zip file for each entry.

## 4.3.7.6.5 CONNECT DOS and FIX Table Types

### Contents

1. [Overview](#)
2. [Specifying the Field Format](#)
3. [Example](#)

### Overview

Tables of type DOS and FIX are based on text files (see [CONNECT Table Types - Data Files](#) ). Within a record, column fields are positioned at a fixed offset from the beginning of the record. Except sometimes for the last field, column fields are also of fixed length. If the last field has varying length, the type of the table is DOS. For instance, having the file *dept.dat* formatted like:

|                   |                   |                        |
|-------------------|-------------------|------------------------|
| 0318 KINGSTON     | 70012 SALES       | Bank/Insurance         |
| 0021 ARMONK       | 87777 CHQ         | Corporate headquarter  |
| 0319 HARRISON     | 40567 SALES       | Federal Administration |
| 2452 POUGHKEEPSIE | 31416 DEVELOPMENT | Research & development |

You can define a table based on it with:

```
create table department (
number char(4) not null,
location char(15) not null flag=5,
director char(5) not null flag=20,
function char(12) not null flag=26,
name char(22) not null flag=38)
engine=CONNECT table_type=DOS file_name='dept.dat';
```

Here the flag column option represents the offset of this column inside the records. If the offset of a column is not specified, it defaults to the end of the previous column and defaults to 0 for the first one. The

*lrecl*

parameter that represents the maximum size of a record is calculated by default as the end of the rightmost column and can be unspecified except when some trailing information exists after the rightmost column.

**Note:** A special case is files having an encoding such as UTF-8 (for instance specifying

*charset=UTF8*

) in which some characters may be represented with several bytes. Unlike the type size that MariaDB interprets as a number of characters, the

*lrecl*

*value* is the record size in bytes and the flag value represents the offset of the field in the record in bytes. If the flag and/or the

*lrecl*

*value* are not specified, they will be calculated by the number of characters in the fields multiplied by a value that is the maximum size in bytes of a character for the corresponding charset. For UTF-8 this value is 3 which is often far too much as there are very few characters requiring 3 bytes to be represented. When creating a new file, you are on the safe side by only doubling the maximum number of characters of a field to

calculate the offset of the next field. Of course, for already existing files, the offset must be specified according to what it is in it.

Although the field representation is always text in the table file, you can freely choose the corresponding column type, characters, date, integer or floating point according to its contents.

Sometimes, as in the *number* column of the above *department* table, you have the choice of the type, numeric or characters. This will modify how the column is internally handled — in characters

```
0021  
is different from  
21  
but not in numeric — as well as how it is displayed.
```

If the last field has fixed length, the table should be referred as having the type

```
FIX  
. For instance, to create a table on the file boys.txt :
```

|        |          |            |            |
|--------|----------|------------|------------|
| John   | Boston   | 25/01/1986 | 02/06/2010 |
| Henry  | Boston   | 07/06/1987 | 01/04/2008 |
| George | San Jose | 10/08/1981 | 02/06/2010 |
| Sam    | Chicago  | 22/11/1979 | 10/10/2007 |
| James  | Dallas   | 13/05/1992 | 14/12/2009 |
| Bill   | Boston   | 11/09/1986 | 10/02/2008 |

You can for instance use the command:

```
create table boys (  
    name char(12) not null,  
    city char(12) not null,  
    birth date not null date_format='DD/MM/YYYY',  
    hired date not null date_format='DD/MM/YYYY' flag=36  
engine=CONNECT table_type=FIX file_name='boys.txt' lrecl=48;
```

Here some *flag* options were not specified because the fields have no intermediate space between them except for the last column. The offsets are calculated by default adding the field length to the *offset* of the preceding field. However, for formatted date columns, the offset in the file depends on the format and cannot be calculated by default. For fixed files, the *lrec* option is the physical length of the record including the line ending character(s). It is calculated by adding to the end of the last field 2 bytes under Windows (CRLF) or 1 byte under UNIX. If the file is imported from another operating system, the

```
ENDING  
option will have to be specified with the proper value.
```

For this table, the last offset and the record length must be specified anyway because the date columns have field length coming from their format that is not known by CONNECT. Do not forget to add the line ending length to the total length of the fields.

This table is displayed as:

| name   | city     | birth      | hired      |
|--------|----------|------------|------------|
| John   | Boston   | 1986-01-25 | 2010-06-02 |
| Henry  | Boston   | 1987-06-07 | 2008-04-01 |
| George | San Jose | 1981-08-10 | 2010-06-02 |
| Sam    | Chicago  | 1979-11-22 | 2007-10-10 |
| James  | Dallas   | 1992-05-13 | 2009-12-14 |
| Bill   | Boston   | 1986-09-11 | 2008-02-10 |

Whenever possible, the fixed format should be preferred to the varying one because it is much faster to deal with fixed tables than with variable tables. Sure enough, instead of being read or written record by record, FIX tables are processed by blocks of

```
BLOCK_SIZE  
records, resulting in far less input/output operations to execute. The block size defaults to 100 if not specified in the Create Table statement.
```

**Note 1:** It is not mandatory to declare in the table all the fields existing in the source file. However, if some fields are ignored, the *flag* option of the following field and/or the *lrec* option will have to be specified.

**Note 2:** Some files have an EOF marker (CTRL+Z 1A) that can prevent the table to be recognized as fixed because the file length is not a multiple of the fixed record size. To indicate this, use in the option list the create option EOF. For instance, if after creating the FIX table *xtab* on the file *foo.dat* that you know have fixed record size, you get, when you try to use it, a message such as:

```
File foo.dat is not fixed length, len=302587 lrecl=141
```

After checking that the LRECL default or specified specification is correct, you can indicate to ignore that extra EOF character by:

```
alter table xtab option_list='eof=1';
```

Of course, you can specify this option directly in the Create statement. All this applies to some other table types, in particular to BIN tables.

**Note 3:** The width of the fields is the length specified in the column declaration. For instance for a column declared as:

```
number int(3) not null,
```

The field width in the file is 3 characters. This is the value used to calculate the offset of the next field if it is not specified. If this length is not specified, it defaults to the MySQL default type length.

## Specifying the Field Format

Some files have specific format for their numeric fields. For instance, the decimal point is absent and/or the field should be filled with leading zeros. To deal with such files, as well in reading as in writing, the format can be specified in the

```
CREATE TABLE
```

column definition. The syntax of the field format specification is:

```
Field_format='[Z][N][d]'
```

The optional parts of the format are:

- Z** The field has leading zeros
- N** No decimal point exist in the file
- d** The number of decimals, defaults to the column precision

## Example

Let us see how it works in the following example. We define a table based on the file xfmt.txt having eight fields of 12 characters:

```
create table xfmt (
    col1 double(12,3) not null,
    col2 double(12,3) not null field_format='4',
    col3 double(12,2) not null field_format='N3',
    col4 double(12,3) not null field_format='Z',
    col5 double(12,3) not null field_format='Z3',
    col6 double(12,5) not null field_format='ZN5',
    col7 int(12) not null field_format='N3',
    col8 smallint(12) not null field_format='N3')
engine=CONNECT table_type=FIX file_name='xfmt.txt';

insert into xfmt values(4567.056,4567.056,4567.056,4567.056,-23456.8,
    3.14159,4567,4567);
select * from xfmt;
```

The first row is displayed as:

| COL1     | COL2     | COL3    | COL4     | COL5       | COL6    | COL7 | COL8 |
|----------|----------|---------|----------|------------|---------|------|------|
| 4567.056 | 4567.056 | 4567.06 | 4567.056 | -23456.800 | 3.14159 | 4567 | 4567 |

The number of decimals displayed for all float columns is the column precision, the second argument of the column type option. Of course, integer columns have no decimals, although their formats specify some.

More interesting is the file layout. To see it let us define another table based on the same file but whose columns are all characters:

```
create table cfmt (
    col1 char(12) not null,
    col2 char(12) not null,
    col3 char(12) not null,
    col4 char(12) not null,
    col5 char(12) not null,
    col6 char(12) not null,
    col7 char(12) not null,
    col8 char(12) not null)
engine=CONNECT table_type=FIX file_name='xfmt.txt';
select * from cfmt;
```

The (transposed) display of the select command shows the file text layout for each field. Below a third column was added in this document to comment this result.

| Column | Row 1 | Comment (all numeric fields are written right justified) |
|--------|-------|----------------------------------------------------------|
|        |       |                                                          |

|             |              |                                                                                                                                 |
|-------------|--------------|---------------------------------------------------------------------------------------------------------------------------------|
| <b>COL1</b> | 4567.056     | No format, the value was entered as is.                                                                                         |
| <b>COL2</b> | 4567.0560    | The format '4' forces to write 4 decimals.                                                                                      |
| <b>COL3</b> | 4567060      | N3 → No decimal point. The last 3 digits are decimals. However, the second decimal was rounded because of the column precision. |
| <b>COL4</b> | 00004567.056 | Z → Leading zeros, 3 decimals (the column precision)                                                                            |
| <b>COL5</b> | -0023456.800 | Z3 → (Minus sign) leading zeros, 3 decimals.                                                                                    |
| <b>COL6</b> | 000000314159 | ZN5 → Leading zeros, no decimal point, 5 decimals.                                                                              |
| <b>COL7</b> | 4567000      | N3 → No decimal point. The last 3 digits are decimals.                                                                          |
| <b>COL8</b> | 4567000      | Same. Any decimals would be ignored.                                                                                            |

**Note:** For columns internally using double precision floating-point numbers, MariaDB limits the decimal precision of any calculation to the column precision. The declared column precision should be at least the number of decimals of the format to avoid a loss of decimals as it happened for

col3

of the above example.

## 4.3.7.6.6 CONNECT DBF Table Type

### Contents

- 1. [Overview](#)
- 2. [Conversion of dBASE Data Types](#)
- 3. [Reading soft deleted lines of a DBF table](#)

## Overview

A table of type

DBF

is physically a dBASE III or IV formatted file (used by many products like dBASE, Xbase, FoxPro etc.). This format is similar to the [FIX](#) type format with in addition a prefix giving the characteristics of the file, describing in particular all the fields (columns) of the table.

Because

DBF

files have a header that contains Meta data about the file, in particular the column description, it is possible to create a table based on an existing

DBF

file without giving the column description, for instance:

```
create table cust engine=CONNECT table_type=DBF file_name='cust.dbf';
```

To see what CONNECT has done, you can use the

DESCRIBE

or

SHOW CREATE TABLE

commands, and eventually modify some options with the

ALTER TABLE

command.

The case of deleted lines is handled in a specific way for DBF tables. Deleted lines are not removed from the file but are "soft deleted" meaning they are marked as deleted. In particular, the number of lines contained in the file header does not take care of soft deleted lines. This is why if you execute these

two commands applied to a DBF table named *tabdbf*:

```
select count(*) from tabdbf;
select count(*) from tabdbf where 1;
```

They can give a different result, the (fast) first one giving the number of physical lines in the file and the second one giving the number of line that are not (soft) deleted.

The commands UPDATE, INSERT, and DELETE can be used with DBF tables. The DELETE command marks the deleted lines as suppressed but keeps them in the file. The INSERT command, if it is used to populate a newly created table, constructs the file header before inserting new lines.

**Note:** For DBF tables, column name length is limited to 11 characters and field length to 256 bytes.

## Conversion of dBASE Data Types

CONNECT handles only types that are stored as characters.

| Symbol | DBF Type        | CONNECT Type                        | Description                                                                                         |
|--------|-----------------|-------------------------------------|-----------------------------------------------------------------------------------------------------|
| B      | Binary (string) | TYPE_STRING                         | 10 digits representing a .DBT block number.                                                         |
| C      | Character       | TYPE_STRING                         | All OEM code page characters - padded with blanks to the width of the field.                        |
| D      | Date            | TYPE_DATE                           | 8 bytes - date stored as a string in the format YYYYMMDD.                                           |
| N      | Numeric         | TYPE_INT TYPE_BIGINT<br>TYPE_DOUBLE | Number stored as a string, right justified, and padded with blanks to the width of the field.       |
| L      | Logical         | TYPE_STRING                         | 1 byte - initialized to 0x20 otherwise T or F.                                                      |
| M      | Memo (string)   | TYPE_STRING                         | 10 digits representing a .DBT block number.                                                         |
| @      | Timestamp       | Not supported                       | 8 bytes - two longs, first for date, second for time. It is the number of days since 01/01/4713 BC. |
| I      | Long            | Not supported                       | 4 bytes. Leftmost bit used to indicate sign, 0 negative.                                            |
| +      | Autoincrement   | Not supported                       | Same as a Long                                                                                      |
| F      | Float           | TYPE_DOUBLE                         | Number stored as a string, right justified, and padded with blanks to the width of the field.       |
| O      | Double          | Not supported                       | 8 bytes - no conversions, stored as a double.                                                       |
| G      | OLE             | TYPE_STRING                         | 10 digits representing a .DBT block number.                                                         |

For the N numeric type, CONNECT converts it to TYPE\_DOUBLE if the decimals value is not 0, to TYPE\_BIGINT if the length value is greater than 10, else to TYPE\_INT.

For M, B, and G types, CONNECT just returns the DBT number.

## Reading soft deleted lines of a DBF table

It is possible to read these lines by changing the read mode of the table. This is specified by an option

READMODE

that can take the values:

- 0 Standard mode. This is the default option.
- 1 Read all lines including soft deleted ones.
- 2 Read only the soft deleted lines.

For example, to read all lines of the *tabdbf* table, you can do:

```
alter table tabdbf option_list='Readmode=1';
```

To come back to normal mode, specify READMODE=0.

## 4.3.7.6.7 CONNECT BIN Table Type

### Contents

- 1. Overview
- 2. Type Conversion in BIN Tables
- 3. Example
- 4. Numeric fields alignment

# Overview

A table of type BIN is physically a binary file in which each row is a logical record of fixed length [1]. Within a record, column fields are of a fixed offset and length as with [FIX tables](#). Specific to BIN tables is that numerical values are internally encoded using native platform representation, so no conversion is needed to handle numerical values in expressions.

It is not required that the lines of a BIN file be separated by characters such as CR and/or LF but this is possible. In such an event, the */rec/* option must be specified accordingly.

**Note:** Unlike for the [DOS and FIX types](#), the width of the fields is the length of their internal representation in the file. For instance for a column declared as:

```
number int(5) not null,
```

The field width in the file is 4 characters, the size of a binary integer. This is the value used to calculate the offset of the next field if it is not specified. Therefore, if the next field is placed 5 characters after this one, this declaration is not enough, and the flag option will have to be used on the next field.

## Type Conversion in BIN Tables

Here are the correspondences between the column type and field format provided by default:

| Column type          | File default format             |
|----------------------|---------------------------------|
| Char( <i>n</i> )     | Text of <i>n</i> characters.    |
| Date                 | Integer (4 bytes)               |
| Int( <i>n</i> )      | Integer (4 bytes)               |
| Smallint( <i>n</i> ) | Short integer (2 bytes)         |
| TinyInt( <i>n</i> )  | Char (1 Byte)                   |
| Bigint( <i>n</i> )   | Large integer (8 bytes)         |
| Double( <i>n,d</i> ) | Double floating point (8 bytes) |

However, the column type need not necessarily match the field format within the table file. In particular, this occurs for field formats that correspond to numeric types that are not handled by CONNECT [2]. Indeed, BIN table files may internally contain float numbers or binary numbers of any byte length in big-endian or little-endian representation [3]. Also, as in [DOS or FIX types](#) tables, you may want to handle some character fields as numeric or vice versa.

This is why it is possible to specify the field format when it does not correspond to the column type default using the *field\_format* column option in the [CREATE TABLE](#) statement. Here are the available field formats for BIN tables:

| Field_format                          | Internal representation                                                                  |
|---------------------------------------|------------------------------------------------------------------------------------------|
| [ <i>n</i> ]{L or B or H}{ <i>n</i> } | <i>n</i> bytes binary number in little endian, big endian or host endian representation. |
| C                                     | Characters string ( <i>n</i> bytes)                                                      |
| I                                     | integer (4 bytes)                                                                        |
| D                                     | Double float (8 bytes)                                                                   |
| S                                     | Short integer (2 bytes)                                                                  |
| T                                     | Tiny integer (1 byte)                                                                    |
| G                                     | Big integer (8 bytes)                                                                    |
| F or R                                | Real or float (Floating point number on 4 bytes)                                         |
| X                                     | Use the default format field for the column type                                         |

All field formats (except the first one) are a one-character specification [4]. 'X' is equivalent to not specifying the field format. For the 'C' character specification, *n* is the column width as specified with the column type. For one-column formats, the number of bytes of the numeric fields corresponds to what it is on most platforms. However, it could vary for some. The G, I, S and T formats are deprecated because they correspond to supported data types and may not be supported in future versions.

## Example

Here is an example of a BIN table. The file record layout is supposed to be:

```
NNNNCCCCCCCCIIIISSFFFFSS
```

Here N represents numeric characters, C any characters, I integer bytes, S short integer bytes, and F float number bytes. The

<sup>1111</sup>

field contains a date in numeric format.

The table could be created by:

```
create table testbal (
fig int(4) not null field_format='C',
name char(10) not null,
birth date not null field_format='L',
id char(5) not null field_format='L2',
salary double(9,2) not null default 0.00 field_format='F',
dept int(4) not null field_format='L2')
engine=CONNECT table_type=BIN block_size=5 file_name='Testbal.dat';
```

Specifying the little-endian representation for binary values is not useful on most machines, but makes the create table statement portable on a machine using big endian, as well as the table file.

The field offsets and the file record length are calculated according the column internal format and eventually modified by the field format. It is not necessary to specify them for a packed binary file without line endings. If a line ending is desired, specify the ending option or specify the

```
lrec1
option adding the ending width. The table can be filled by:
```

```
insert into testbal values
(5500,'ARCHIBALD','1980-01-25','3789',4380.50,318),
(123,'OLIVER','1953-08-10',23456,3400.68,2158),
(3123,'FOO','2002-07-23','888',default,318);
```

Note that the types of the inserted values must match the column type, not the field format type.

The query:

```
select * from testbal;
```

returns:

| fig  | name      | birth      | id    | salary  | dept |
|------|-----------|------------|-------|---------|------|
| 5500 | ARCHIBALD | 1980-01-25 | 3789  | 4380.50 | 318  |
| 123  | OLIVER    | 1953-08-10 | 23456 | 3400.68 | 2158 |
| 3123 | FOO       | 2002-07-23 | 888   | 0.00    | 318  |

## Numeric fields alignment

In binary files, numeric fields and record length can be aligned on 4-or-8-byte boundaries to optimize performance on certain processors. This can be modified in the OPTION\_LIST with an "align" option ("packed" meaning

```
align=1
is the default).
```

1. ↑ Sometimes it can be a physical record if LF or CRLF have been written in the file.
2. ↑ Most of these are obsolete because CONNECT supports all column types except float
3. ↑ The default endian representation used in the table file can be specified by setting the ENDIAN option as 'L' or 'B' in the option list.
4. ↑ It can be specified with more than one character, but only the first one is significant.

## 4.3.7.6.8 CONNECT VEC Table Type

### Contents

1. [Integral vector formats](#)
2. [Differences between vector formats](#)
3. [Header option](#)

**Warning:** Avoid using this table type in production applications. This file format is specific to CONNECT and may not be supported in future versions.

Tables of type

VEC

are binary files that in some cases can provide good performance on read-intensive query workloads. CONNECT organizes their data on disk as columns of values from the same attribute, as opposed to storing it as rows of tabular records. This organization means that when a query needs to access only a few columns of a particular table, only those columns need to be read from disk. Conversely, in a row-oriented table, all values in a table are typically read from disk, wasting I/O bandwidth.

CONNECT provides two integral VEC formats, in which each column's data is adjacent.

## Integral vector formats

In these true vertical formats, the VEC files are made of all the data of the first column, followed by all the data of the second column etc. All this can be in one physical file or each column data can be in a separate file. In the first case, the option `max_rows=m`, where m is the estimate of the maximum size (number of rows) of the table, must be specified to be able to insert some new records. This leaves an empty space after each column area in which new data can be inserted. In the second case, the "Split" option can be specified [2] at table creation and each column will be stored in a file named sequentially from the table file name followed by the rank of the column. Inserting new lines can freely augment such a table.

## Differences between vector formats

These formats correspond to different needs. The integral vector format provides the best performance gain. It will be chosen when the speed of decisional queries must be optimized.

In the case of a unique file, inserting new data will be limited but there will be only one open and close to do. However, the size of the table cannot be calculated from the file size because of the eventual unused space in the file. It must be kept in a header containing the maximum number of rows and the current number of valid rows in the table. To achieve this, specify the option `Header= n` when creating the table. If

`n=1`  
the header will be placed at the beginning of the file, if  
`n=2`  
it will be a separate file with the type '.blk', and if  
`n=3`

the header will be place at the end of the file. This last value is provided because batch inserting is sometimes slower when the header is at the beginning of the file. If not specified, the header option will default to 2 for this table type.

On the other hand, the "Split" format with separate files have none of these issues, and is a much safer solution when the table must frequently inserted or shared among several users.

For instance:

```
create table vtab (
  a int not null,
  b char(10) not null
  engine=CONNECT table_type=VEC file_name='vt.vec';
```

This table, split by default, will have the column values in files vt1.vec and vt2.vec.

For vector tables, the option `block_size=n` is used for block reading and writing; however, to have a file made of blocks of equal size, the internal value of the `max_rows=m` option is eventually increased to become a multiple of n.

Like for BIN tables, numeric values are stored using platform internal layout, the correspondence between column types and internal format being the same than the default ones given above for BIN. However, field formats are not available for VEC tables.

## Header option

This applies to VEC tables that are not split. Because the file size depends on the MAX\_ROWS value, CONNECT cannot know how many valid records exist in the file. Depending on the value of the HEADER option, this information is stored in a header that can be placed at the beginning of the file, at the end of the file or in a separate file called fn.blk. The valid values for the HEADER option are:

- 0 Defaults to 2 for standard tables and to 3 for inward tables.
- 1 The header is at the beginning of the file.
- 2 The header is in a separate file.
- 3 The header is at the end of the file.

The value 2 can be used when dealing with files created by another application with no header. The value 3 makes sometimes inserting in the file faster than when the header is at the beginning of the file.

Note: VEC being a file format specific to CONNECT, no big endian / little endian conversion is provided. These files are not portable between machines using a different byte order setting.

## 4.3.7.6.9 CONNECT CSV and FMT Table Types

## Contents

- 1. CSV Type
  - 1. Restrictions on CSV Tables
- 2. FMT Type
- 3. Column Format Specification of FMT tables
- 4. Optional Fields
- 5. Bad Record Error Processing
- 6. Fields Containing a Formatted Date

## CSV Type

Many source data files are formatted with variable length fields and records. The simplest format, known as

CSV

(Comma Separated Variables), has column fields separated by a separator character. By default, the separator is a comma but can be specified by the

SEP\_CHAR

option as any character, for instance a semi-colon.

If the CSV file first record is the list of column names, specifying the

HEADER=1

option will skip the first record on reading. On writing, if the file is empty, the column names record is automatically written.

For instance, given the following *people.csv* file:

```
Name;birth;children
"Archibald";17/05/01;3
"Nabucho";12/08/03;2
```

You can create the corresponding table by:

```
create table people (
    name char(12) not null,
    birth date not null date_format='DD/MM/YY',
    children smallint(2) not null
engine=CONNECT table_type=CSV file_name='people.csv'
header=1 sep_char=';' quoted=1;
```

For CSV tables, the *flag* column option is the rank of the column into the file starting from 1 for the leftmost column. This is to enable having column displayed in a different order than in the file and/or to define the table specifying only some columns of the CSV file. For instance:

```
create table people (
    name char(12) not null,
    children smallint(2) not null flag=3,
    birth date not null flag=2 date_format='DD/MM/YY')
engine=CONNECT table_type=CSV file_name='people.csv'
header=1 sep_char=';' quoted=1;
```

In this case the command:

```
select * from people;
```

will display the table as:

| name      | children | birth      |
|-----------|----------|------------|
| Archibald | 3        | 2001-05-17 |
| Nabucho   | 2        | 2003-08-12 |

Many applications produce CSV files having some fields quoted, in particular because the field text contains the separator character. For such files, specify the 'QUOTED= *n*' option to indicate the level of quoting and/or the '

QCHAR=c

' to specify what is this eventual quoting character, which is

"

by default. Quoting with single quotes must be specified as

QCHAR='''

. On writing, fields will be quoted depending on the value of the quoting level, which is

-1

by default meaning no quoting:

0 The fields between quotes are read and the quotes discarded. On writing, fields will be quoted only if they contain the separator character or begin with the quoting character. If they contain the quoting character, it will be doubled.

1 Only text fields will be written between quotes, except null fields. This includes also the column names of an eventual header.

2 All fields will be written between quotes, except null fields.

3 All fields will be written between quotes, including null fields.

Files written this way are successfully read by most applications including spreadsheets.

**Note 1:** If only the QCHAR option is specified, the QUOTED option will default to 1.

**Note 2:** For CSV tables whose separator is the tab character, specify

```
sep_char='\t'
```

**Note 3:** When creating a table on an existing CSV file, you can let CONNECT analyze the file and make the column description. However, this is not an elaborate analysis of the file and, for instance,

DATE

fields will not be recognized as such but will be regarded as string fields.

## Restrictions on CSV Tables

- If

```
secure_file_priv
```

is set to the path of some directory, then CSV tables can only be created with files in that directory.

## FMT Type

FMT tables handle files of various formats that are an extension of the concept of CSV files. CONNECT supports these files providing all lines have the same format and that all fields present in all records are recognizable (optional fields must have recognizable delimiters). These files are made by specific application and CONNECT handles them in read only mode.

FMT tables must be created as CSV tables, specifying their type as FMT. In addition, each column description must be added to its format specification.

## Column Format Specification of FMT tables

The input format for each column is specified as a FIELD\_FORMAT option. A simple example is:

```
IP Char(15) not null field_format=' %n%s%n',
```

In the above example, the format for this (1st) field is

' %n%s%n'

. Note that the blank character at the beginning of this format is significant. No trailing blank should be specified in the column formats.

The syntax and meaning of the column input format is the one of the C **scanf** function.

However, CONNECT uses the input format in a specific way. Instead of using it to directly store the input value in the column buffer; it uses it to delimit the sub string of the input record that contains the corresponding column value. Retrieving this value is done later by the column functions as for standard CSV files.

This is why all column formats are made of five components:

1. An eventual description of what is met and ignored before the column value.
2. A marker of the beginning of the column value written as

%n

(or

%m

for optional fields).

5. An eventual description of what is met after the column value (not valid is

%m

was used).

For example, taking the file *funny.txt*:

```
12345, 'BERTRAND', #200;5009.13
56, 'POIROT-DELMOTTE', #4256 ;18009
345 , 'TRUCMUCHE' , #67; 19000.25
```

You can make a table *fmtsamp*e with 4 columns ID, NAME, DEPNO and SALARY, using the Create Table statement and column formats:

```
create table FMTSAMPLE (
  ID Integer(5) not null field_format=' %n%d%n',
  NAME Char(16) not null field_format=' ''%n[%^']%n''',
  DEPNO Integer(4) not null field_format=' ', #%n%d%n',
  SALARY Double(12,2) not null field_format=' ; %n%f%n')
Engine=CONNECT table_type=FMT file_name='funny.txt';
```

**Field 1** is an integer (

%d  
with eventual leading blanks.

**Field 2** is separated from field 1 by optional blanks, a comma, and other optional blanks and is between single quotes. The leading quote is included in component 1 of the column format, followed by the

%n  
marker. The column value is specified as  
%[^']  
meaning to keep any characters read until a quote is met. The ending marker (

%n  
) is followed by the 5th component of the column format, the single quote that follows the column value.

**Field 3**, also separated by a comma, is a number preceded by a pound sign.

**Field 4**, separated by a semicolon eventually surrounded by blanks, is a number with an optional decimal point (

%f  
).

This table will be displayed as:

| ID    | NAME            | DEPNO | SALARY   |
|-------|-----------------|-------|----------|
| 12345 | BERTRAND        | 200   | 5009.13  |
| 56    | POIROT-DELMOTTE | 4256  | 18009.00 |
| 345   | TRUCMUCHE       | 67    | 19000.25 |

## Optional Fields

To be recognized, a field normally must be at least one character long. For instance, a numeric field must have at least one digit, or a character field cannot be void. However many existing files do not follow this format.

Let us suppose for instance that the preceding example file could be:

```
12345, 'BERTRAND', #200;5009.13
56, 'POIROT-DELMOTTE', # ;18009
345 , ' ', #67; 19000.25
```

This will display an error message such as "*Bad format line x field y of FMTSAMPLE*". To avoid this and accept these records, the corresponding fields must be specified as "optional". In the above example, fields 2 and 3 can have null values (in lines 3 and 2 respectively). To specify them as optional, their format must be terminated by

%m  
(instead of the second  
%n

```
create table FMTSAMPLE (
  ID Integer(5) not null field_format=' %n%d%n',
  NAME Char(16) not null field_format=' ''%n[%^']%m',
  DEPNO Integer(4) field_format=' ', #%n%d%m',
  SALARY Double(12,2) field_format=' ; %n%f%n')
Engine=CONNECT table_type=FMT file_name='funny.txt';
```

Note that, because the statement must be terminated by

%m  
with no additional characters, skipping the ending quote of field 2 was moved from the end of the second column format to the beginning of the third column format.

The table result is:

| ID    | NAME     | DEPNO | SALARY   |
|-------|----------|-------|----------|
| 12345 | BERTRAND | 200   | 5,009.13 |

|     |                 |      |           |
|-----|-----------------|------|-----------|
| 56  | POIROT-DELMOTTE | NULL | 18,009.00 |
| 345 | NULL            | 67   | 19,000.25 |

Missing fields are replaced by null values if the column is nullable, blanks for character strings and 0 for numeric fields if it is not.

**Note 1:** Because the formats are specified between quotes, quotes belonging to the formats must be doubled or escaped to avoid a CREATE TABLE statement syntax error.

**Note 2:** Characters separating columns can be included as well in component 5 of the preceding column format or in component 1 of the succeeding column format but for blanks, which should be always included in component 1 of the succeeding column format because line trailing blanks can be sometimes lost. This is also mandatory for optional fields.

**Note 3:** Because the format is mainly used to find the sub-string corresponding to a column value, the field specification does not necessarily match the column type. For instance supposing a table contains two integer columns, NBONE and NBTWO, the two lines describing these columns could be:

```
NBONE integer(5) not null field_format=' %n%d%n',
NBTWO integer(5) field_format=' %n%s%n',
```

The first one specifies a required integer field (

%d

), the second line describes a field that can be an integer, but can be replaced by a "-" (or any other) character. Specifying the format specification for this column as a character field (

%s

) enables to recognize it with no error in all cases. Later on, this field will be converted to integer by the column read function, and a null 0 value will be generated for field specified in their format as non-numeric.

## Bad Record Error Processing

When no match is found for a column field the process aborts with a message such as:

```
Bad format line 3 field 4 of funny.txt
```

This can mean as well that one line of the input line is ill formed or that the column format for this field has been wrongly specified. When you know that your file contains records that are ill formatted and should be eliminated from normal processing, set the "maxerr" option of the CREATE TABLE statement, for instance:

```
Option_list='maxerr=100'
```

This will indicate that no error message be raised for the 100 first wrong lines. You can set Maxerr to a number greater than the number of wrong lines in your files to ignore them and get no errors.

Additionally, the "accept" option permit to keep those ill formatted lines with the bad field, and all succeeding fields of the record, nullified. If "accept" is specified without "maxerr", all ill formatted lines will be accepted.

**Note:** This error processing also applies to CSV tables.

## Fields Containing a Formatted Date

A special case is one of columns containing a formatted date. In this case, two formats must be specified:

1. The field recognition format used to delimit the date in the input record.
2. The date format used to interpret the date.
3. The field length option if the date representation is different than the standard type size.

For example, let us suppose we have a web log source file containing records such as:

```
165.91.215.31 - - [17/Jul/2001:00:01:13 -0400] - "GET /usnews/home.htm HTTP/1.1" 302
```

The create table statement shall be like this:

```
create table WEBSAMP (
  IP char(15) not null field_format=' %n%s%n',
  DATE datetime not null field_format=' -- [%n%s% -0400]' 
  date_format='DD/MMM/YYYY:hh:mm:ss' field_length=20,
  FILE char(128) not null field_format=' - "GET %n%s%n',
  HTTP double(4,2) not null field_format=' HTTP/%n%f%n',
  NBONE int(5) not null field_format=' %n%d%n')
Engine=CONNECT table_type=FMT lrecl=400
file_name='e:\\data\\token\\Websamp.dat';
```

**Note 1:** Here,

```
field_length=20  
was necessary because the default size for datetime columns is only 19. The  
lrec1=400  
was also specified because the actual file contains more information in each records making the record size calculated by default too small.
```

**Note 2:** The file name could have been specified as

```
'e:/data/token/Websamp.dat'
```

**Note 3:** FMT tables are currently read only.

## 4.3.7.6.10 CONNECT - NoSQL Table Types

They are based on files that do not match the relational format but often represent hierarchical data. CONNECT can handle **JSON**, **INI-CFG**, **XML**, and some **HTML** files.

The way it is done is different from what MySQL or PostgreSQL does. In addition to including in a table some column values of a specific data format (JSON, XML) to be handled by specific functions, CONNECT can directly use JSON, XML or INI files that are produced by other applications, and this is the table definition that describes where and how the contained information must be retrieved.

This is also different from what MariaDB does with dynamic columns, which is close to what MySQL and PostgreSQL do with the **JSON** column type.

Note: The **LEVEL** option used with these tables should, from Connect 1.07.0002, be specified as **DEPTH**. Also, what was specified with the **FIELD\_FORMAT** column option should now also be specified using **JPATH** or **XPATH**.

## 4.3.7.6.11 CONNECT - Files Retrieved Using Rest Queries

Starting with **CONNECT version 1.07.0001**, JSON, XML and possibly CSV data files can be retrieved as results from REST queries when creating or querying such tables. This is done internally by CONNECT using the CURL program generally available on all systems (if not just install it).

This can also be done using the Microsoft Casablanca (cpprestsdk) package. To enable it, first, install the package as explained in <https://github.com/microsoft/cpprestsdk>. Then make the GetRest library (dll or so) as explained in [Making the GetRest Library](#).

Note: If both are available, cpprestsdk is used preferably because it is faster. This can be changed by specifying 'curl=1' in the option list.

Note: If you want to use this feature with an older distributed version of MariaDB not featuring REST, it is possible to add it as an OEM module as explained in [Adding the REST Feature as a Library Called by an OEM Table](#).

### Creating Tables using REST

To do so, specify the HTTP of the web client and eventually the URI of the request in the **CREATE TABLE** statement. For example, for a query returning JSON data:

```
CREATE TABLE webusers (  
    id bigint(2) NOT NULL,  
    name char(24) NOT NULL,  
    username char(16) NOT NULL,  
    email char(25) NOT NULL,  
    address varchar(256) DEFAULT NULL,  
    phone char(21) NOT NULL,  
    website char(13) NOT NULL,  
    company varchar(256) DEFAULT NULL  
) ENGINE=CONNECT DEFAULT CHARSET=utf8  
TABLE_TYPE=JSON FILE_NAME='users.json' HTTP='http://jsonplaceholder.typicode.com' URI='/users';
```

As with standard JSON tables, discovery is possible, meaning that you can leave CONNECT to define the columns by analyzing the JSON file. Here you could just do:

```
CREATE TABLE webusers  
ENGINE=CONNECT DEFAULT CHARSET=utf8  
TABLE_TYPE=JSON FILE_NAME='users.json'  
HTTP='http://jsonplaceholder.typicode.com' URI='/users';
```

For example, executing:

```
SELECT name, address FROM webusers2 LIMIT 1;
```

returns:

| name          | address                                                      |
|---------------|--------------------------------------------------------------|
| Leanne Graham | Kulas Light Apt. 556 Gwenborough 92998-3874 -37.3159 81.1496 |

Here we see that for some complex elements such as **address**, which is a Json object containing values and objects, CONNECT by default has just listed their texts separated by blanks. But it is possible to ask it to analyze in more depth the json result by adding the **DEPTH** option. For instance:

```

CREATE OR REPLACE TABLE webusers
ENGINE=CONNECT DEFAULT CHARSET=utf8
TABLE_TYPE=JSON FILE_NAME='users.json'
HTTP='http://jsonplaceholder.typicode.com' URI='/users'
OPTION_LIST='Depth=2';

```

Then the table will be created as:

```

CREATE TABLE `webusers3` (
`id` bigint(2) NOT NULL,
`name` char(24) NOT NULL,
`username` char(16) NOT NULL,
`email` char(25) NOT NULL,
`address_street` char(17) NOT NULL `JPATH`='$.address.street',
`address_suite` char(9) NOT NULL `JPATH`='$.address.suite',
`address_city` char(14) NOT NULL `JPATH`='$.address.city',
`address_zipcode` char(10) NOT NULL `JPATH`='$.address.zipcode',
`address_geo_lat` char(8) NOT NULL `JPATH`='$.address.geo.lat',
`address_geo_lng` char(9) NOT NULL `JPATH`='$.address.geo.lng',
`phone` char(21) NOT NULL,
`website` char(13) NOT NULL,
`company_name` char(18) NOT NULL `JPATH`='$.company.name',
`company_catchPhrase` char(40) NOT NULL `JPATH`='$.company.catchPhrase',
`company_bs` varchar(36) NOT NULL `JPATH`='$.company.bs'
) ENGINE=CONNECT DEFAULT CHARSET=utf8 `TABLE_TYPE`='JSON' `FILE_NAME`='users.json' `OPTION_LIST`='Depth=2' `HTTP`='http://jsonpl

```

Allowing one to get all the values of the Json result, for example:

```
SELECT name, address_city city, company_name company FROM webusers3;
```

That results in:

| name                           | city               | company           |
|--------------------------------|--------------------|-------------------|
| Leanne Graham                  | Gwenborough        | Romaguera-Crona   |
| Ervin Howell                   | Wisokyburgh        | Deckow-Crist      |
| Clementine Bauch McKenziehaven | Romaguera-Jacobson |                   |
| Patricia Lebsack               | South Elvis        | Robel-Corkery     |
| Chelsey Dietrich               | Roscoeview         | Keebler LLC       |
| Mrs. Dennis Schulist           | South Christy      | Considine-Lockman |
| Kurtis Weissnat                | Howemouth          | Johns Group       |
| Nicholas Runolfsdottir V       | Aliyaview          | Abernathy Group   |
| Glenna Reichert                | Bartholomebury     | Yost and Sons     |
| Clementina DuBuque             | Lebsackbury        | Hoeger LLC        |

Of course, the complete create table (obtained by SHOW CREATE TABLE) can later be edited to make your table return exactly what you want to get. See the [JSON table type](#) for details about what and how to specify these.

Note that such tables are read only. In addition, the data will be retrieved from the web each time you query the table with a `SELECT` statement. This is fine if the result varies each time, such as when you query a weather forecasting site. But if you want to use the retrieved file many times without reloading it, just create another table on the same file without specifying the HTTP option.

Note: For JSON tables, specifying the file name is optional and defaults to tablename.type. However, you should specify it if you want to use the file later for other tables.

See the [JSON table type](#) for changes that will occur in the new CONNECT versions (distributed in early 2021).

## 4.3.7.6.12 CONNECT JSON Table Type

## Contents

- 1. Overview
- 2. The Jpath Specification
- 3. Handling of NULL Values
- 4. Having Columns defined by Discovery
- 5. JSON Catalogue Tables
- 6. Finding the table within a JSON file
- 7. JSON File Formats
- 8. Alternate Table Arrangement
- 9. Getting and Setting JSON Representation of a Column
- 10. Create, Read, Update and Delete Operations on JSON Tables
- 11. JSON User Defined Functions
  - 1. Jfile\_Bjson
  - 2. Jfile\_Convert
  - 3. Jfile\_Make
  - 4. Json\_Array\_Add
  - 5. Json\_Array\_Add\_Values
  - 6. Json\_Array\_Delete
  - 7. Json\_Array\_Grp
  - 8. JsonContains
  - 9. JsonContains\_Path
  - 10. Json\_File
  - 11. Json\_Get\_Item
  - 12. JsonGet\_Grp\_Size
  - 13. JsonGet\_String / JsonGet\_Int / JsonGet\_Real
  - 14. Json\_Item\_Merge
  - 15. JsonLocate
  - 16. Json\_Locate\_All
  - 17. Json\_Make\_Array
  - 18. Json\_Make\_Object
  - 19. Json\_Object\_Add
  - 20. Json\_Object\_Delete
  - 21. Json\_Object\_Grp
  - 22. Json\_Object\_Key
  - 23. Json\_Object\_List
  - 24. Json\_Object\_Nonull
  - 25. Json\_Object\_Values
  - 26. JsonSet\_Grp\_Size
  - 27. Json\_Set\_Item / Json\_Insert\_Item / Json\_Update\_Item
  - 28.JsonValue
- 12. The "JBIN" return type
  - 1. Using a file as json UDF first argument
  - 2. Using "Jbin" to control what the query execution does
- 13. Using JSON as Dynamic Columns
- 14. New Set of BSON Functions
- 15. Converting Tables to JSON
- 16. Converting json files
- 17. Performance Consideration
  - 1. Bjson files
- 18. Specifying a JSON table Encoding
- 19. Retrieving JSON data from MongoDB
- 20. Summary of Options and Variables Used with Json Tables
- 21. Notes

## Overview

JSON (JavaScript Object Notation) is a lightweight data-interchange format widely used on the Internet. Many applications, generally written in JavaScript or PHP use and produce JSON data, which are exchanged as files of different physical formats. JSON data is often returned from REST queries.

It is also possible to query, create or update such information in a database-like manner. MongoDB does it using a JavaScript-like language. PostgreSQL includes these facilities by using a specific data type and related functions like dynamic columns.

The CONNECT engine adds this facility to MariaDB by supporting tables based on JSON data files. This is done like for XML tables by creating tables describing what should be retrieved from the file and how it should be processed.

Starting with 1.07.0002, the internal way JSON was parsed and handled was changed. The main advantage of the new way is to reduce the memory required to parse JSON. It was from 6 to 10 times the size of the JSON source and is now only 2 to 4 times. However, this is in Beta mode and JSON

tables are still handled using the old mode. To use the new mode, tables should be created with TABLE\_TYPE=BSON. Another way is to set the `connect_force_bson` session variable to 1 or ON. Then all JSON tables will be handled as BSON. Of course, this is temporary and when successfully tested, the new way will replace the old way and all tables be created as JSON.

Let us start from the file "biblio3.json" that is the JSON equivalent of the XML Xsample file described in the XML table chapter:

```
[  
  {  
    "ISBN": "9782212090819",  
    "LANG": "fr",  
    "SUBJECT": "applications",  
    "AUTHOR": [  
      {  
        "FIRSTNAME": "Jean-Christophe",  
        "LASTNAME": "Bernadac"  
      },  
      {  
        "FIRSTNAME": "François",  
        "LASTNAME": "Knab"  
      }  
    ],  
    "TITLE": "Construire une application XML",  
    "PUBLISHER": {  
      "NAME": "Eyrolles",  
      "PLACE": "Paris"  
    },  
    "DATEPUB": 1999  
  },  
  {  
    "ISBN": "9782840825685",  
    "LANG": "fr",  
    "SUBJECT": "applications",  
    "AUTHOR": [  
      {  
        "FIRSTNAME": "William J.",  
        "LASTNAME": "Pardi"  
      }  
    ],  
    "TITLE": "XML en Action",  
    "TRANSLATED": {  
      "PREFIX": "adapté de l'anglais par",  
      "TRANSLATOR": {  
        "FIRSTNAME": "James",  
        "LASTNAME": "Guerin"  
      }  
    },  
    "PUBLISHER": {  
      "NAME": "Microsoft Press",  
      "PLACE": "Paris"  
    },  
    "DATEPUB": 1999  
  }]  
]
```

This file contains the different items existing in JSON.

- **Arrays**  
: They are enclosed in square brackets and contain a list of comma separated values.
- **Objects**  
: They are enclosed in curly brackets. They contain a comma separated list of pairs, each pair composed of a key name between double quotes, followed by a ":" character and followed by a value.
- **Values**  
: Values can be an array or an object. They also can be a string between double quotes, an integer or float number, a Boolean value or a null value. The simplest way for CONNECT to locate a table in such a file is by an array containing a list of objects (this is what MongoDB calls a collection of documents). Each array value will be a table row and each pair of the row objects will represent a column, the key being the column name and the value the column value.

A first try to create a table on this file will be to take the outer array as the table:

```

create table jsample (
ISBN char(15),
LANG char(2),
SUBJECT char(32),
AUTHOR char(128),
TITLE char(32),
TRANSLATED char(80),
PUBLISHER char(20),
DATEPUB int(4))
engine=CONNECT table_type=JSON
File_name='biblio3.json';

```

If we execute the query:

```
select isbn, author, title, publisher from jsample;
```

We get the result:

| isbn          | author                   | title                          | publisher            |
|---------------|--------------------------|--------------------------------|----------------------|
| 9782212090819 | Jean-Christophe Bernadac | Construire une application XML | Eyrolles Paris       |
| 9782840825685 | William J. Pardi         | XML en Action                  | Microsoft Press Pari |

Note that by default, column values that are objects have been set to the concatenation of all the string values of the object separated by a blank. When a column value is an array, only the first item of the array is retrieved (This will change in later versions of Connect).

However, things are generally more complicated. If JSON files do not contain attributes (although object pairs are similar to attributes) they contain a new item, arrays. We have seen that they can be used like XML multiple nodes, here to specify several authors, but they are more general because they can contain objects of different types, even it may not be advisable to do so.

This is why CONNECT enables the specification of a column field\_format option "JPATH" (FIELD\_FORMAT until Connect 1.6) that is used to describe exactly where the items to display are and how to handle arrays.

Here is an example of a new table that can be created on the same file, allowing choosing the column names, to get some sub-objects and to specify how to handle the author array.

Until Connect 1.5:

```

create table jsampall (
ISBN char(15),
Language char(2) field_format='LANG',
Subject char(32) field_format='SUBJECT',
Author char(128) field_format='AUTHOR:[ " and "]',
Title char(32) field_format='TITLE',
Translation char(32) field_format='TRANSLATOR:PREFIX',
Translator char(80) field_format='TRANSLATOR',
Publisher char(20) field_format='PUBLISHER:NAME',
Location char(16) field_format='PUBLISHER:PLACE',
Year int(4) field_format='DATEPUB')
engine=CONNECT table_type=JSON File_name='biblio3.json';

```

From Connect 1.6:

```

create table jsampall (
ISBN char(15),
Language char(2) field_format='LANG',
Subject char(32) field_format='SUBJECT',
Author char(128) field_format='AUTHOR.[ " and "]',
Title char(32) field_format='TITLE',
Translation char(32) field_format='TRANSLATOR.PREFIX',
Translator char(80) field_format='TRANSLATOR',
Publisher char(20) field_format='PUBLISHER.NAME',
Location char(16) field_format='PUBLISHER.PLACE',
Year int(4) field_format='DATEPUB')
engine=CONNECT table_type=JSON File_name='biblio3.json';

```

From Connect 1.07.0002

```

create table jsampall (
ISBN char(15),
Language char(2) jpath='$.LANG',
Subject char(32) jpath='$.SUBJECT',
Author char(128) jpath='$.AUTHOR[" and "]',
Title char(32) jpath='$.TITLE',
Translation char(32) jpath='$.TRANSLATOR.PREFIX',
Translator char(80) jpath='$.TRANSLATOR',
Publisher char(20) jpath='$.PUBLISHER.NAME',
Location char(16) jpath='$.PUBLISHER.PLACE',
Year int(4) jpath='$.DATEPUB'
engine=CONNECT table_type=JSON File_name='biblio3.json';

```

Given the query:

```
select title, author, publisher, location from jsampall;
```

The result is:

| title                          | author                                     | publisher       | location |
|--------------------------------|--------------------------------------------|-----------------|----------|
| Construire une application XML | Jean-Christophe Bernadac and François Knab | Eyrolles        | Paris    |
| XML en Action                  | William J. Pardi                           | Microsoft Press | Paris    |

Note: The JPATH was not specified for column ISBN because it defaults to the column name.

Here is another example showing that one can choose what to extract from the file and how to “expand” an array, meaning to generate one row for each array value:

Until Connect 1.5:

```

create table jsampex (
ISBN char(15),
Title char(32) field_format='TITLE',
AuthorFN char(128) field_format='AUTHOR:[X]:FIRSTNAME',
AuthorLN char(128) field_format='AUTHOR:[X]:LASTNAME',
Year int(4) field_format='DATEPUB'
engine=CONNECT table_type=JSON File_name='biblio3.json';

```

From Connect 1.6:

```

create table jsampex (
ISBN char(15),
Title char(32) field_format='TITLE',
AuthorFN char(128) field_format='AUTHOR.[X].FIRSTNAME',
AuthorLN char(128) field_format='AUTHOR.[X].LASTNAME',
Year int(4) field_format='DATEPUB'
engine=CONNECT table_type=JSON File_name='biblio3.json';

```

From Connect 1.06.006:

```

create table jsampex (
ISBN char(15),
Title char(32) field_format='TITLE',
AuthorFN char(128) field_format='AUTHOR[*].FIRSTNAME',
AuthorLN char(128) field_format='AUTHOR[*].LASTNAME',
Year int(4) field_format='DATEPUB'
engine=CONNECT table_type=JSON File_name='biblio3.json';

```

From Connect 1.07.0002

```

create table jsampex (
ISBN char(15),
Title char(32) jpath='TITLE',
AuthorFN char(128) jpath='AUTHOR[*].FIRSTNAME',
AuthorLN char(128) jpath='AUTHOR[*].LASTNAME',
Year int(4) jpath='DATEPUB'
engine=CONNECT table_type=JSON File_name='biblio3.json';

```

It is displayed as:

| ISBN | Title | AuthorFN | AuthorLN | Year |
|------|-------|----------|----------|------|
|------|-------|----------|----------|------|

|               |                                |                 |          |      |
|---------------|--------------------------------|-----------------|----------|------|
| 9782212090819 | Construire une application XML | Jean-Christophe | Bernadac | 1999 |
| 9782212090819 | Construire une application XML | François        | Knab     | 1999 |
| 9782840825685 | XML en Action                  | William J.      | Pardi    | 1999 |

Note: The example above shows that the '\$.', that means the beginning of the path, can be omitted.

## The Jpath Specification

From Connect 1.6, the Jpath specification has changed to be the one of the native JSON functions and more compatible with what is generally used. It is close to the standard definition and compatible to what MongoDB and other products do. The ':' separator is replaced by '.'. Position in array is accepted MongoDB style with no square brackets. Array specification specific to CONNECT are still accepted but [\*] is used for expanding and [x] for multiply. However, tables created with the previous syntax can still be used by adding SEP\_CHAR=':' (can be done with alter table). Also, it can be now specified as JPATH (was FIELD\_FORMAT) but FIELD\_FORMAT is still accepted.

Until Connect 1.5, it is the description of the path to follow to reach the required item. Each step is the key name (case sensitive) of the pair when crossing an object, and the number of the value between square brackets when crossing an array. Each specification is separated by a ':' character.

From Connect 1.6, It is the description of the path to follow to reach the required item. Each step is the key name (case sensitive) of the pair when crossing an object, and the position number of the value when crossing an array. Key specifications are separated by a '.' character.

For instance, in the above file, the last name of the second author of a book is reached by:

`$AUTHOR[1].LASTNAME` standard style `$AUTHOR.1.LASTNAME` MongoDB style `AUTHOR:[1]:LASTNAME` old style when `SEP_CHAR=':'` or until Connect 1.5

The '\$' or "\$." prefix specifies the root of the path and can be omitted with CONNECT.

The array specification can also indicate how it must be processed:

For instance, in the above file, the last name of the second author of a book is reached by:

`AUTHOR:[1]:LASTNAME`

The array specification can also indicate how it must be processed:

| Specification                                     | Array Type | Limit | Description                                                                                               |
|---------------------------------------------------|------------|-------|-----------------------------------------------------------------------------------------------------------|
| n (Connect >= 1.6) or [n]                         | All        | N.A   | Take the nth value of the array.                                                                          |
| [*] (Connect >= 1.6), [X] or [x] (Connect <= 1.5) | All        |       | Expand. Generate one row for each array value.                                                            |
| ["string"]                                        | String     |       | Concatenate all values separated by the specified string.                                                 |
| [+]                                               | Numeric    |       | Make the sum of all the non-null array values.                                                            |
| [x] (Connect >= 1.6), [*] (Connect <= 1.5)        | Numeric    |       | Make the product of all non-null array values.                                                            |
| [!]                                               | Numeric    |       | Make the average of all the non-null array values.                                                        |
| [>] or [<]                                        | All        |       | Return the greatest or least non-null value of the array.                                                 |
| [#]                                               | All        | N.A   | Return the number of values in the array.                                                                 |
| []                                                | All        |       | Expand if under an expanded object. Otherwise sum if numeric, else concatenation separated by ", ".       |
|                                                   | All        |       | Between two separators, if an array, expand it if under an expanded object or take the first value of it. |

Note 1: When the LIMIT restriction is applicable, only the first *m* array items are used, *m* being the value of the LIMIT option (to be specified in option\_list). The LIMIT default value is 10.

Note 2: An alternative way to indicate what is to be expanded is to use the expand option in the option list, for instance:

`OPTION_LIST='Expand=AUTHOR'`

`AUTHOR`

is here the key of the pair that has the array as a value (case sensitive). Expand is limited to only one branch (expanded arrays must be under the same object).

Let us take as an example the file

`expense.json`

([found here](#)). The table jexpall expands all under and including the week array:

From Connect 1.07.0002

```
create table jexpall (
WHO char(12),
WEEK int(2) jpath='$.WEEK[*].NUMBER',
WHAT char(32) jpath='$.WEEK[*].EXPENSE[*].WHAT',
AMOUNT double(8,2) jpath='$.WEEK[*].EXPENSE[*].AMOUNT')
engine=CONNECT table_type=JSON File_name='expense.json';
```

From Connect.1.6

```
create table jexpall (
WHO char(12),
WEEK int(2) field_format='$.WEEK[*].NUMBER',
WHAT char(32) field_format='$.WEEK[*].EXPENSE[*].WHAT',
AMOUNT double(8,2) field_format='$.WEEK[*].EXPENSE[*].AMOUNT')
engine=CONNECT table_type=JSON File_name='expense.json';
```

Until Connect 1.5:

```
create table jexpall (
WHO char(12),
WEEK int(2) field_format='WEEK:[x]:NUMBER',
WHAT char(32) field_format='WEEK:[x]:EXPENSE:[x]:WHAT',
AMOUNT double(8,2) field_format='WEEK:[x]:EXPENSE:[x]:AMOUNT'
engine=CONNECT table_type=JSON File_name='expense.json';
```

| WHO   | WEEK | WHAT | AMOUNT |
|-------|------|------|--------|
| Joe   | 3    | Beer | 18.00  |
| Joe   | 3    | Food | 12.00  |
| Joe   | 3    | Food | 19.00  |
| Joe   | 3    | Car  | 20.00  |
| Joe   | 4    | Beer | 19.00  |
| Joe   | 4    | Beer | 16.00  |
| Joe   | 4    | Food | 17.00  |
| Joe   | 4    | Food | 17.00  |
| Joe   | 4    | Beer | 14.00  |
| Joe   | 5    | Beer | 14.00  |
| Joe   | 5    | Food | 12.00  |
| Beth  | 3    | Beer | 16.00  |
| Beth  | 4    | Food | 17.00  |
| Beth  | 4    | Beer | 15.00  |
| Beth  | 5    | Food | 12.00  |
| Beth  | 5    | Beer | 20.00  |
| Janet | 3    | Car  | 19.00  |
| Janet | 3    | Food | 18.00  |
| Janet | 3    | Beer | 18.00  |
| Janet | 4    | Car  | 17.00  |
| Janet | 5    | Beer | 14.00  |
| Janet | 5    | Car  | 12.00  |
| Janet | 5    | Beer | 19.00  |
| Janet | 5    | Food | 12.00  |

The table

jexpw

shows what was bought and the sum and average of amounts for each person and week:

From Connect 1.07.0002

```

create table jexpw (
WHO char(12) not null,
WEEK int(2) not null jpath='$.WEEK[*].NUMBER',
WHAT char(32) not null jpath='$.WEEK[].EXPENSE[", "] .WHAT',
SUM double(8,2) not null jpath='$.WEEK[].EXPENSE[+].AMOUNT',
AVERAGE double(8,2) not null jpath='$.WEEK[].EXPENSE[!].AMOUNT')
engine=CONNECT table_type=JSON File_name='expense.json';

```

From Connect 1.6:

```

create table jexpw (
WHO char(12) not null,
WEEK int(2) not null field_format='$.WEEK[*].NUMBER',
WHAT char(32) not null field_format='$.WEEK[].EXPENSE[", "] .WHAT',
SUM double(8,2) not null field_format='$.WEEK[].EXPENSE[+].AMOUNT',
AVERAGE double(8,2) not null field_format='$.WEEK[].EXPENSE[!].AMOUNT')
engine=CONNECT table_type=JSON File_name='expense.json';

```

Until Connect 1.5:

```

create table jexpw (
WHO char(12) not null,
WEEK int(2) not null field_format='WEEK:[x]:NUMBER',
WHAT char(32) not null field_format='WEEK::EXPENSE[", "] :WHAT',
SUM double(8,2) not null field_format='WEEK::EXPENSE[+]:AMOUNT',
AVERAGE double(8,2) not null field_format='WEEK::EXPENSE[!]:AMOUNT')
engine=CONNECT table_type=JSON File_name='expense.json';

```

| WHO   | WEEK | WHAT                         | SUM   | AVERAGE |
|-------|------|------------------------------|-------|---------|
| Joe   | 3    | Beer, Food, Food, Car        | 69.00 | 17.25   |
| Joe   | 4    | Beer, Beer, Food, Food, Beer | 83.00 | 16.60   |
| Joe   | 5    | Beer, Food                   | 26.00 | 13.00   |
| Beth  | 3    | Beer                         | 16.00 | 16.00   |
| Beth  | 4    | Food, Beer                   | 32.00 | 16.00   |
| Beth  | 5    | Food, Beer                   | 32.00 | 16.00   |
| Janet | 3    | Car, Food, Beer              | 55.00 | 18.33   |
| Janet | 4    | Car                          | 17.00 | 17.00   |
| Janet | 5    | Beer, Car, Beer, Food        | 57.00 | 14.25   |

Let us see what the table

```

jexpz
does:

```

From Connect 1.6:

```

create table jexpz (
WHO char(12) not null,
WEEKS char(12) not null field_format='WEEK[", "] .NUMBER',
SUMS char(64) not null field_format='WEEK["+"].EXPENSE[+].AMOUNT',
SUM double(8,2) not null field_format='WEEK[+].EXPENSE[+].AMOUNT',
AVGS char(64) not null field_format='WEEK["+"].EXPENSE[!].AMOUNT',
SUMAVG double(8,2) not null field_format='WEEK[+].EXPENSE[!].AMOUNT',
AVGSUM double(8,2) not null field_format='WEEK[!].EXPENSE[+].AMOUNT',
AVERAGE double(8,2) not null field_format='WEEK[!].EXPENSE[*].AMOUNT')
engine=CONNECT table_type=JSON File_name='expense.json';

```

From Connect 1.07.0002

```

create table jexpz (
WHO char(12) not null,
WEEKS char(12) not null jpath='WEEK[", "].NUMBER',
SUMS char(64) not null jpath='WEEK["+"].EXPENSE[+].AMOUNT',
SUM double(8,2) not null jpath='WEEK[+].EXPENSE[+].AMOUNT',
AVGS char(64) not null jpath='WEEK["+"].EXPENSE[!].AMOUNT',
SUMAVG double(8,2) not null jpath='WEEK[+].EXPENSE[!].AMOUNT',
AVGSUM double(8,2) not null jpath='WEEK[!].EXPENSE[+].AMOUNT',
AVERAGE double(8,2) not null jpath='WEEK[!].EXPENSE[*].AMOUNT')
engine=CONNECT table_type=JSON File_name='expense.json';

```

Until Connect 1.5:

```

create table jexpz (
WHO char(12) not null,
WEEKS char(12) not null field_format='WEEK:[", "]:NUMBER',
SUMS char(64) not null field_format='WEEK:[+]::EXPENSE:[+]:AMOUNT',
SUM double(8,2) not null field_format='WEEK:[+]:EXPENSE:[+]:AMOUNT',
AVGS char(64) not null field_format='WEEK:[+]::EXPENSE:[!]:AMOUNT',
SUMAVG double(8,2) not null field_format='WEEK:[+]:EXPENSE:[!]:AMOUNT',
AVGSUM double(8,2) not null field_format='WEEK:[!]:EXPENSE:[+]:AMOUNT',
AVERAGE double(8,2) not null field_format='WEEK:[!]:EXPENSE:[x]:AMOUNT'
engine=CONNECT table_type=JSON
File_name='E:/Data/Json/expense2.json';

```

| WHO   | WEEKS   | SUMS              | SUM    | AVGS              | SUMAVG | AVGSUM | AVERAGE |
|-------|---------|-------------------|--------|-------------------|--------|--------|---------|
| Joe   | 3, 4, 5 | 69.00+83.00+26.00 | 178.00 | 17.25+16.60+13.00 | 46.85  | 59.33  | 16.18   |
| Beth  | 3, 4, 5 | 16.00+32.00+32.00 | 80.00  | 16.00+16.00+16.00 | 48.00  | 26.67  | 16.00   |
| Janet | 3, 4, 5 | 55.00+17.00+57.00 | 129.00 | 18.33+17.00+14.25 | 49.58  | 43.00  | 16.12   |

For all persons:

- Column 1 show the person name.
- Column 2 shows the weeks for which values are calculated.
- Column 3 lists the sums of expenses for each week.
- Column 4 calculates the sum of all expenses by person.
- Column 5 shows the week's expense averages.
- Column 6 calculates the sum of these averages.
- Column 7 calculates the average of the week's sum of expenses.
- Column 8 calculates the average expense by person.

It would be very difficult, if even possible, to obtain this result from table

```

jexpall
using an SQL query.

```

## Handling of NULL Values

Json has a null explicit value that can be met in arrays or object key values. When regarding json as a relational table, a column value can be null because the corresponding json item is explicitly null, or implicitly because the corresponding item is missing in an array or object. CONNECT does not make any distinction between explicit and implicit nulls.

However, it is possible to specify how nulls are handled and represented. This is done by setting the string session variable `connect_json_null`. The default value of `connect_json_null` is "`<null>`"; it can be changed, for instance, by:

```
SET connect_json_null='NULL';
```

This changes its representation when a column displays the text of an object or the concatenation of the values of an array.

It is also possible to tell CONNECT to ignore nulls by:

```
SET connect_json_null=NULL;
```

When doing so, nulls do not appear in object text or array lists. However, this does not change the behavior of array calculation nor the result of array count.

## Having Columns defined by Discovery

It is possible to let the MariaDB discovery process do the job of column specification. When columns are not defined in the create table statement, CONNECT endeavors to analyze the JSON file and to provide the column specifications. This is possible only for tables represented by an array of objects because CONNECT retrieves the column names from the object pair keys and their definition from the object pair values. For instance, the

jsample table could be created saying:

```
create table jsample engine=connect table_type=JSON file_name='biblio3.json';
```

Let's check how it was actually specified using the show create table statement:

```
CREATE TABLE `jsample` (
  `ISBN` char(13) NOT NULL,
  `LANG` char(2) NOT NULL,
  `SUBJECT` char(12) NOT NULL,
  `AUTHOR` varchar(256) DEFAULT NULL,
  `TITLE` char(30) NOT NULL,
  `TRANSLATED` varchar(256) DEFAULT NULL,
  `PUBLISHER` varchar(256) DEFAULT NULL,
  `DATEPUB` int(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON' `FILE_NAME`='biblio3.json';
```

It is equivalent except for the column sizes that have been calculated from the file as the maximum length of the corresponding column when it was a normal value. For columns that are json arrays or objects, the column is specified as a varchar string of length 256, supposedly big enough to contain the sub-object's concatenated values. Nullable is set to true if the column is null or missing in some rows or if its JPATH contains arrays.

If a more complex definition is desired, you can ask CONNECT to analyse the JPATH up to a given depth using the DEPTH or LEVEL option in the option list. Its default value is 0 but can be changed setting the [connect\\_default\\_depth](#) session variable (in future versions the default will be 5). The depth value is the number of sub-objects that are taken in the JPATH2 (this is different from what is defined and returned by the native [Json\\_Depth](#) function).

For instance:

```
create table jsampall2 engine=connect table_type=JSON
file_name='biblio3.json' option_list='level=1';
```

This will define the table as:

From Connect 1.07.0002

```
CREATE TABLE `jsampall2` (
  `ISBN` char(13) NOT NULL,
  `LANG` char(2) NOT NULL,
  `SUBJECT` char(12) NOT NULL,
  `AUTHOR_FIRSTNAME` char(15) NOT NULL `JPATH`='$.AUTHOR.[0].FIRSTNAME',
  `AUTHOR_LASTNAME` char(8) NOT NULL `JPATH`='$.AUTHOR.[0].LASTNAME',
  `TITLE` char(30) NOT NULL,
  `TRANSLATED_PREFIX` char(23) DEFAULT NULL `JPATH`='$.TRANSLATED.PREFIX',
  `TRANSLATED_TRANSLATOR` varchar(256) DEFAULT NULL `JPATH`='$.TRANSLATED.TRANSLATOR',
  `PUBLISHER_NAME` char(15) NOT NULL `JPATH`='$.PUBLISHER.NAME',
  `PUBLISHER_PLACE` char(5) NOT NULL `JPATH`='$.PUBLISHER.PLACE',
  `DATEPUB` int(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON'
`FILE_NAME`='biblio3.json' `OPTION_LIST`='depth=1';
```

From Connect 1.6:

```
CREATE TABLE `jsampall2` (
  `ISBN` char(13) NOT NULL,
  `LANG` char(2) NOT NULL,
  `SUBJECT` char(12) NOT NULL,
  `AUTHOR_FIRSTNAME` char(15) NOT NULL `FIELD_FORMAT`='AUTHOR..FIRSTNAME',
  `AUTHOR_LASTNAME` char(8) NOT NULL `FIELD_FORMAT`='AUTHOR..LASTNAME',
  `TITLE` char(30) NOT NULL,
  `TRANSLATED_PREFIX` char(23) DEFAULT NULL `FIELD_FORMAT`='TRANSLATED.PREFIX',
  `TRANSLATED_TRANSLATOR` varchar(256) DEFAULT NULL `FIELD_FORMAT`='TRANSLATED.TRANSLATOR',
  `PUBLISHER_NAME` char(15) NOT NULL `FIELD_FORMAT`='PUBLISHER.NAME',
  `PUBLISHER_PLACE` char(5) NOT NULL `FIELD_FORMAT`='PUBLISHER.PLACE',
  `DATEPUB` int(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON'
`FILE_NAME`='biblio3.json' `OPTION_LIST`='level=1';
```

Until Connect 1.5:

```

CREATE TABLE `jsampall2` (
  `ISBN` char(13) NOT NULL,
  `LANG` char(2) NOT NULL,
  `SUBJECT` char(12) NOT NULL,
  `AUTHOR_FIRSTNAME` char(15) NOT NULL `FIELD_FORMAT`='AUTHOR::FIRSTNAME',
  `AUTHOR_LASTNAME` char(8) NOT NULL `FIELD_FORMAT`='AUTHOR::LASTNAME',
  `TITLE` char(30) NOT NULL,
  `TRANSLATED_PREFIX` char(23) DEFAULT NULL `FIELD_FORMAT`='TRANSLATED:PREFIX',
  `TRANSLATED_TRANSLATOR` varchar(256) DEFAULT NULL `FIELD_FORMAT`='TRANSLATED:TRANSLATOR',
  `PUBLISHER_NAME` char(15) NOT NULL `FIELD_FORMAT`='PUBLISHER:NAME',
  `PUBLISHER_PLACE` char(5) NOT NULL `FIELD_FORMAT`='PUBLISHER:PLACE',
  `DATEPUB` int(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON' `FILE_NAME`='biblio3.json' `OPTION_LIST`='level=1';

```

For columns that are a simple value, the Json path is the column name. This is the default when the Jpath option is not specified, so it was not specified for such columns. However, you can force discovery to specify it by setting the connect\_all\_path variable to 1 or ON. This can be useful if you plan to change the name of such columns and relieves you of manually specifying the path (otherwise it would default to the new name and cause the column to not or wrongly be found).

Another problem is that CONNECT cannot guess what you want to do with arrays. Here the AUTHOR array is set to 0, which means that only its first value will be retrieved unless you also had specified "Expand=AUTHOR" in the option list. But of course, you can replace it with anything else.

This method can be used as a quick way to make a "template" table definition that can later be edited to make the desired definition. In particular, column names are constructed from all the object keys of their path in order to have distinct column names. This can be manually edited to have the desired names, provided their JPATH key names are not modified.

DEPTH can also be given the value -1 to create only columns that are simple values (no array or object). It normally defaults to 0 but this can be modified setting the `connect_default_depth` variable.

Note: Since version 1.6.4, CONNECT eliminates columns that are "void" or whose type cannot be determined. For instance given the file sresto.json:

```
{"_id":1,"name":"Corner Social","cuisine":"American","grades":[{"grade":"A","score":6}]}
{"_id":2,"name":"La Nueva Clasica Antillana","cuisine":"Spanish","grades":[]}
```

Previously, when using discovery, creating the table by:

```
create table sjr0
engine=connect table_type=JSON file_name='sresto.json'
option_list='Pretty=0,Depth=1' lrecl=128;
```

The table was previously created as:

```

CREATE TABLE `sjr0` (
  `_id` bigint(1) NOT NULL,
  `name` char(26) NOT NULL,
  `cuisine` char(8) NOT NULL,
  `grades` char(1) DEFAULT NULL,
  `grades_grade` char(1) DEFAULT NULL `JPATH`='$.grades[0].grade',
  `grades_score` bigint(1) DEFAULT NULL `JPATH`='$.grades[0].score'
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='JSON'
`FILE_NAME`='sresto.json'
`OPTION_LIST`='Pretty=0,Depth=1,Accept=1' `LRECL`=128;

```

The column "grades" was added because of the void array in line 2. Now this column is skipped and does not appear anymore (unless the option

```
Accept=1
is added in the option list).
```

## JSON Catalogue Tables

Another way to see JSON table column specifications is to use a catalogue table. For instance:

```
create table bibcol engine=connect table_type=JSON file_name='biblio3.json'
option_list='level=2' catfunc=columns;
select column_name, type_name type, column_size size, jpath from bibcol;
```

which returns:

From Connect 1.07.0002:

| column_name | type | size | jpath   |
|-------------|------|------|---------|
| ISBN        | CHAR | 13   | \$.ISBN |

|                                 |         |    |                                    |
|---------------------------------|---------|----|------------------------------------|
| LANG                            | CHAR    | 2  | \$.LANG                            |
| SUBJECT                         | CHAR    | 12 | \$.SUBJECT                         |
| AUTHOR_FIRSTNAME                | CHAR    | 15 | \$.AUTHOR[0].FIRSTNAME             |
| AUTHOR_LASTNAME                 | CHAR    | 8  | \$.AUTHOR[0].LASTNAME              |
| TITLE                           | CHAR    | 30 | \$.TITLE                           |
| TRANSLATED_PREFIX               | CHAR    | 23 | \$.TRANSLATED.PREFIX               |
| TRANSLATED_TRANSLATOR_FIRSTNAME | CHAR    | 5  | \$.TRANSLATED.TRANSLATOR.FIRSTNAME |
| TRANSLATED_TRANSLATOR_LASTNAME  | CHAR    | 6  | \$.TRANSLATED.TRANSLATOR.LASTNAME  |
| PUBLISHER_NAME                  | CHAR    | 15 | \$.PUBLISHER.NAME                  |
| PUBLISHER_PLACE                 | CHAR    | 5  | \$.PUBLISHER.PLACE                 |
| DATEPUB                         | INTEGER | 4  | \$.DATEPUB                         |

From Connect 1.6:

| column_name                     | type    | size | jpath                           |
|---------------------------------|---------|------|---------------------------------|
| ISBN                            | CHAR    | 13   |                                 |
| LANG                            | CHAR    | 2    |                                 |
| SUBJECT                         | CHAR    | 12   |                                 |
| AUTHOR_FIRSTNAME                | CHAR    | 15   | AUTHOR..FIRSTNAME               |
| AUTHOR_LASTNAME                 | CHAR    | 8    | AUTHOR..LASTNAME                |
| TITLE                           | CHAR    | 30   |                                 |
| TRANSLATED_PREFIX               | CHAR    | 23   | TRANSLATED.PREFIX               |
| TRANSLATED_TRANSLATOR_FIRSTNAME | CHAR    | 5    | TRANSLATED.TRANSLATOR.FIRSTNAME |
| TRANSLATED_TRANSLATOR_LASTNAME  | CHAR    | 6    | TRANSLATED.TRANSLATOR.LASTNAME  |
| PUBLISHER_NAME                  | CHAR    | 15   | PUBLISHER.NAME                  |
| PUBLISHER_PLACE                 | CHAR    | 5    | PUBLISHER.PLACE                 |
| DATEPUB                         | INTEGER | 4    |                                 |

Until Connect 1.5:

| column_name                     | type    | size | jpath                           |
|---------------------------------|---------|------|---------------------------------|
| ISBN                            | CHAR    | 13   |                                 |
| LANG                            | CHAR    | 2    |                                 |
| SUBJECT                         | CHAR    | 12   |                                 |
| AUTHOR_FIRSTNAME                | CHAR    | 15   | AUTHOR::FIRSTNAME               |
| AUTHOR_LASTNAME                 | CHAR    | 8    | AUTHOR::LASTNAME                |
| TITLE                           | CHAR    | 30   |                                 |
| TRANSLATED_PREFIX               | CHAR    | 23   | TRANSLATED:PREFIX               |
| TRANSLATED_TRANSLATOR_FIRSTNAME | CHAR    | 5    | TRANSLATED:TRANSLATOR:FIRSTNAME |
| TRANSLATED_TRANSLATOR_LASTNAME  | CHAR    | 6    | TRANSLATED:TRANSLATOR:LASTNAME  |
| PUBLISHER_NAME                  | CHAR    | 15   | PUBLISHER:NAME                  |
| PUBLISHER_PLACE                 | CHAR    | 5    | PUBLISHER:PLACE                 |
| DATEPUB                         | INTEGER | 4    |                                 |

All this is mostly useful when creating a table on a remote file that you cannot easily see.

## Finding the table within a JSON file

Given the file "facebook.json":

```
{
  "data": [
    {
      "id": "X999_Y999",
      "from": {
        "name": "Tom Brady", "id": "X12"
      },
      "message": "Looking forward to 2010!",
      "actions": [
        {
          "name": "Comment",
          "link": "http://www.facebook.com/X999/posts/Y999"
        },
        {
          "name": "Like",
          "link": "http://www.facebook.com/X999/posts/Y999"
        }
      ],
      "type": "status",
      "created_time": "2010-08-02T21:27:44+0000",
      "updated_time": "2010-08-02T21:27:44+0000"
    },
    {
      "id": "X998_Y998",
      "from": {
        "name": "Peyton Manning", "id": "X18"
      },
      "message": "Where's my contract?",
      "actions": [
        {
          "name": "Comment",
          "link": "http://www.facebook.com/X998/posts/Y998"
        },
        {
          "name": "Like",
          "link": "http://www.facebook.com/X998/posts/Y998"
        }
      ],
      "type": "status",
      "created_time": "2010-08-02T21:27:44+0000",
      "updated_time": "2010-08-02T21:27:44+0000"
    }
  ]
}
```

The table we want to analyze is represented by the array value of the “data” object. Here is how this is specified in the create table statement:

From Connect 1.07.0002:

```
create table jfacebook (
`ID` char(10) jpath='id',
`Name` char(32) jpath='from.name',
`MyID` char(16) jpath='from.id',
`Message` varchar(256) jpath='message',
`Action` char(16) jpath='actions..name',
`Link` varchar(256) jpath='actions..link',
`Type` char(16) jpath='type',
`Created` datetime date_format='YYYY-MM-DD\T\hh:mm:ss' jpath='created_time',
`Updated` datetime date_format='YYYY-MM-DD\T\hh:mm:ss' jpath='updated_time')
engine=connect table_type=JSON file_name='facebook.json' option_list='Object=data,Expand=actions';
```

From Connect 1.6:

```
create table jfacebook (
`ID` char(10) field_format='id',
`Name` char(32) field_format='from.name',
`MyID` char(16) field_format='from.id',
`Message` varchar(256) field_format='message',
`Action` char(16) field_format='actions..name',
`Link` varchar(256) field_format='actions..link',
`Type` char(16) field_format='type',
`Created` datetime date_format='YYYY-MM-DD\T\hh:mm:ss' field_format='created_time',
`Updated` datetime date_format='YYYY-MM-DD\T\hh:mm:ss' field_format='updated_time')
engine=connect table_type=JSON file_name='facebook.json' option_list='Object=data,Expand=actions';
```

Until Connect 1.5:

```
create table jfacebook (
`ID` char(10) field_format='id',
`Name` char(32) field_format='from:name',
`MyID` char(16) field_format='from:id',
`Message` varchar(256) field_format='message',
`Action` char(16) field_format='actions::name',
`Link` varchar(256) field_format='actions::link',
`Type` char(16) field_format='type',
`Created` datetime date_format='YYYY-MM-DD\T\hh:mm:ss' field_format='created_time',
`Updated` datetime date_format='YYYY-MM-DD\T\hh:mm:ss' field_format='updated_time')
engine=connect table_type=JSON file_name='facebook.json' option_list='Object=data,Expand=actions';
```

This is the object option that gives the Jpath of the table. Note also an alternate way to declare the array to be expanded by the expand option of the option\_list.

Because some string values contain a date representation, the corresponding columns are declared as datetime and the date format is specified for them.

The Jpath of the object option has the same syntax as the column Jpath but of course all array steps must be specified using the [n] (until Connect 1.5) or n (from Connect 1.6) format.

Note: This applies to the whole document for tables having

PRETTY = 2

(see below). Otherwise, it applies to the document objects of each file records.

## JSON File Formats

The examples we have seen so far are files that, even they can be formatted in different ways (blanks, tabs, carriage return and line feed are ignored when parsing them), respect the JSON syntax and are made of only one item (Object or Array). Like for XML files, they are entirely parsed and a memory representation is made used to process them. This implies that they are of reasonable size to avoid an out of memory condition. Tables based on such files are recognized by the option Pretty=2 that we did not specify above because this is the default.

An alternate format, which is the format of exported MongoDB files, is a file where each row is physically stored in one file record. For instance:

```
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51564999999999, 42.377017 ], "pop" : 36963, "state" : "MA" }
{ "_id" : "01005", "city" : "BARRE", "loc" : [ -72.1083540000001, 42.409698 ], "pop" : 4546, "state" : "MA" }
{ "_id" : "01007", "city" : "BELCHERTOWN", "loc" : [ -72.4109530000001, 42.275103 ], "pop" : 10579, "state" : "MA" }
...
{ "_id" : "99929", "city" : "WRANGELL", "loc" : [ -132.352918, 56.433524 ], "pop" : 2573, "state" : "AK" }
{ "_id" : "99950", "city" : "KETCHIKAN", "loc" : [ -133.18479, 55.942471 ], "pop" : 422, "state" : "AK" }
```

The original file, "cities.json", has 29352 records. To base a table on this file we must specify the option Pretty=0 in the option list. For instance:

From Connect 1.07.0002:

```
create table cities (
`_id` char(5) key,
`city` char(32),
`lat` double(12,6) jpath='loc.0',
`long` double(12,6) jpath='loc.1',
`pop` int(8),
`state` char(2) distrib='clustered')
engine=CONNECT table_type=JSON file_name='cities.json' lrec=128 option_list='pretty=0';
```

From Connect 1.6:

```
create table cities (
`_id` char(5) key,
`city` char(32),
`lat` double(12,6) field_format='loc.0',
`long` double(12,6) field_format='loc.1',
`pop` int(8),
`state` char(2) distrib='clustered')
engine=CONNECT table_type=JSON file_name='cities.json' lrec=128 option_list='pretty=0';
```

Until Connect 1.5:

```

create table cities (
`_id` char(5) key,
`city` char(32),
`long` double(12,6) field_format='loc:[0]',
`lat` double(12,6) field_format='loc:[1]',
`pop` int(8),
`state` char(2) distrib='clustered')
engine=CONNECT table_type=JSON file_name='cities.json' lrecl=128 option_list='pretty=0';

```

Note the use of [n] (until Connect 1.5) or n (from Connect 1.6) array specifications for the longitude and latitude columns.

When using this format, the table is processed by CONNECT like a DOS, CSV or FMT table. Rows are retrieved and parsed by records and the table can be very large. Another advantage is that such a table can be indexed, which can be of great value for very large tables. The “distrib” option of the “state” column tells CONNECT to use block indexing when possible.

For such tables – as well as for pretty=1 ones – the record size must be specified using the LRECL option. Be sure you don’t specify it too small as it is used to allocate the read/write buffers and the memory used for parsing the rows. If in doubt, be generous as it does not cost much in memory allocation.

Another format exists, noted by Pretty=1, which is similar to this one but has some additions to represent a JSON array. A header and a trailer records are added containing the opening and closing square bracket, and all records but the last are followed by a comma. It has the same advantages for reading and updating, but inserting and deleting are executed in the pretty=2 way.

## Alternate Table Arrangement

We have seen that the most natural way to represent a table in a JSON file is to make it on an array of objects. However, other possibilities exist. A table can be an array of arrays, a one column table can be an array of values, or a one row table can be just one object or one value. Single row tables are internally handled by adding a one value array around them.

Let us see how to handle, for instance, a table that is an array of arrays. The file:

```

[
  [56, "Coucou", 500.00],
  [[2,0,1,4], "Hello World", 2.0316],
  ["1784", "John Doo", 32.4500],
  [1914, ["Nabucho","donosor"], 5.12],
  [7, "sept", [0.77,1.22,2.01]],
  [8, "huit", 13.0]
]
```

A table can be created on this file as:

From Connect 1.07.0002:

```

create table xjson (
`a` int(6) jpath='1',
`b` char(32) jpath='2',
`c` double(10,4) jpath='3')
engine=connect table_type=JSON file_name='test.json' option_list='Pretty=1,Jmode=1,Base=1' lrecl=128;

```

From Connect 1.6:

```

create table xjson (
`a` int(6) field_format='1',
`b` char(32) field_format='2',
`c` double(10,4) field_format='3')
engine=connect table_type=JSON file_name='test.json' option_list='Pretty=1,Jmode=1,Base=1' lrecl=128;

```

Until Connect 1.5:

```

create table xjson (
`a` int(6) field_format='[1]',
`b` char(32) field_format='[2]',
`c` double(10,4) field_format='[3]')
engine=connect table_type=JSON file_name='test.json'
option_list='Pretty=1,Jmode=1,Base=1' lrecl=128;

```

Columns are specified by their position in the row arrays. By default, this is zero-based but for this table the base was set to 1 by the Base option of the option list. Another new option in the option list is Jmode=1. It indicates what type of table this is. The Jmode values are:

1. An array of objects. This is the default.
2. An array of Array. Like this one.
3. An array of values.

When reading, this is not required as the type of the array items is specified for the columns; however, it is required when inserting new rows so

CONNECT knows what to insert. For instance:

```
insert into xjson values(25, 'Breakfast', 1.414);
```

After this, it is displayed as:

| a    | b           | c        |
|------|-------------|----------|
| 56   | Coucou      | 500.0000 |
| 2    | Hello World | 2.0316   |
| 1784 | John Doo    | 32.4500  |
| 1914 | Nabucho     | 5.1200   |
| 7    | sept        | 0.7700   |
| 8    | huit        | 13.0000  |
| 25   | Breakfast   | 1.4140   |

Unspecified array values are represented by their first element.

## Getting and Setting JSON Representation of a Column

We have seen that columns corresponding to a Json object or array are retrieved by default as the concatenation of all its values separated by a blank. It is also possible to retrieve and display such column contains as the full JSON string corresponding to it in the JSON file. This is specified in the JPATH by a "\*" where the object or array would be specified.

Note: When having columns generated by discovery, this can be specified by adding the STRINGIFY option to ON or 1 in the option list.

For instance:

From Connect 1.07.0002:

```
create table jsample2 (
ISBN char(15),
Lng char(2) jpath='LANG',
json_Author char(255) jpath='AUTHOR.*',
Title char(32) jpath='TITLE',
Year int(4) jpath='DATEPUB')
engine=CONNECT table_type=JSON file_name='biblio3.json';
```

From Connect 1.6:

```
create table jsample2 (
ISBN char(15),
Lng char(2) field_format='LANG',
json_Author char(255) field_format='AUTHOR.*',
Title char(32) field_format='TITLE',
Year int(4) field_format='DATEPUB')
engine=CONNECT table_type=JSON file_name='biblio3.json';
```

Until Connect 1.5:

```
create table jsample2 (
ISBN char(15),
Lng char(2) field_format='LANG',
json_Author char(255) field_format='AUTHOR:*',
Title char(32) field_format='TITLE',
Year int(4) field_format='DATEPUB')
engine=CONNECT table_type=JSON file_name='biblio3.json';
```

Now the query:

```
select json_Author from jsample2;
```

will return and display :

| json_Author                                                                                         |
|-----------------------------------------------------------------------------------------------------|
| [{"FIRSTNAME":"Jean-Christophe","LASTNAME":"Bernadac"}, {"FIRSTNAME":"François","LASTNAME":"Knab"}] |
| [{"FIRSTNAME":"William J.","LASTNAME":"Pardi"}]                                                     |

Note: Prefixing the column name by `json_` is optional but is useful when using the column as argument to Connect UDF functions, making it to be surely recognized as valid Json without aliasing.

This also works on input, a column specified so that it can be directly set to a valid JSON string.

This feature is of great value as we will see below.

## Create, Read, Update and Delete Operations on JSON Tables

The SQL commands INSERT, UPDATE and DELETE are fully supported for JSON tables except those returned by REST queries. For INSERT and UPDATE, if the target values are simple values, there are no problems.

However, there are some issues when the added or modified values are objects or arrays.

Concerning objects, the same problems exist that we have already seen with the XML type. The added or modified object will have the format described in the table definition, which can be different from the one of the JSON file. Modifications should be done using a file specifying the full path of modified objects.

New problems are raised when trying to modify the values of an array. Only updates can be done on the original table. First of all, for the values of the array to be distinct values, all update operations concerning array values must be done using a table expanding this array.

For instance, to modify the authors of the

```
biblio.json  
based table, the  
jsampex
```

table must be used. Doing so, updating and deleting authors is possible using standard SQL commands. For example, to change the first name of Knab from François to John:

```
update jsampex set authorfn = 'John' where authorln = 'Knab';
```

However It would be wrong to do:

```
update jsampex set authorfn = 'John' where isbn = '9782212090819';
```

Because this would change the first name of both authors as they share the same ISBN.

Where things become more difficult is when trying to delete or insert an author of a book. Indeed, a delete command will delete the whole book and an insert command will add a new complete row instead of adding a new author in the same array. Here we are penalized by the SQL language that cannot give us a way to specify this. Something like:

```
update jsampex add authorfn = 'Charles', authorln = 'Dickens'  
where title = 'XML en Action';
```

However this does not exist in SQL. Does this mean that it is impossible to do it? No, but it requires us to use a table specified on the same file but adapted to this task. One way to do it is to specify a table for which the authors are no more an expanded array. Supposing we want to add an author to the "XML en Action" book. We will do it on a table containing just the author(s) of that book, which is the second book of the table.

From Connect 1.6:

```
create table jaauthor (  
FIRSTNAME char(64),  
LASTNAME char(64))  
engine=CONNECT table_type=JSON File_name='biblio3.json' option_list='Object=1.AUTHOR';
```

Until Connect 1.5

```
create table jaauthor (  
FIRSTNAME char(64),  
LASTNAME char(64))  
engine=CONNECT table_type=JSON File_name='biblio3.json' option_list='Object=[1]:AUTHOR';
```

The command:

```
select * from jaauthor;
```

replies:

| FIRSTNAME  | LASTNAME |
|------------|----------|
| William J. | Pardi    |

It is a standard JSON table that is an array of objects in which we can freely insert or delete rows.

```
insert into jauthor values('Charles','Dickens');
```

We can check that this was done correctly by:

```
select * from jsampex;
```

This will display:

| ISBN          | Title                          | AuthorFN        | AuthorLN | Year |
|---------------|--------------------------------|-----------------|----------|------|
| 9782212090819 | Construire une application XML | Jean-Christophe | Bernadac | 1999 |
| 9782212090819 | Construire une application XML | John            | Knab     | 1999 |
| 9782840825685 | XML en Action                  | William J.      | Pardi    | 1999 |
| 9782840825685 | XML en Action                  | Charles         | Dickens  | 1999 |

Note: If this table were a big table with many books, it would be difficult to know what the order of a specific book is in the table. This can be found by adding a special ROWID column in the table.

However, an alternate way to do it is by using direct JSON column representation as in the JSAMPLE2 table. This can be done by:

```
update jsample2 set json_Author =
'[{"FIRSTNAME":"William J.","LASTNAME":"Pardi"}, {"FIRSTNAME":"Charles","LASTNAME":"Dickens"}]'
where isbn = '9782840825685';
```

Here, we didn't have to find the index of the sub array to modify. However, this is not quite satisfying because we had to manually write the whole JSON value to set to the json\_Author column.

Therefore we need specific functions to do so. They are introduced now.

## JSON User Defined Functions

Although such functions written by other parties do exist, [2] CONNECT provides its own UDFs that are specifically adapted to the JSON table type and easily available because, being inside the CONNECT library or DLL, they require no additional module to be loaded (see [CONNECT - Compiling JSON UDFs in a Separate Library](#) to make these functions in a separate library module).

In particular, [MariaDB 10.2](#) and 10.3 feature native JSON functions. In some cases, it is possible that these native functions can be used. However, mixing native and UDF JSON functions in the same query often does not work because the way they recognize their arguments is different and might even cause a server crash.

Here is the list of the CONNECT functions; more can be added if required.

| Name                  | Type     | Return  | Description                                               | Added                          |
|-----------------------|----------|---------|-----------------------------------------------------------|--------------------------------|
| jbin_array            | Function | STRING* | Make a JSON array containing its arguments.               | <a href="#">MariaDB 10.1.9</a> |
| jbin_array_add        | Function | STRING* | Adds to its first array argument its second arguments.    | <a href="#">MariaDB 10.1.9</a> |
| jbin_array_add_values | Function | STRING* | Adds to its first array argument all following arguments. |                                |
| jbin_array_delete     | Function | STRING* | Deletes the nth element of its first array argument.      | <a href="#">MariaDB 10.1.9</a> |
| jbin_file             | Function | STRING* | Returns of a (json) file contain.                         | <a href="#">MariaDB 10.1.9</a> |
| jbin_get_item         | Function | STRING* | Access and returns a json item by a JPATH key.            | <a href="#">MariaDB 10.1.9</a> |
| jbin_insert_item      | Function | STRING  | Insert item values located to paths.                      |                                |
| jbin_item_merge       | Function | STRING* | Merges two arrays or two objects.                         | <a href="#">MariaDB 10.1.9</a> |
| jbin_object           | Function | STRING* | Make a JSON object containing its arguments.              | <a href="#">MariaDB 10.1.9</a> |
| jbin_object_nonull    | Function | STRING* | Make a JSON object containing its not null arguments.     | <a href="#">MariaDB 10.1.9</a> |
| jbin_object_add       | Function | STRING* | Adds to its first object argument its second argument.    | <a href="#">MariaDB 10.1.9</a> |
| jbin_object_delete    | Function | STRING* | Deletes the nth element of its first object argument.     | <a href="#">MariaDB 10.1.9</a> |
| jbin_object_key       | Function | STRING* | Make a JSON object for key/value pairs.                   |                                |
| jbin_object_list      | Function | STRING* | Returns the list of object keys as an array.              | <a href="#">MariaDB 10.1.9</a> |
| jbin_set_item         | Function | STRING  | Set item values located to paths.                         |                                |

|                       |           |         |                                                                                                          |                                                                      |
|-----------------------|-----------|---------|----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| jbin_update_item      | Function  | STRING  | Update item values located to paths.                                                                     |                                                                      |
| jfile_json            | Function  | STRING  | Convert a pretty=0 file to another BJson file.                                                           | MariaDB 10.5.9 , MariaDB 10.4.18 , MariaDB 10.3.28 , MariaDB 10.2.36 |
| jfile_convert         | Function  | STRING  | Convert a Json file to another pretty=0 file.                                                            | MariaDB 10.5.9 , MariaDB 10.4.18 , MariaDB 10.3.28 , MariaDB 10.2.36 |
| jfile_make            | Function  | STRING  | Make a json file from its json item first argument.                                                      | MariaDB 10.1.9                                                       |
| json_array            | Function  | STRING  | Make a JSON array containing its arguments.                                                              | MariaDB 10.0.17 until Connect 1.5                                    |
| json_array_add        | Function  | STRING  | Adds to its first array argument its second arguments (before MariaDB 10.1.9 , all following arguments). |                                                                      |
| json_array_add_values | Function  | STRING  | Adds to its first array argument all following arguments.                                                | MariaDB 10.1.9                                                       |
| json_array_delete     | Function  | STRING  | Deletes the nth element of its first array argument.                                                     |                                                                      |
| json_array_grp        | Aggregate | STRING  | Makes JSON arrays from coming argument.                                                                  |                                                                      |
| json_file             | Function  | STRING  | Returns the contains of (json) file.                                                                     | MariaDB 10.1.9                                                       |
| json_get_item         | Function  | STRING  | Access and returns a json item by a JPATH key.                                                           | MariaDB 10.1.9                                                       |
| json_insert_item      | Function  | STRING  | Insert item values located to paths.                                                                     |                                                                      |
| json_item_merge       | Function  | STRING  | Merges two arrays or two objects.                                                                        | MariaDB 10.1.9                                                       |
| json_locate_all       | Function  | STRING  | Returns the JPATH's of all occurrences of an element.                                                    | MariaDB 10.1.9                                                       |
| json_make_array       | Function  | STRING  | Make a JSON array containing its arguments.                                                              | From Connect 1.6                                                     |
| json_make_object      | Function  | STRING  | Make a JSON object containing its arguments.                                                             | From Connect 1.6                                                     |
| json_object           | Function  | STRING  | Make a JSON object containing its arguments.                                                             | MariaDB 10.0.17 until Connect 1.5                                    |
| json_object_delete    | Function  | STRING  | Deletes the nth element of its first object argument.                                                    | MariaDB 10.1.9                                                       |
| json_object_grp       | Aggregate | STRING  | Makes JSON objects from coming arguments.                                                                |                                                                      |
| json_object_list      | Function  | STRING  | Returns the list of object keys as an array.                                                             | MariaDB 10.1.9                                                       |
| json_object_nonull    | Function  | STRING  | Make a JSON object containing its not null arguments.                                                    |                                                                      |
| json_serialize        | Function  | STRING  | Serializes the return of a "Jbin" function.                                                              | MariaDB 10.1.9                                                       |
| json_set_item         | Function  | STRING  | Set item values located to paths.                                                                        |                                                                      |
| json_update_item      | Function  | STRING  | Update item values located to paths.                                                                     |                                                                      |
| jsonvalue             | Function  | STRING  | Make a JSON value from its unique argument. Called json_value until MariaDB 10.0.22 and MariaDB 10.1.8 . | MariaDB 10.0.17                                                      |
| jsoncontains          | Function  | INTEGER | Returns 0 or 1 if an element is contained in the document.                                               |                                                                      |
| jsoncontains_path     | Function  | INTEGER | Returns 0 or 1 if a JPATH is contained in the document.                                                  |                                                                      |
| jsonget_string        | Function  | STRING  | Access and returns a string element by a JPATH key.                                                      | MariaDB 10.1.9                                                       |
| jsonget_int           | Function  | INTEGER | Access and returns an integer element by a JPATH key.                                                    | MariaDB 10.1.9                                                       |
| jsonget_real          | Function  | REAL    | Access and returns a real element by a JPATH key.                                                        | MariaDB 10.1.9                                                       |
| jsonlocate            | Function  | STRING  | Returns the JPATH to access one element.                                                                 | MariaDB 10.1.9                                                       |

String values are mapped to JSON strings. These strings are automatically escaped to conform to the JSON syntax. The automatic escaping is bypassed when the value has an alias beginning with 'json\_'. This is automatically the case when a JSON UDF argument is another JSON UDF whose name begins with "json\_" (not case sensitive). This is why all functions that do not return a Json item are not prefixed by "json\_".

Argument string values, for some functions, can alternatively be json file names. When this is ambiguous, alias them as *jfile\_*. Full path should be used because UDF functions has no means to know what the current database is. Apparently, when the file name path is not full, it is based on the MariaDB data directory but I am not sure it is always true.

Numeric values are (big) integers, double floating point values or decimal values. Decimal values are character strings containing a numeric representation and are treated as strings. Floating point values contain a decimal point and/or an exponent. Integers are written without decimal points.

To install these functions execute the following commands : [3]

Note: Json function names are often written on this page with leading upper case letters for clarity. It is possible to do so in SQL queries because function names are case insensitive. However, when creating or dropping them, their names must match the case they are in the library module (lower case from MariaDB 10.1.9 ).

On Unix systems (from Connect 1.7.02):

```
create function jsonvalue returns string soname 'ha_connect.so';
create function json_make_array returns string soname 'ha_connect.so';
create function json_array_add_values returns string soname 'ha_connect.so';
create function json_array_add returns string soname 'ha_connect.so';
create function json_array_delete returns string soname 'ha_connect.so';
create function json_make_object returns string soname 'ha_connect.so';
create function json_object_nonull returns string soname 'ha_connect.so';
create function json_object_key returns string soname 'ha_connect.so';
create function json_object_add returns string soname 'ha_connect.so';
create function json_object_delete returns string soname 'ha_connect.so';
create function json_object_list returns string soname 'ha_connect.so';
create function json_object_values returns string soname 'ha_connect.so';
create function jsonset_grp_size returns integer soname 'ha_connect.so';
create function jsonget_grp_size returns integer soname 'ha_connect.so';
create aggregate function json_array_grp returns string soname 'ha_connect.so';
create aggregate function json_object_grp returns string soname 'ha_connect.so';
create function jsonlocate returns string soname 'ha_connect.so';
create function json_locate_all returns string soname 'ha_connect.so';
create function jsoncontains returns integer soname 'ha_connect.so';
create function jsoncontains_path returns integer soname 'ha_connect.so';
create function json_item_merge returns string soname 'ha_connect.so';
create function json_get_item returns string soname 'ha_connect.so';
create function jsonget_string returns string soname 'ha_connect.so';
create function jsonget_int returns integer soname 'ha_connect.so';
create function jsonget_real returns real soname 'ha_connect.so';
create function json_set_item returns string soname 'ha_connect.so';
create function json_insert_item returns string soname 'ha_connect.so';
create function json_update_item returns string soname 'ha_connect.so';
create function json_file returns string soname 'ha_connect.so';
create function jfile_make returns string soname 'ha_connect.so';
create function jfile_convert returns string soname 'ha_connect.so';
create function jfile_bjson returns string soname 'ha_connect.so';
create function json_serialize returns string soname 'ha_connect.so';
create function jbin_array returns string soname 'ha_connect.so';
create function jbin_array_add_values returns string soname 'ha_connect.so';
create function jbin_array_add returns string soname 'ha_connect.so';
create function jbin_array_delete returns string soname 'ha_connect.so';
create function jbin_object returns string soname 'ha_connect.so';
create function jbin_object_nonull returns string soname 'ha_connect.so';
create function jbin_object_key returns string soname 'ha_connect.so';
create function jbin_object_add returns string soname 'ha_connect.so';
create function jbin_object_delete returns string soname 'ha_connect.so';
create function jbin_object_list returns string soname 'ha_connect.so';
create function jbin_item_merge returns string soname 'ha_connect.so';
create function jbin_get_item returns string soname 'ha_connect.so';
create function jbin_set_item returns string soname 'ha_connect.so';
create function jbin_insert_item returns string soname 'ha_connect.so';
create function jbin_update_item returns string soname 'ha_connect.so';
create function jbin_file returns string soname 'ha_connect.so';
```

On Unix systems (from Connect 1.6):

```
create function jsonvalue returns string soname 'ha_connect.so';
create function json_make_array returns string soname 'ha_connect.so';
create function json_array_add_values returns string soname 'ha_connect.so';
create function json_array_add returns string soname 'ha_connect.so';
create function json_array_delete returns string soname 'ha_connect.so';
create function json_make_object returns string soname 'ha_connect.so';
create function json_object_nonull returns string soname 'ha_connect.so';
create function json_object_key returns string soname 'ha_connect.so';
create function json_object_add returns string soname 'ha_connect.so';
create function json_object_delete returns string soname 'ha_connect.so';
create function json_object_list returns string soname 'ha_connect.so';
create function jsonset_grp_size returns integer soname 'ha_connect.so';
create function jsonget_grp_size returns integer soname 'ha_connect.so';
create aggregate function json_array_grp returns string soname 'ha_connect.so';
create aggregate function json_object_grp returns string soname 'ha_connect.so';
create function jsonlocate returns string soname 'ha_connect.so';
create function json_locate_all returns string soname 'ha_connect.so';
create function jsoncontains returns integer soname 'ha_connect.so';
create function jsoncontains_path returns integer soname 'ha_connect.so';
create function json_item_merge returns string soname 'ha_connect.so';
create function json_get_item returns string soname 'ha_connect.so';
create function jsonget_string returns string soname 'ha_connect.so';
create function jsonget_int returns integer soname 'ha_connect.so';
create function jsonget_real returns real soname 'ha_connect.so';
create function json_set_item returns string soname 'ha_connect.so';
create function json_insert_item returns string soname 'ha_connect.so';
create function json_update_item returns string soname 'ha_connect.so';
create function json_file returns string soname 'ha_connect.so';
create function jfile_make returns string soname 'ha_connect.so';
create function json_serialize returns string soname 'ha_connect.so';
create function jbin_array returns string soname 'ha_connect.so';
create function jbin_array_add_values returns string soname 'ha_connect.so';
create function jbin_array_add returns string soname 'ha_connect.so';
create function jbin_array_delete returns string soname 'ha_connect.so';
create function jbin_object returns string soname 'ha_connect.so';
create function jbin_object_nonull returns string soname 'ha_connect.so';
create function jbin_object_key returns string soname 'ha_connect.so';
create function jbin_object_add returns string soname 'ha_connect.so';
create function jbin_object_delete returns string soname 'ha_connect.so';
create function jbin_object_list returns string soname 'ha_connect.so';
create function jbin_item_merge returns string soname 'ha_connect.so';
create function jbin_get_item returns string soname 'ha_connect.so';
create function jbin_set_item returns string soname 'ha_connect.so';
create function jbin_insert_item returns string soname 'ha_connect.so';
create function jbin_update_item returns string soname 'ha_connect.so';
create function jbin_file returns string soname 'ha_connect.so';
```

On Unix systems (from MariaDB 10.1.9 until Connect 1.5):

```
create function jsonvalue returns string soname 'ha_connect.so';
create function json_array returns string soname 'ha_connect.so';
create function json_array_add_values returns string soname 'ha_connect.so';
create function json_array_add returns string soname 'ha_connect.so';
create function json_array_delete returns string soname 'ha_connect.so';
create function json_object returns string soname 'ha_connect.so';
create function json_object_nonull returns string soname 'ha_connect.so';
create function json_object_key returns string soname 'ha_connect.so';
create function json_object_add returns string soname 'ha_connect.so';
create function json_object_delete returns string soname 'ha_connect.so';
create function json_object_list returns string soname 'ha_connect.so';
create function jsonset_grp_size returns integer soname 'ha_connect.so';
create function jsonget_grp_size returns integer soname 'ha_connect.so';
create aggregate function json_array_grp returns string soname 'ha_connect.so';
create aggregate function json_object_grp returns string soname 'ha_connect.so';
create function jsonlocate returns string soname 'ha_connect.so';
create function json_locate_all returns string soname 'ha_connect.so';
create function jsoncontains returns integer soname 'ha_connect.so';
create function jsoncontains_path returns integer soname 'ha_connect.so';
create function json_item_merge returns string soname 'ha_connect.so';
create function json_get_item returns string soname 'ha_connect.so';
create function jsonget_string returns string soname 'ha_connect.so';
create function jsonget_int returns integer soname 'ha_connect.so';
create function jsonget_real returns real soname 'ha_connect.so';
create function json_set_item returns string soname 'ha_connect.so';
create function json_insert_item returns string soname 'ha_connect.so';
create function json_update_item returns string soname 'ha_connect.so';
create function json_file returns string soname 'ha_connect.so';
create function jfile_make returns string soname 'ha_connect.so';
create function json_serialize returns string soname 'ha_connect.so';
create function jbin_array returns string soname 'ha_connect.so';
create function jbin_array_add_values returns string soname 'ha_connect.so';
create function jbin_array_add returns string soname 'ha_connect.so';
create function jbin_array_delete returns string soname 'ha_connect.so';
create function jbin_object returns string soname 'ha_connect.so';
create function jbin_object_nonull returns string soname 'ha_connect.so';
create function jbin_object_key returns string soname 'ha_connect.so';
create function jbin_object_add returns string soname 'ha_connect.so';
create function jbin_object_delete returns string soname 'ha_connect.so';
create function jbin_object_list returns string soname 'ha_connect.so';
create function jbin_item_merge returns string soname 'ha_connect.so';
create function jbin_get_item returns string soname 'ha_connect.so';
create function jbin_set_item returns string soname 'ha_connect.so';
create function jbin_insert_item returns string soname 'ha_connect.so';
create function jbin_update_item returns string soname 'ha_connect.so';
create function jbin_file returns string soname 'ha_connect.so';
```

On WIndows (from Connect 1.7.02):

```
create function jsonvalue returns string soname 'ha_connect';
create function json_make_array returns string soname 'ha_connect';
create function json_array_add_values returns string soname 'ha_connect';
create function json_array_add returns string soname 'ha_connect';
create function json_array_delete returns string soname 'ha_connect';
create function json_make_object returns string soname 'ha_connect';
create function json_object_nonull returns string soname 'ha_connect';
create function json_object_key returns string soname 'ha_connect';
create function json_object_add returns string soname 'ha_connect';
create function json_object_delete returns string soname 'ha_connect';
create function json_object_list returns string soname 'ha_connect';
create function json_object_values returns string soname 'ha_connect';
create function jsonset_grp_size returns integer soname 'ha_connect';
create function jsonget_grp_size returns integer soname 'ha_connect';
create aggregate function json_array_grp returns string soname 'ha_connect';
create aggregate function json_object_grp returns string soname 'ha_connect';
create function jsonlocate returns string soname 'ha_connect';
create function json_locate_all returns string soname 'ha_connect';
create function jsoncontains returns integer soname 'ha_connect';
create function jsoncontains_path returns integer soname 'ha_connect';
create function json_item_merge returns string soname 'ha_connect';
create function json_get_item returns string soname 'ha_connect';
create function jsonget_string returns string soname 'ha_connect';
create function jsonget_int returns integer soname 'ha_connect';
create function jsonget_real returns real soname 'ha_connect';
create function json_set_item returns string soname 'ha_connect';
create function json_insert_item returns string soname 'ha_connect';
create function json_update_item returns string soname 'ha_connect';
create function json_file returns string soname 'ha_connect';
create function jfile_make returns string soname 'ha_connect';
create function jfile_convert returns string soname 'ha_connect';
create function jfile_bjson returns string soname 'ha_connect';
create function json_serialize returns string soname 'ha_connect';
create function jbin_array returns string soname 'ha_connect';
create function jbin_array_add_values returns string soname 'ha_connect';
create function jbin_array_add returns string soname 'ha_connect';
create function jbin_array_delete returns string soname 'ha_connect';
create function jbin_object returns string soname 'ha_connect';
create function jbin_object_nonull returns string soname 'ha_connect';
create function jbin_object_key returns string soname 'ha_connect';
create function jbin_object_add returns string soname 'ha_connect';
create function jbin_object_delete returns string soname 'ha_connect';
create function jbin_object_list returns string soname 'ha_connect';
create function jbin_item_merge returns string soname 'ha_connect';
create function jbin_get_item returns string soname 'ha_connect';
create function jbin_set_item returns string soname 'ha_connect';
create function jbin_insert_item returns string soname 'ha_connect';
create function jbin_update_item returns string soname 'ha_connect';
create function jbin_file returns string soname 'ha_connect';
```

On WIndows (from Connect 1.6):

```
create function jsonvalue returns string soname 'ha_connect';
create function json_make_array returns string soname 'ha_connect';
create function json_array_add_values returns string soname 'ha_connect';
create function json_array_add returns string soname 'ha_connect';
create function json_array_delete returns string soname 'ha_connect';
create function json_make_object returns string soname 'ha_connect';
create function json_object_nonull returns string soname 'ha_connect';
create function json_object_key returns string soname 'ha_connect';
create function json_object_add returns string soname 'ha_connect';
create function json_object_delete returns string soname 'ha_connect';
create function json_object_list returns string soname 'ha_connect';
create function jsonset_grp_size returns integer soname 'ha_connect';
create function jsonget_grp_size returns integer soname 'ha_connect';
create aggregate function json_array_grp returns string soname 'ha_connect';
create aggregate function json_object_grp returns string soname 'ha_connect';
create function jsonlocate returns string soname 'ha_connect';
create function json_locate_all returns string soname 'ha_connect';
create function jsoncontains returns integer soname 'ha_connect';
create function jsoncontains_path returns integer soname 'ha_connect';
create function json_item_merge returns string soname 'ha_connect';
create function json_get_item returns string soname 'ha_connect';
create function jsonget_string returns string soname 'ha_connect';
create function jsonget_int returns integer soname 'ha_connect';
create function jsonget_real returns real soname 'ha_connect';
create function json_set_item returns string soname 'ha_connect';
create function json_insert_item returns string soname 'ha_connect';
create function json_update_item returns string soname 'ha_connect';
create function json_file returns string soname 'ha_connect';
create function jfile_make returns string soname 'ha_connect';
create function json_serialize returns string soname 'ha_connect';
create function jbin_array returns string soname 'ha_connect';
create function jbin_array_add_values returns string soname 'ha_connect';
create function jbin_array_add returns string soname 'ha_connect';
create function jbin_array_delete returns string soname 'ha_connect';
create function jbin_object returns string soname 'ha_connect';
create function jbin_object_nonull returns string soname 'ha_connect';
create function jbin_object_key returns string soname 'ha_connect';
create function jbin_object_add returns string soname 'ha_connect';
create function jbin_object_delete returns string soname 'ha_connect';
create function jbin_object_list returns string soname 'ha_connect';
create function jbin_item_merge returns string soname 'ha_connect';
create function jbin_get_item returns string soname 'ha_connect';
create function jbin_set_item returns string soname 'ha_connect';
create function jbin_insert_item returns string soname 'ha_connect';
create function jbin_update_item returns string soname 'ha_connect';
create function jbin_file returns string soname 'ha_connect';
```

On WIndows (until Connect 1.5):

```

create function jsonvalue returns string soname 'ha_connect';
create function json_array returns string soname 'ha_connect';
create function json_array_add_values returns string soname 'ha_connect';
create function json_array_add returns string soname 'ha_connect';
create function json_array_delete returns string soname 'ha_connect';
create function json_object returns string soname 'ha_connect';
create function json_object_nonull returns string soname 'ha_connect';
create function json_object_key returns string soname 'ha_connect';
create function json_object_add returns string soname 'ha_connect';
create function json_object_delete returns string soname 'ha_connect';
create function json_object_list returns string soname 'ha_connect';
create function jsonset_grp_size returns integer soname 'ha_connect';
create function jsonget_grp_size returns integer soname 'ha_connect';
create aggregate function json_array_grp returns string soname 'ha_connect';
create aggregate function json_object_grp returns string soname 'ha_connect';
create function jsonlocate returns string soname 'ha_connect';
create function json_locate_all returns string soname 'ha_connect';
create function jsoncontains returns integer soname 'ha_connect';
create function jsoncontains_path returns integer soname 'ha_connect';
create function json_item_merge returns string soname 'ha_connect';
create function json_get_item returns string soname 'ha_connect';
create function jsonget_string returns string soname 'ha_connect';
create function jsonget_int returns integer soname 'ha_connect';
create function jsonget_real returns real soname 'ha_connect';
create function json_set_item returns string soname 'ha_connect';
create function json_insert_item returns string soname 'ha_connect';
create function json_update_item returns string soname 'ha_connect';
create function json_file returns string soname 'ha_connect';
create function jfile_make returns string soname 'ha_connect';
create function json_serialize returns string soname 'ha_connect';
create function jbin_array returns string soname 'ha_connect';
create function jbin_array_add_values returns string soname 'ha_connect';
create function jbin_array_add returns string soname 'ha_connect';
create function jbin_array_delete returns string soname 'ha_connect';
create function jbin_object returns string soname 'ha_connect';
create function jbin_object_nonull returns string soname 'ha_connect';
create function jbin_object_key returns string soname 'ha_connect';
create function jbin_object_add returns string soname 'ha_connect';
create function jbin_object_delete returns string soname 'ha_connect';
create function jbin_object_list returns string soname 'ha_connect';
create function jbin_item_merge returns string soname 'ha_connect';
create function jbin_get_item returns string soname 'ha_connect';
create function jbin_set_item returns string soname 'ha_connect';
create function jbin_insert_item returns string soname 'ha_connect';
create function jbin_update_item returns string soname 'ha_connect';
create function jbin_file returns string soname 'ha_connect';

```

## Jfile\_Bjson

MariaDB starting with 10.2.36

JFile\_Bjson was introduced in [MariaDB 10.5.9](#) , [MariaDB 10.4.18](#) , [MariaDB 10.3.28](#) and [MariaDB 10.2.36](#) .

```
Jfile_Bjson(in_file_name, out_file_name, lrecl)
```

Converts the first argument pretty=0 json file to Bjson file. B(binary)json is a pre-parsed json format. It is described below in the Performance chapter (available in next Connect versions).

## Jfile\_Convert

MariaDB starting with 10.2.36

JFile\_Convert was introduced in [MariaDB 10.5.9](#) , [MariaDB 10.4.18](#) , [MariaDB 10.3.28](#) and [MariaDB 10.2.36](#) .

```
Jfile_Convert(in_file_name, out_file_name, lrecl)
```

Converts the first argument json file to another *pretty=0* json file. The third integer argument is the record length to use. This is often required to process huge json files that would be very slow if they were in *pretty=2* format.

This is done without completely parsing the file, is very fast and requires no big memory.

## Jfile\_Make

MariaDB starting with [10.1.9](#)

Jfile\_Make was added in CONNECT 1.4 (from [MariaDB 10.1.9](#) ).

```
Jfile_Make(arg1, arg2, [arg3], ...)
```

The first argument must be a json item (if it is just a string, Jfile\_Make will try its best to see if it is a json item or an input file name). The following arguments are a string file name and an integer pretty value (defaulting to 2) in any order. This function creates a json file containing the first argument item.

The returned string value is the created file name. If not specified as an argument, the file name can in some cases be retrieved from the first argument; in such cases the file itself is modified.

This function can be used to create or format a json file. For instance, supposing we want to format the file tb.json, this can be done with the query:

```
select Jfile_Make('tb.json' jfile_, 2);
```

The tb.json file will be changed to:

```
[  
  {  
    "_id": 5,  
    "type": "food",  
    "ratings": [  
      5,  
      8,  
      9  
    ]  
  },  
  {  
    "_id": 6,  
    "type": "car",  
    "ratings": [  
      5,  
      9  
    ]  
  }  
]
```

## Json\_Array\_Add

```
Json_Array_Add(arg1, arg2, [arg3][, arg4][, ...])
```

Note: In CONNECT version 1.3 (before [MariaDB 10.1.9](#) ), this function behaved like the new

`Json_Array_Add_Values`

function. The following describes this function for CONNECT version 1.4 (from [MariaDB 10.1.9](#) ) only. The first argument must be a JSON array. The second argument is added as member of this array. For example:

```
select Json_Array_Add(Json_Array(56,3.1416,'machin',NULL),  
'One more') Array;
```

Array

```
[56,3.141600,"machin",null,"One more"]
```

Note: The first array is not escaped, its (alias) name beginning with 'json\_.'

Now we can see how adding an author to the JSAMPLE2 table can alternatively be done:

```
update jsample2 set  
  json_author = json_array_add(json_author, json_object('Charles' FIRSTNAME, 'Dickens' LASTNAME))  
  where isbn = '9782840825685';
```

Note: Calling a column returning JSON a name prefixed by `json_` (like `json_author` here) is good practice and removes the need to give it an alias to prevent escaping when used as an argument.

Additional arguments: If a third integer argument is given, it specifies the position (zero based) of the added value:

```
select Json_Array_Add('[5,3,8,7,9]' json_, 4, 2) Array;
```

## Array

```
[5,3,4,8,7,9]
```

If a string argument is added, it specifies the Json path to the array to be modified. For instance:

```
select Json_Array_Add('{"a":1,"b":2,"c":[3,4]}' json_, 5, 1, 'c');
```

## Json\_Array\_Add('{"a":1,"b":2,"c":[3,4]}' json\_, 5, 1, 'c')

```
{"a":1,"b":2,"c":[3,5,4]}
```

## Json\_Array\_Add\_Values

Json\_Array\_Add\_Values added in CONNECT 1.4 replaces the function Json\_Array\_Add of CONNECT version 1.3 (before MariaDB 10.1.9 ).

```
Json_Array_Add_Values(arg, arglist)
```

The first argument must be a JSON array string. Then all other arguments are added as members of this array. For example:

```
select Json_Array_Add_Values  
(Json_Array(56, 3.1416, 'machin', NULL), 'One more', 'Two more') Array;
```

## Array

```
[56,3.141600,"machin",null,"One more","Two more"]
```

## Json\_Array\_Delete

```
Json_Array_Delete(arg1, arg2 [,arg3] [...])
```

The first argument should be a JSON array. The second argument is an integer indicating the rank (0 based conforming to general json usage) of the element to delete. For example:

```
select Json_Array_Delete(Json_Array(56,3.1416,'foo',NULL),1) Array;
```

## Array

```
[56,"foo",null]
```

Now we can see how to delete the second author from the JSAMPLE2 table:

```
update jsample2 set json_author = json_array_delete(json_author, 1)  
where isbn = '9782840825685';
```

A Json path can be specified as a third string argument

## Json\_Array\_Grp

```
Json_Array_Grp(arg)
```

This is an aggregate function that makes an array filled from values coming from the rows retrieved by a query. Let us suppose we have the pet table:

| name    | race   | number |
|---------|--------|--------|
| John    | dog    | 2      |
| Bill    | cat    | 1      |
| Mary    | dog    | 1      |
| Mary    | cat    | 1      |
| Lisbeth | rabbit | 2      |
| Kevin   | cat    | 2      |

|        |      |   |
|--------|------|---|
| Kevin  | bird | 6 |
| Donald | dog  | 1 |
| Donald | fish | 3 |

The query:

```
select name, json_array_grp(race) from pet group by name;
```

will return:

| name    | json_array_grp(race) |
|---------|----------------------|
| Bill    | ["cat"]              |
| Donald  | ["dog", "fish"]      |
| John    | ["dog"]              |
| Kevin   | ["cat", "bird"]      |
| Lisbeth | ["rabbit"]           |
| Mary    | ["dog", "cat"]       |

One problem with the JSON aggregate functions is that they construct their result in memory and cannot know the needed amount of storage, not knowing the number of rows of the used table.

Therefore, the number of values for each group is limited. This limit is the value of JsonGrpSize whose default value is 10 but can be set using the JsonSet\_Grp\_Size function. Nevertheless, working on a larger table is possible, but only after setting JsonGrpSize to the ceiling of the number of rows per group for the table. Try not to set it to a very large value to avoid memory exhaustion.

## JsonContains

```
JsonContains(json_doc, item [, int])<
```

This function can be used to check whether an item is contained in a document. Its arguments are the same than the ones of the JsonLocate function; only the return value changes. The integer returned value is 1 if the item is contained in the document or 0 otherwise.

## JsonContains\_Path

```
JsonContains_Path(json_doc, path)
```

This function can be used to check whether a Json path is contained in the document. The integer returned value is 1 if the path is contained in the document or 0 otherwise.

## Json\_File

```
Json_File(arg1, [arg2, [arg3]], ...)
```

The first argument must be a file name. This function returns the text of the file that is supposed to be a json file. If only one argument is specified, the file text is returned without being parsed. Up to two additional arguments can be specified:

A string argument is the path to the sub-item to be returned. An integer argument specifies the pretty format value of the file.

This function is chiefly used to get the json item argument of other json functions from a json file. For instance, supposing the file tb.json is:

```
{ "_id" : 5, "type" : "food", "ratings" : [ 5, 8, 9 ] }
{ "_id" : 6, "type" : "car", "ratings" : [ 5, 9 ] }
```

Extracting a value from it can be done with a query such as:

```
select JsonGet_String(Json_File('tb.json', 0), '[1]:type') "Type";
```

or, from MariaDB 10.2.8 :

```
select JsonGet_String(Json_File('tb.json', 0), '$[1].type') "Type";
```

This query returns:

**Type**

car

However, we'll see that, most of the time, it is better to use Jbin\_File or to directly specify the file name in queries. In particular this function should not be used for queries that must modify the json item because, even if the modified json is returned, the file itself would be unchanged.

## Json\_Get\_Item

MariaDB starting with [10.1.9](#)

Json\_Get\_Item was added in CONNECT 1.4 (from [MariaDB 10.1.9](#) ).

```
Json_Get_Item(arg1, arg2, ...)
```

This function returns a subset of the json document passed as first argument. The second argument is the json path of the item to be returned and should be one returning a json item (terminated by a '\*'). If not, the function will try to make it right but this is not foolproof. For instance:

```
select Json_Get_Item(Json_Object('foo' as "first", Json_Array('a', 33)
                                as "json_second"), 'second') as "item";
```

The correct path should have been 'second:'\* (or from [MariaDB 10.2.8](#) , 'second.\*'), but in this simple case the function was able to make it right. The returned item:

**item**

["a",33]

Note: The array is aliased "json\_second" to indicate it is a json item and avoid escaping it. However, the "json\_" prefix is skipped when making the object and must not be added to the path.

## JsonGet\_Grp\_Size

```
JsonGet_Grp_Size(val)
```

This function returns the JsonGrpSize value.

## JsonGet\_String / JsonGet\_Int / JsonGet\_Real

MariaDB starting with [10.1.9](#)

JsonGet\_String, JsonGet\_Int and JsonGet\_Real were added in CONNECT 1.4 (from [MariaDB 10.1.9](#) ).

```
JsonGet_String(arg1, arg2, [arg3] ...)
JsonGet_Int(arg1, arg2, [arg3] ...)
JsonGet_Real(arg1, arg2, [arg3] ...)
```

The first argument should be a JSON item. If it is a string with no alias, it will be converted as a json item. The second argument is the path of the item to be located in the first argument and returned, eventually converted according to the used function. For example:

```
select
JsonGet_String('{"qty":7,"price":29.50,"garanty":null}', 'price') "String",
JsonGet_Int('{"qty":7,"price":29.50,"garanty":null}', 'price') "Int",
JsonGet_Real('{"qty":7,"price":29.50,"garanty":null}', 'price') "Real";
```

This query returns:

| String | Int | Real               |
|--------|-----|--------------------|
| 29.50  | 29  | 29.500000000000000 |

The function `JsonGet_Real` can be given a third argument to specify the number of decimal digits of the returned value. For instance:

```
select
JsonGet_Real('{"qty":7,"price":29.50,"garanty":null}', 'price', 4) "Real";
```

This query returns:

## String

29.50

The given path can specify all operators for arrays except the "expand" [X] operator (or from [MariaDB 10.2.8](#), the "expand" [\*] operator). For instance:

```
select
JsonGet_Int(Json_Array(45,28,36,45,89), '[4]') "Rank",
JsonGet_Int(Json_Array(45,28,36,45,89), '[#]') "Number",
JsonGet_String(Json_Array(45,28,36,45,89), '[""]') "Concat",
JsonGet_Int(Json_Array(45,28,36,45,89), '[+]') "Sum",
JsonGet_Real(Json_Array(45,28,36,45,89), '[!]', 2) "Avg";
```

The result:

| Rank | Number | Concat         | Sum | Avg   |
|------|--------|----------------|-----|-------|
| 89   | 5      | 45,28,36,45,89 | 243 | 48.60 |

## Json\_Item\_Merge

```
Json_Item_Merge(arg1, arg2, ...)
```

This function merges two arrays or two objects. For arrays, this is done by adding to the first array all the values of the second array. For instance:

```
select Json_Item_Merge(Json_Array('a','b','c'), Json_Array('d','e','f')) as "Result";
```

The function returns:

| Result                    |
|---------------------------|
| ["a","b","c","d","e","f"] |

For objects, the pairs of the second object are added to the first object if the key does not yet exist in it; otherwise the pair of the first object is set with the value of the matching pair of the second object. For instance:

```
select Json_Item_Merge(Json_Object(1 "a", 2 "b", 3 "c"), Json_Object(4 "d",5 "b",6 "f"))
as "Result";
```

The function returns:

| Result                          |
|---------------------------------|
| {"a":1,"b":5,"c":3,"d":4,"f":6} |

## JsonLocate

```
JsonLocate(arg1, arg2, [arg3], ...):
```

The first argument must be a JSON tree. The second argument is the item to be located. The item to be located can be a constant or a json item. Constant values must be equal in type and value to be found. This is "shallow equality" – strings, integers and doubles won't match.

This function returns the json path to the located item or null if it is not found. For example:

```
select JsonLocate('{"AUTHORS":[{"FN":"Jules", "LN":"Verne"}, {"FN":"Jack", "LN":"London"}]}' json_, 'Jack') Path;
```

This query returns:

| Path           |
|----------------|
| AUTHORS:[1]:FN |

or, from [MariaDB 10.2.8](#):

| Path             |
|------------------|
| \$.AUTHORS[1].FN |

The path syntax is the same used in JSON CONNECT tables.

By default, the path of the first occurrence of the item is returned. The third parameter can be used to specify the occurrence whose path is to be returned.

For instance:

```
select
JsonLocate('[[45,28],[36,45],89]',45) first,
JsonLocate('[[45,28],[36,45],89]',45,2) second,
JsonLocate('[[45,28],[36,45],89]',45,0) `wrong type`,
JsonLocate('[[45,28],[36,45],89]','[36,45]' json_) json;
```

| first | second  | wrong type | json |
|-------|---------|------------|------|
| [0]   | [2]:[1] | <null>     | [2]  |

or, from [MariaDB 10.2.8](#):

| first | second   | wrong type | json  |
|-------|----------|------------|-------|
| [\$0] | [\$2][1] | <null>     | [\$2] |

For string items, the comparison is case sensitive by default. However, it is possible to specify a string to be compared case insensitively by giving it an alias beginning by "ci":

```
select JsonLocate('{"AUTHORS":[{"FN":"Jules", "LN":"Verne"}, {"FN":"Jack", "LN":"London"}]}', json_, 'VERNE' ci) Path;
```

| Path           |
|----------------|
| AUTHORS:[0]:LN |

or, from [MariaDB 10.2.8](#):

| Path             |
|------------------|
| \$.AUTHORS[0].LN |

## Json\_Locate\_All

```
Json_Locate_All(arg1, arg2, [arg3], ...):
```

The first argument must be a JSON item. The second argument is the item to be located. This function returns the paths to all locations of the item as an array of strings. For example:

```
select Json_Locate_All('[[45,28],[[36,45],89]]',45);
```

This query returns:

| All paths                |
|--------------------------|
| "[0]:[0]", "[1]:[0]:[1]" |

or, from [MariaDB 10.2.8](#):

| All paths                 |
|---------------------------|
| "\$[0][0]", "\$[1][0][1]" |

The returned array can be applied other functions. For instance, to get the number of occurrences of an item in a json tree, you can do:

```
select JsonGet_Int(Json_Locate_All('[[45,28],[[36,45],89]]',45), '[#]') "Nb of occurs";
```

or, from [MariaDB 10.2.8](#):

```
select JsonGet_Int(Json_Locate_All('[[45,28],[[36,45],89]]',45), '$[#]') "Nb of occurs";
```

The displayed result:

| Nb of occurs |
|--------------|
| 2            |

If specified, the third integer argument set the depth to search in the document. This means the maximum items in the paths (until [MariaDB 10.2.7](#), the number of ':' separator characters in them plus one). This value defaults to 10 but can be increased for complex documents or reduced to set the maximum wanted depth of the returned paths.

## Json\_Make\_Array

```
Json_Make_Array(val1, ..., valn)
```

This function was named “Json\_Array” in previous versions of CONNECT. It was renamed because MariaDB 10.2 features native JSON functions including a [Json\\_Array](#) function. The native function does almost the same as the UDF one, but does not accept CONNECT-specific arguments such as the result from JBIN functions.

Json\_Make\_Array returns a string denoting a JSON array with all its arguments as members. For example:

```
select Json_Make_Array(56, 3.1416, 'My name is "Foo"', NULL);
```

```
Json_Make_Array(56, 3.1416, 'My name is "Foo"', NULL)
```

```
[56,3.141600,"My name is \"Foo\"",null]
```

Note: The argument list can be void. If so, a void array is returned.

This function was named “Json\_Array” in previous versions of CONNECT. It was renamed because MariaDB 10.2 features native JSON functions including a “Json\_Array” function. The native function does almost the same as the UDF one but does not accept CONNECT specific arguments such as the result from JBIN functions.

## Json\_Make\_Object

```
Json_Make_Object(arg1, ..., argn)
```

This function was named “Json\_Object” in previous versions of CONNECT. It was renamed because MariaDB 10.2 features native JSON functions including a [Json\\_Object](#) function. The native function does what the UDF Json\_Object\_Key does.

Json\_Make\_Object returns a string denoting a JSON object. For instance:

```
select Json_Make_Object(56, 3.1416, 'machin', NULL);
```

The object is filled with pairs corresponding to the given arguments. The key of each pair is made from the argument (default or specified) alias.

```
Json_Make_Object(56, 3.1416, 'machin', NULL)
```

```
{"56":56,"3.1416":3.141600,"machin":"machin","NULL":null}
```

When needed, it is possible to specify the keys by giving an alias to the arguments:

```
select Json_Make_Object(56 qty, 3.1416 price, 'machin' truc, NULL garanty);
```

```
Json_Make_Object(56 qty, 3.1416 price, 'machin' truc, NULL garanty)
```

```
{"qty":56,"price":3.141600,"truc":"machin","garanty":null}
```

If the alias is prefixed by ‘json\_’ (to prevent escaping) the key name is stripped from that prefix.

This function is chiefly useful when entering values retrieved from a table, the key being by default the column name:

```
select Json_Make_Object(matricule, nom, titre, salaire) from connect.employe where nom = 'PANTIER';
```

```
Json_Make_Object(matricule, nom, titre, salaire)
```

```
{"matricule":40567,"nom":"PANTIER","titre":"DIRECTEUR","salaire":14000.000000}
```

This function was named “Json\_Object” in previous versions of CONNECT. It was renamed because MariaDB 10.2 features native JSON functions including a “Json\_Object” function. The native function does what the UDF Json\_Object\_Key does.

## Json\_Object\_Add

```
Json_Object_Add(arg1, arg2, [arg3] ...)
```

The first argument must be a JSON object. The second argument is added as a pair to this object. For example:

```
select Json_Object_Add
('{"item":"T-shirt","qty":27,"price":24.99}' json_old,'blue' color) newobj;
```

## newobj

```
{"item":"T-shirt","qty":27,"price":24.990000,"color":"blue"}
```

Note: If the specified key already exists in the object, its value is replaced by the new one.

The third string argument is a Json path to the target object.

## Json\_Object\_Delete

```
Json_Object_Delete(arg1, arg2, [arg3] ...):
```

The first argument must be a JSON object. The second argument is the key of the pair to delete. For example:

```
select Json_Object_Delete('{"item":"T-shirt","qty":27,"price":24.99}' json_old, 'qty') newobj;
```

## newobj

```
{"item":"T-shirt","price":24.99}
```

The third string argument is a Json path to the object to be the target of deletion.

## Json\_Object\_Grp

```
Json_Object_Grp(arg1,arg2)
```

This function works like Json\_Array\_Grp. It makes a JSON object filled with value pairs whose keys are passed from its first argument and values are passed from its second argument.

This can be seen with the query:

```
select name, json_object_grp(number,race) from pet group by name;
```

This query returns:

| name    | json_object_grp(number,race) |
|---------|------------------------------|
| Bill    | {"cat":1}                    |
| Donald  | {"dog":1,"fish":3}           |
| John    | {"dog":2}                    |
| Kevin   | {"cat":2,"bird":6}           |
| Lisbeth | {"rabbit":2}                 |
| Mary    | {"dog":1,"cat":1}            |

## Json\_Object\_Key

```
Json_Object_Key([key1, val1 [, ..., keyn, valn]])
```

Return a string denoting a JSON object. For instance:

```
select Json_Object_Key('qty', 56, 'price', 3.1416, 'truc', 'machin', 'garanty', NULL);
```

The object is filled with pairs made from each key/value arguments.

## Json\_Object\_Key('qty', 56, 'price', 3.1416, 'truc', 'machin', 'garanty', NULL)

```
{"qty":56,"price":3.1416,"truc":"machin","garanty":null}
```

## Json\_Object\_List

```
Json_Object_List(arg1, ...):
```

The first argument must be a JSON object. This function returns an array containing the list of all keys existing in the object. For example:

```
select Json_Object_List(Json_Object(56 qty, 3.1416 price, 'machin' truc, NULL garanty))
"Key List";
```

## Key List

```
["qty","price","truc","garanty"]
```

## Json\_Object\_Nonull

```
Json_Object_Nonull(arg1, ..., argn)
```

This function works like [Json\\_Make\\_Object](#) but “null” arguments are ignored and not inserted in the object. Arguments are regarded as “null” if they are JSON null values, void arrays or objects, or arrays or objects containing only null members.

It is mainly used to avoid constructing useless null items when converting tables (see later).

## Json\_Object\_Values

```
Json_Object_Values(json_object)
```

The first argument must be a JSON object. This function returns an array containing the list of all values existing in the object. For example:

```
select Json_Object_Values('{"One":1,"Two":2,"Three":3}') "Value List";
```

## Value List

```
[1,2,3]
```

## JsonSet\_Grp\_Size

```
JsonSet_Grp_Size(val)
```

This function is used to set the JsonGrpSize value. This value is used by the following aggregate functions as a ceiling value of the number of items in each group. It returns the JsonGrpSize value that can be its default value when passed 0 as argument.

## Json\_Set\_Item / Json\_Insert\_Item / Json\_Update\_Item

```
Json_{Set | Insert | Update}_Item(json_doc, [item, path [, val, path ...]])
```

These functions insert or update data in a JSON document and return the result. The value/path pairs are evaluated left to right. The document produced by evaluating one pair becomes the new value against which the next pair is evaluated.

- `Json_Set_Item` replaces existing values and adds non-existing values.
- `Json_Insert_Item` inserts values without replacing existing values.
- `Json_Update_Item` replaces only existing values.

Example:

```
set @j = Json_Array(1, 2, 3, Json_Object_Key('quatre', 4));
select Json_Set_Item(@j, 'foo', '[1]', 5, '[3]:cinq') as "Set",
Json_Insert_Item(@j, 'foo', '[1]', 5, '[3]:cinq') as "Insert",
Json_Update_Item(@j, 'foo', '[1]', 5, '[3]:cinq') as "Update";
```

or, from [MariaDB 10.2.8](#) :

```
set @j = Json_Array(1, 2, 3, Json_Object_Key('quatre', 4));
select Json_Set_Item(@j, 'foo', '$[1]', 5, '$[3].cinq') as "Set",
Json_Insert_Item(@j, 'foo', '$[1]', 5, '$[3].cinq') as "Insert",
Json_Update_Item(@j, 'foo', '$[1]', 5, '$[3].cinq') as "Update";
```

This query returns:

| Set                               | Insert                        | Update                   |
|-----------------------------------|-------------------------------|--------------------------|
| [1,"foo",3,{"quatre":4,"cinq":5}] | [1,2,3,{"quatre":4,"cinq":5}] | [1,"foo",3,{"quatre":4}] |

## JsonValue

```
JsonValue (val)
```

Returns a JSON value as a string, for instance:

```
select JsonValue(3.1416);
```

**JsonValue(3.1416)**

3.141600

Before MariaDB 10.1.9 , this function was called `Json_Value`, but was renamed to avoid clashing with the `JSON_VALUE` function.

## The “JBIN” return type

Almost all functions returning a json string - whose name begins with `Json_` - have a counterpart with a name beginning with `Jbin_`. This is both for performance (speed and memory) as well as for better control of what the functions should do.

This is due to the way CONNECT UDFs work internally. The `Json` functions, when receiving json strings as parameters, parse them and construct a binary tree in memory. They work on this tree and before returning; serialize this tree to return a new json string.

If the json document is large, this can take up a large amount of time and storage space. It is all right when one simple `json` function is called – it must be done anyway – but is a waste of time and memory when `json` functions are used as parameters to other `json` functions.

To avoid multiple serializing and parsing, the `Jbin` functions should be used as parameters to other functions. Indeed, they do not serialize the memory document tree, but return a structure allowing the receiving function to have direct access to the memory tree. This saves the serialize-parse steps otherwise needed to pass the argument and removes the need to reallocate the memory of the binary tree, which by the way is 6 to 7 times the size of the json string. For instance:

```
select Json_Object(Jbin_Array_Add(Jbin_Array('a','b','c'), 'd') as "Jbin_foo") as "Result";
```

This query returns:

**Result**

{"foo":["a","b","c","d"]}

Here the binary json tree allocated by `Jbin_Array` is completed by `Jbin_Array_Add` and `Json_Object` and serialized only once to make the final result string. It would be serialized and parsed two more times if using “`Json`” functions.

Note that `Jbin` results are recognized as such because they are aliased beginning with “`Jbin_`”. This is why in the `Json_Object` function the alias is specified as “`Jbin_foo`”.

What happens if it is not recognized as such? These functions are declared as returning a string and to take care of this, the returned structure begins with a zero-terminated string. For instance:

```
select Jbin_Array('a','b','c');
```

This query replies:

**Jbin\_Array('a','b','c')**

Binary Json array

Note: When testing, the tree returned by a “`Jbin`” function can be seen using the `Json_Serialize` function whose unique parameter must be a “`Jbin`” result. For instance:

```
select Json_Serialize(Jbin_Array('a','b','c'));
```

This query returns:

**Json\_Serialize(Jbin\_Array('a','b','c'))**

["a","b","c"]

Note: For this simple example, this is equivalent to using the `Json_Array` function.

## Using a file as json UDF first argument

We have seen that many json UDFs can have an additional argument not yet described. This is in the case where the json item argument was referring to a file. Then the additional integer argument is the pretty value of the json file. It matters only when the first argument is just a file name (to make the UDF understand this argument is a file name, it should be aliased with a name beginning with `jfile_`) or if the function modifies the file, in which case it will be

rewritten with this pretty format.

The json item is created by extracting the required part from the file. This can be the whole file but more often only some of it. There are two ways to specify the sub-item of the file to be used:

1. Specifying it in the *Json\_File* or *Jbin\_File* arguments.
2. Specifying it in the receiving function (not possible for all functions).

It doesn't make any difference when the *Jbin\_File* is used but it does with *Json\_File*. For instance:

```
select Jfile_Make('{"a":1, "b":[44, 55]}' json_, 'test.json');
select Json_Array_Add(Json_File('test.json', 'b'), 66);
```

The second query returns:

```
Json_Array_Add(Json_File('test.json', 'b'), 66)
[44,55,66]
```

It just returns the – modified -- subset returned by the *Json\_File* function, while the query:

```
select Json_Array_Add(Json_File('test.json'), 66, 'b');
```

returns what was received from *Json\_File* with the modification made on the subset.

```
Json_Array_Add(Json_File('test.json'), 66, 'b')
>{"a":1,"b":[44,55,66]}
```

Note that in both case the *test.json* file is not modified. This is because the *Json\_File* function returns a string representing all or part of the file text but no information about the file name. This is all right to check what would be the effect of the modification to the file.

However, to have the file modified, use the *Jbin\_File* function or directly give the file name. *Jbin\_File* returns a structure containing the file name, a pointer to the file parsed tree and eventually a pointer to the subset when a path is given as a second argument:

```
select Json_Array_Add(Jbin_File('test.json', 'b'), 66);
```

This query returns:

```
Json_Array_Add(Jbin_File('test.json', 'b'), 66)
test.json
```

This time the file is modified. This can be checked with:

```
select Json_File('test.json', 3);
```

```
Json_File('test.json', 3)
>{"a":1,"b":[44,55,66]}
```

The reason why the first argument is returned by such a query is because of tables such as:

```
create table tb (
n int key,
jfile_cols char(10) not null;
insert into tb values(1,'test.json');
```

In this table, the *jfile\_cols* column just contains a file name. If we update it by:

```
update tb set jfile_cols = select Json_Array_Add(Jbin_File('test.json', 'b'), 66)
where n = 1;
```

This is the *test.json* file that must be modified, not the *jfile\_cols* column. This can be checked by:

```
select JsonGet_String(jfile_cols, '[1]:*') from tb;
```

```
JsonGet_String(jfile_cols, '[1]:*')
>{"a":1,"b":[44,55,66]}
```

Note: It was an important facility to name the second column of the table beginning by “*jfile\_*” so the json functions knew it was a file name without obliging to specify an alias in the queries.

## Using “Jbin” to control what the query execution does

This is applying in particular when acting on json files. We have seen that a file was not modified when using the `Json_File` function as an argument to a modifying function because the modifying function just received a copy of the json file. This is not true when using the `Jbin_File` function that does not serialize the binary document and make it directly accessible. Also, as we have seen earlier, json functions that modify their first file parameter modify the file and return the file name. This is done by directly serializing the internal binary document as a file.

However, the “Jbin” counterpart of these functions does not serialize the binary document and thus does not modify the json file. For example let us compare these two queries:

```
/* First query */
```

```
select Json_Object(Jbin_Object_Add(Jbin_File('bt2.json'), 4 as "d") as "Jbin_bt1")
as "Result";
```

```
/* Second query */
```

```
select Json_Object(Json_Object_Add(Jbin_File('bt2.json'), 4 as "d") as "Jfile_bt1")
as "Result";
```

Both queries return:

| Result                            |
|-----------------------------------|
| {"bt1":{"a":1,"b":2,"c":3,"d":4}} |

In the first query `Jbin_Object_Add` does not serialize the document (no “Jbin” functions do) and `Json_Object` just returns a serialized modified tree. Consequently, the file `bt2.json` is not modified. This query is all right to copy a modified version of the json file without modifying it.

However, in the second query `Json_Object_Add` does modify the json file and returns the file name. The `Json_Object` function receives this file name, reads and parses the file, makes an object from it and returns the serialized result. This modification can be done willingly but can be an unwanted side effect of the query.

Therefore, using “Jbin” argument functions, in addition to being faster and using less memory, are also safer when dealing with json files that should not be modified.

## Using JSON as Dynamic Columns

The JSON nosql language has all the features to be used as an alternative to dynamic columns. For instance, take the following example of dynamic columns:

```
create table assets (
    item_name varchar(32) primary key, /* A common attribute for all items */
    dynamic_cols blob /* Dynamic columns will be stored here */
);

INSERT INTO assets VALUES
('MariaDB T-shirt', COLUMN_CREATE('color', 'blue', 'size', 'XL'));

INSERT INTO assets VALUES
('Thinkpad Laptop', COLUMN_CREATE('color', 'black', 'price', 500));

SELECT item_name, COLUMN_GET(dynamic_cols, 'color' as char) AS color FROM assets;
+-----+-----+
| item_name | color |
+-----+-----+
| MariaDB T-shirt | blue |
| Thinkpad Laptop | black |
+-----+-----+
```

```
/* Remove a column: */
```

```
UPDATE assets SET dynamic_cols=COLUMN_DELETE(dynamic_cols, "price")
WHERE COLUMN_GET(dynamic_cols, 'color' as char)='black';
```

```
/* Add a column: */
```

```
UPDATE assets SET dynamic_cols=COLUMN_ADD(dynamic_cols, 'warranty', '3 years')
WHERE item_name='Thinkpad Laptop';
```

```
/* You can also list all columns, or get them together with their values in JSON format: */
```

```
SELECT item_name, column_list(dynamic_cols) FROM assets;
+-----+-----+
| item_name | column_list(dynamic_cols) |
+-----+-----+
| MariaDB T-shirt | `size`, `color` |
| Thinkpad Laptop | `color`, `warranty` |
+-----+-----+  
  
SELECT item_name, COLUMN_JSON(dynamic_cols) FROM assets;
+-----+-----+
| item_name | COLUMN_JSON(dynamic_cols) |
+-----+-----+
| MariaDB T-shirt | [{"size": "XL", "color": "blue"}] |
| Thinkpad Laptop | [{"color": "black", "warranty": "3 years"}] |
+-----+-----+
```

The same result can be obtained with json columns using the json UDF's:

/\* JSON equivalent \*/

```

create table jassets (
    item_name varchar(32) primary key, /* A common attribute for all items */
    json_cols varchar(512) /* Jason columns will be stored here */
);

INSERT INTO jassets VALUES
    ('MariaDB T-shirt', Json_Object('blue' color, 'XL' size));

INSERT INTO jassets VALUES
    ('Thinkpad Laptop', Json_Object('black' color, 500 price));

SELECT item_name, JsonGet_String(json_cols, 'color') AS color FROM jassets;
+-----+-----+
| item_name | color |
+-----+-----+
| MariaDB T-shirt | blue |
| Thinkpad Laptop | black |
+-----+-----+

```

```
/* Remove a column: */
```

```
UPDATE jassets SET json_cols=Json_Object_Delete(json_cols, 'price')  
WHERE JsonGet String(json_cols, 'color')='black';
```

```
/* Add a column */
```

```
UPDATE jassets SET json_cols=Json_Object_Add(json_cols, '3 years' warranty)
 WHERE item name='Thinkpad Laptop';
```

*/\* You can also list all columns, or get them together with their values in JSON format: \*/*

```
SELECT item_name, Json_Object_List(json_cols) FROM jassets;
+-----+-----+
| item_name | Json_Object_List(json_cols) |
+-----+-----+
| MariaDB T-shirt | [{"color","size"}] |
| Thinkpad Laptop | [{"color","warranty"}] |
+-----+-----+  
  
SELECT item_name, json_cols FROM jassets;
+-----+-----+
| item_name | json_cols |
+-----+-----+
| MariaDB T-shirt | [{"color":"blue","size":"XL"}] |
| Thinkpad Laptop | [{"color":"black","warranty":"3 years"}] |
+-----+-----+
```

However, using JSON brings features not existing in dynamic columns:

- Use of a language used by many implementation and developers.
  - Full support of arrays, currently missing from dynamic columns.
  - Access of subpart of json by JPATH that can include calculations on arrays.
  - Possible references to json files

With more experience, additional UDFs can be easily written to support new needs.

## New Set of BSON Functions

All these functions have been rewritten using the new JSON handling way and are temporarily available changing the J starting name to B. Then Json\_Make\_Array new style is called using Bson\_Make\_Array. Some, such as Bson\_Item\_Delete, are new and some fix bugs found in their Json counterpart.

## Converting Tables to JSON

The JSON UDF's and the direct Jpath "\*" facility are powerful tools to convert table and files to the JSON format. For instance, the file

```
biblio3.json  
we used previously can be obtained by converting the  
xsample.xml file  
. This can be done like this:
```

From Connect 1.07.0002

```
create table xj1 (row varchar(500) jpath='*') engine=connect table_type=JSON file_name='biblio3.json' option_list='jmode=2';
```

Before Connect 1.07.0002

```
create table xj1 (row varchar(500) field_format='*'  
engine=connect table_type=JSON file_name='biblio3.json' option_list='jmode=2';
```

And then :

```
insert into xj1  
select json_object_nonull(ISBN, language LANG, SUBJECT,  
json_array_grp(json_object(authorfn FIRSTNAME, authorln LASTNAME)) json_AUTHOR, TITLE,  
json_object(translated PREFIX, json_object(tranfn FIRSTNAME, tranln LASTNAME) json_TRANSLATOR)  
json_TRANSLATED, json_object(publisher NAME, location PLACE) json_PUBLISHER, date DATEPUB)  
from xsampall2 group by isbn;
```

The xj1 table rows will directly receive the Json object made by the select statement used in the insert statement and the table file will be made as shown (xj1 is pretty=2 by default) Its mode is Jmode=2 because the values inserted are strings even if they denote json objects.

Another way to do this is to create a table describing the file format we want before the

```
biblio3.json  
file existed:
```

From Connect 1.07.0002

```
create table jsampall3 (  
ISBN char(15),  
LANGUAGE char(2) jpath='LANG',  
SUBJECT char(32),  
AUTHORFN char(128) jpath='AUTHOR:[X]:FIRSTNAME',  
AUTHORLN char(128) jpath='AUTHOR:[X]:LASTNAME',  
TITLE char(32),  
TRANSLATED char(32) jpath='TRANSLATOR:PREFIX',  
TRANSLATORFN char(128) jpath='TRANSLATOR:FIRSTNAME',  
TRANSLATORLN char(128) jpath='TRANSLATOR:LASTNAME',  
PUBLISHER char(20) jpath='PUBLISHER:NAME',  
LOCATION char(20) jpath='PUBLISHER:PLACE',  
DATE int(4) jpath='DATEPUB')  
engine=CONNECT table_type=JSON file_name='biblio3.json';
```

Before Connect 1.07.0002

```

create table jsampall3 (
ISBN char(15),
LANGUAGE char(2) field_format='LANG',
SUBJECT char(32),
AUTHORFN char(128) field_format='AUTHOR:[X]:FIRSTNAME',
AUTHORLN char(128) field_format='AUTHOR:[X]:LASTNAME',
TITLE char(32),
TRANSLATED char(32) field_format='TRANSLATOR:PREFIX',
TRANSLATORFN char(128) field_format='TRANSLATOR:FIRSTNAME',
TRANSLATORLN char(128) field_format='TRANSLATOR:LASTNAME',
PUBLISHER char(20) field_format='PUBLISHER:NAME',
LOCATION char(20) field_format='PUBLISHER:PLACE',
DATE int(4) field_format='DATEPUB'
engine=CONNECT table_type=JSON file_name='biblio3.json';

```

and to populate it by:

```
insert into jsampall3 select * from xsampall;
```

This is a simpler method. However, the issue is that this method cannot handle the multiple column values. This is why we inserted from

```

xsampall
not from
xsampall2
```

. How can we add the missing multiple authors in this table? Here again we must create a utility table able to handle JSON strings. From Connect 1.07.0002

```

create table xj2 (ISBN char(15), author varchar(150) jpath='AUTHOR:*') engine=connect table_type=JSON file_name='biblio3.json' o
,
```

Before Connect 1.07.0002

```

create table xj2 (ISBN char(15), author varchar(150) field_format='AUTHOR:*')
engine=connect table_type=JSON file_name='biblio3.json' option_list='jmode=1';
```

```

update xj2 set author =
(select json_array_gra(json_object(authorfn FIRSTNAME, authorln LASTNAME))
 from xsampall2 where isbn = xj2.isbn);
```

Voilà !

## Converting json files

We have seen that json files can be formatted differently depending on the pretty option. In particular, big data files should be formatted with pretty equal to 0 when used by a CONNECT json table. The best and simplest way to convert a file from one format to another is to use the *Jfile\_Make* function. Indeed this function makes a file of specified format using the syntax:

```
Jfile_Make(json_document, [file_name], [pretty]);
```

The file name is optional when the json document comes from a *Jbin\_File* function because the returned structure makes it available. For instance, to convert back the json file tb.json to pretty=0, this can be simply done by:

```
select Jfile_Make(Jbin_File('tb.json'), 0);
```

## Performance Consideration

MySQL and PostgreSQL have a JSON data type that is not just text but an internal encoding of JSON data. This is to save parsing time when executing JSON functions. Of course, the parse must be done anyway when creating the data and serializing must be done to output the result.

CONNECT directly works on character strings impersonating JSON values with the need of parsing them all the time but with the advantage of working easily on external data. Generally, this is not too penalizing because JSON data are often of some or reasonable size. The only case where it can be a serious problem is when working on a big JSON file.

Then, the file should be formatted or converted to *pretty=0*.

From Connect 1.7.002, this easily done using the *Jfile\_Convert* function, for instance:

```
select jfile_convert('bibdoc.json','bibdoc0.json',350);
```

Such a json file should not be used directly by JSON UDFs because they parse the whole file, even when only a subset is used. Instead, it should be used

by a JSON table created on it. Indeed, JSON tables do not parse the whole document but just the item corresponding to the row they are working on. In addition, indexing can be used by the table as explained previously on this page.

Generally speaking, the maximum flexibility offered by CONNECT is by using JSON tables and JSON UDFs together. Some things are better handled by tables, other by UDFs. The tools are there but it is up to you to discover the best way to resolve your problems.

## Bjson files

Starting with Connect 1.7.002, *pretty=0* json files can be converted to a binary format that is a pre-parsed representation of json. This can be done with the Jfile\_Bjson UDF function, for instance:

```
select jfile_bjson('bigfile.json','binfile.json',3500);
```

Here the third argument, the record length, must 6 to 10 times larger than the lrec of the initial json file because the parsed representation is bigger than the original json text representation.

Tables using such Bjson files must specify 'Pretty=-1' in the option list.

It is probably similar to the BSON used by MongoDB and PostgreSQL and permits to process queries up to 10 times faster than working on text json files. Indexing is also available for tables using this format making even more performance improvement. For instance, some queries on a json table of half a million rows, that were previously done in more than 10 seconds, took only 0.1 second when converted and indexed.

Here again, this has been remade to use the new way Json is handled. The files made using the bfile\_bjson function are only from two to four times the size of the source files. This new representation is not compatible with the old one. Therefore, these files must be used with BSON tables only.

## Specifying a JSON table Encoding

An important feature of JSON is that strings should in UNICODE. As a matter of fact, all examples we have found on the Internet seemed to be just ASCII. This is because UNICODE is generally encoded in JSON files using UTF8 or UTF16 or UTF32.

To specify the required encoding, just use the data\_charset CONNECT option or the native DEFAULT CHARSET option.

## Retrieving JSON data from MongoDB

Classified as a NoSQL database program, MongoDB uses JSON-like documents (BSON) grouped in collections. The simplest way, and only method available before Connect 1.6, to access MongoDB data was to export a collection to a JSON file. This produces a file having the pretty=0 format. Viewed as SQL, a collection is a table and documents are table rows.

Since CONNECT version 1.6, it is now possible to directly access MongoDB collections via their MongoDB C Driver. This is the purpose of the MONGO table type described later. However, JSON tables can also do it in a somewhat different way (providing MONGO support is installed as described for MONGO tables).

It is achieved by specifying the MongoDB connection URI while creating the table. For instance:

From Connect 1.7.002

```
create or replace table jinvent (
  _id char(24) not null,
  item char(12) not null,
  instock varchar(300) not null jpath='instock.*')
engine=connect table_type=JSON tablename='inventory' lrec=512
connection='mongodb://localhost:27017';
```

Before Connect 1.7.002

```
create or replace table jinvent (
  _id char(24) not null,
  item char(12) not null,
  instock varchar(300) not null field_format='instock.*')
engine=connect table_type=JSON tablename='inventory' lrec=512
connection='mongodb://localhost:27017';
```

In this statement, the *file\_name* option was replaced by the *connection* option. It is the URI enabling to retrieve data from a local or remote MongoDB server. The *tablename* option is the name of the MongoDB collection that will be used and the *dbname* option could have been used to indicate the database containing the collection (it defaults to the current database).

The way it works is that the documents retrieved from MongoDB are serialized and CONNECT uses them as if they were read from a file. This implies serializing by MongoDB and parsing by CONNECT and is not the best performance wise. CONNECT tries its best to reduce the data transfer when a query contains a reduced column list and/or a where clause. This way makes all the possibilities of the JSON table type available, such as calculated arrays.

However, to work on large JSON collations, using the MONGO table type is generally the normal way.

Note: JSON tables using the MongoDB access accept the specific MONGO options *colist*, *filter* and *pipeline*. They are described in the MONGO table chapter.

# Summary of Options and Variables Used with Json Tables

Options and variables that can be used when creating Json tables are listed here:

| Table Option | Type    | Description                                                                                                                                                                    |
|--------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ENGINE       | String  | Must be specified as CONNECT.                                                                                                                                                  |
| TABLE_TYPE   | String  | Must be JSON or BSON.                                                                                                                                                          |
| FILE_NAME    | String  | The optional file (path) name of the Json file. Can be absolute or relative to the current data directory. If not specified, it defaults to the table name and json file type. |
| DATA_CHARSET | String  | Set it to 'utf8' for most Unicode Json documents.                                                                                                                              |
| LRECL        | Number  | The file record size for pretty < 2 json files.                                                                                                                                |
| HTTP         | String  | The HTTP of the server of REST queries.                                                                                                                                        |
| URI          | String  | THE URI of REST queries                                                                                                                                                        |
| CONNECTION*  | String  | Specifies a connection to<br>MONGODB                                                                                                                                           |
| ZIPPED       | Boolean | True if the json file(s) is/are zipped in one or several zip files.                                                                                                            |
| MULTIPLE     | Number  | Used to specify a multiple file table.                                                                                                                                         |
| SEP_CHAR     | String  | Set it to ':' for old tables using the old json path syntax.                                                                                                                   |
| CATFUNC      | String  | The catalog function (column) used when creating a catalog table.                                                                                                              |
| OPTION_LIST  | String  | Used to specify all other options listed below.                                                                                                                                |

(\*) For Json tables connected to MongoDB, Mongo specific options can also be used.

Other options must be specified in the option list:

| Table Option | Type    | Description                                                                                                                                                    |
|--------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DEPTH_LEVEL  | Number  | Specifies the depth in the document CONNECT looks when defining columns by discovery or in catalog tables                                                      |
| PRETTY       | Number  | Specifies the format of the Json file (-1 for Bjson files)                                                                                                     |
| EXPAND       | String  | The name of the column to expand.                                                                                                                              |
| OBJECT       | String  | The json path of the sub-document used for the table.                                                                                                          |
| BASE         | Number  | The numbering base for arrays: 0 (the default) or 1.                                                                                                           |
| LIMIT        | Number  | The maximum number of array values to use when concatenating, calculating or expanding arrays. Defaults to 50 (>= Connect 1.7.0003), 10 (<= Connect 1.7.0002). |
| FULLARRAY    | Boolean | Used when creating with Discovery. Make a column for each value of arrays (up to LIMIT).                                                                       |
| JMODE        | Number  | The Json mode (array of objects, array of arrays, or array of values) Only used when inserting new rows.                                                       |
| ACCEPT       | Boolean | Keep null columns (for discovery).                                                                                                                             |
| AVGLEN       | Number  | An estimate average length of rows. This is used only when indexing and can be set if indexing fails by miscalculating the table max size.                     |
| STRINGIFY    | String  | Ask discovery to make a column to return the Json representation of this object.                                                                               |

Column options:

| Column Option      | Type   | Description                                                                                 |
|--------------------|--------|---------------------------------------------------------------------------------------------|
| JPATH_FIELD_FORMAT | String | Defaults to the column name.                                                                |
| DATE_FORMAT        | String | Specifies the date format into the Json file when defining a DATE, DATETIME or TIME column. |

Variables used with Json tables are:

- [connect\\_default\\_depth](#)
- [connect\\_json\\_null](#)
- [connect\\_json\\_all\\_path](#)
- [connect\\_force\\_bson](#)

# Notes

1. ↑ The value n can be 0 based or 1 based depending on the base table option. The default is 0 to match what is the current usage in the Json world but it can be set to 1 for tables created in old versions.
2. ↑ See for instance: <https://mariadb.com/kb/en/mariadb/json-functions/> , [https://github.com/mysqludf/lib\\_mysqludf\\_json#readme](https://github.com/mysqludf/lib_mysqludf_json#readme) and [https://blogs.oracle.com/svetasmirnova/entry/json\\_udf\\_functions\\_version\\_04](https://blogs.oracle.com/svetasmirnova/entry/json_udf_functions_version_04)
3. ↑ This will not work when CONNECT is compiled embedded

## 4.3.7.6.13 CONNECT XML Table Type

### Contents

1. [Overview](#)
2. [Creating XML tables](#)
3. [Using Xpaths with XML tables](#)
  1. [Libxml2 default name space issue](#)
  2. [Direct access on XML tables](#)
  3. [Accessing tags with namespaces](#)
4. [Having Columns defined by Discovery](#)
5. [Write operations on XML tables](#)
6. [Multiple nodes in the XML document](#)
7. [Intermediate multiple node](#)
8. [Making a List of Multiple Values](#)
  1. [What if a table contains several multiple nodes](#)
9. [Support of HTML Tables](#)
  1. [New file setting](#)
10. [Notes](#)

### Overview

CONNECT supports tables represented by XML files. For these tables, the standard input/output functions of the operating system are not used but the parsing and processing of the file is delegated to a specialized library. Currently two such systems are supported: libxml2, a part of the GNOME framework, but which does not require GNOME and, on Windows, MS-DOM (DOMDOC), the Microsoft standard support of XML documents.

DOMDOC is the default for the Windows version of CONNECT and libxml2 is always used on other systems. On Windows the choice can be specified using the XMLSUP [CREATE TABLE](#) list option, for instance specifying

```
option_list='xmlsup=libxml2'
```

### Creating XML tables

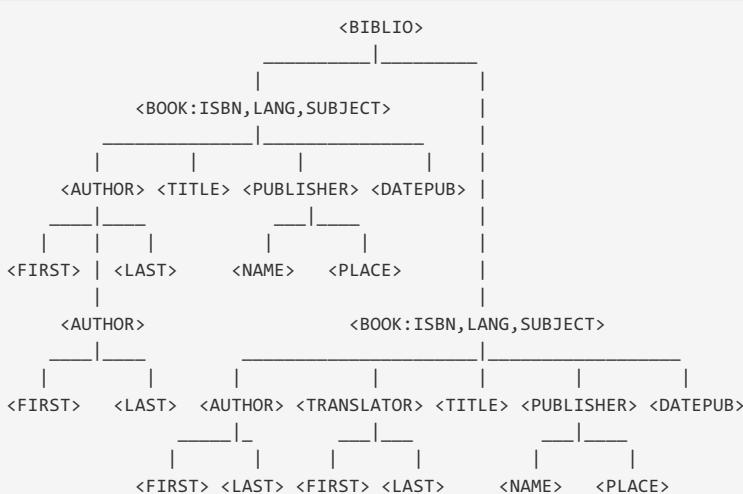
First of all, it must be understood that XML is a very general language used to encode data having any structure. In particular, the tag hierarchy in an XML file describes a tree structure of the data. For instance, consider the file:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<BIBLIO SUBJECT="XML">
  <BOOK ISBN="9782212090819" LANG="fr" SUBJECT="applications">
    <AUTHOR>
      <FIRSTNAME>Jean-Christophe</FIRSTNAME>
      <LASTNAME>Bernadac</LASTNAME>
    </AUTHOR>
    <AUTHOR>
      <FIRSTNAME>François</FIRSTNAME>
      <LASTNAME>Khab</LASTNAME>
    </AUTHOR>
    <TITLE>Construire une application XML</TITLE>
    <PUBLISHER>
      <NAME>Eyrolles</NAME>
      <PLACE>Paris</PLACE>
    </PUBLISHER>
    <DATEPUB>1999</DATEPUB>
  </BOOK>
  <BOOK ISBN="9782840825685" LANG="fr" SUBJECT="applications">
    <AUTHOR>
      <FIRSTNAME>William J.</FIRSTNAME>
      <LASTNAME>Pardi</LASTNAME>
    </AUTHOR>
    <TRANSLATOR PREFIX="adapté de l'anglais par">
      <FIRSTNAME>James</FIRSTNAME>
      <LASTNAME>Guerin</LASTNAME>
    </TRANSLATOR>
    <TITLE>XML en Action</TITLE>
    <PUBLISHER>
      <NAME>Microsoft Press</NAME>
      <PLACE>Paris</PLACE>
    </PUBLISHER>
    <DATEPUB>1999</DATEPUB>
  </BOOK>
</BIBLIO>

```

It represents data having the structure:



This structure seems at first view far from being tabular. However, modern database management systems, including MariaDB, implement something close to the relational model and work on tables that are structurally not hierarchical but tabular with rows and columns.

Nevertheless, CONNECT can do it. Of course, it cannot guess what you want to extract from the XML structure, but gives you the possibility to specify it when you create the table [1].

Let us take a first example. Suppose you want to make a table from the above document, displaying the node contents.

For this, you can define a table `xsampletag` as:

```

create table xsampletag (
  AUTHOR char(50),
  TITLE char(32),
  TRANSLATOR char(40),
  PUBLISHER char(40),
  DATEPUB int(4))
engine=CONNECT table_type=XML file_name='Xsample.xml';

```

It will be displayed as:

| AUTHOR                   | TITLE                          | TRANSLATOR   | PUBLISHER             | DATEPUB |
|--------------------------|--------------------------------|--------------|-----------------------|---------|
| Jean-Christophe Bernadac | Construire une application XML | <null>       | Eyrolles Paris        | 1999    |
| William J. Pardi         | XML en Action                  | James Guerin | Microsoft Press Paris | 1999    |

Let us try to understand what happened. By default the column names correspond to tag names. Because this file is rather simple, CONNECT was able to default the top tag of the table as the root node

```
<BIBLIO>
of the file, and the row tags as the
<BOOK>
children of the table tag. In a more complex file, this should have been specified, as we will see later. Note that we didn't have to worry about the
sub-tags such as
<FIRSTNAME>
or
<LASTNAME>
because CONNECT automatically retrieves the entire text contained in a tag and its sub-tags [2].
```

Only the first author of the first book appears. This is because only the first occurrence of a column tag has been retrieved so the result has a proper tabular structure. We will see later what we can do about that.

How can we retrieve the values specified by attributes? By using a *Coltype* table option to specify the default column type. The value '@' means that column names match attribute names. Therefore, we can retrieve them by creating a table such as:

```
create table xsampattr (
  ISBN char(15),
  LANG char(2),
  SUBJECT char(32))
engine=CONNECT table_type=XML file_name='Xsample.xml'
option_list='Coltype=@';
```

This table returns the following:

ISBN	LANG	SUBJECT
9782212090819	fr	applications
9782840825685	fr	applications

Now to define a table that will give us all the previous information, we must specify the column type for each column. Because in the next statement the column type defaults to Node, the *field\_format* column parameter was used to indicate which columns are attributes:

From Connect 1.7.0002

```
create table xsamp (
  ISBN char(15) xpath='@',
  LANG char(2) xpath='@',
  SUBJECT char(32) xpath='@',
  AUTHOR char(50),
  TITLE char(32),
  TRANSLATOR char(40),
  PUBLISHER char(40),
  DATEPUB int(4))
engine=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK';
```

Before Connect 1.7.0002

```
create table xsamp (
  ISBN char(15) field_format='@',
  LANG char(2) field_format='@',
  SUBJECT char(32) field_format='@',
  AUTHOR char(50),
  TITLE char(32),
  TRANSLATOR char(40),
  PUBLISHER char(40),
  DATEPUB int(4))
engine=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK';
```

Once done, we can enter the query:

```
select subject, lang, title, author from xsamp;
```

This will return the following result:

SUBJECT	LANG	TITLE	AUTHOR
applications	fr	Construire une application XML	Jean-Christophe Bernadac
applications	fr	XML en Action	William J. Pardi

Note that we have been lucky. Because unlike SQL, XML is case sensitive and the column names have matched the node names only because the column names were given in upper case. Note also that the order of the columns in the table could have been different from the order in which the nodes appear in the XML file.

## Using Xpaths with XML tables

Xpath is used by XML to locate and retrieve nodes. The table's main node Xpath is specified by the

tabname  
option. If just the node name is given, CONNECT constructs an Xpath such as

'

BIBLIO'

in the example above that should retrieve the

BIBLIO  
node wherever it is within the XML file.

The row nodes are by default the children of the table node. However, for instance to eliminate some children nodes that are not real row nodes, the row node name can be specified using the

rownode  
sub-option of the  
option\_list  
option.

The field\_format options we used above can be specified to locate more precisely where and what information to retrieve using an Xpath-like syntax. For instance:

From Connect 1.7.0002

```
create table xsampall (
isbn char(15) xpath='@ISBN',
language char(2) xpath='@LANG',
subject char(32) xpath='@SUBJECT',
authorfn char(20) xpath='AUTHOR/FIRSTNAME',
authorln char(20) xpath='AUTHOR/LASTNAME',
title char(32) xpath='TITLE',
translated char(32) xpath='TRANSLATOR/@PREFIX',
tranfn char(20) xpath='TRANSLATOR/FIRSTNAME',
tranln char(20) xpath='TRANSLATOR/LASTNAME',
publisher char(20) xpath='PUBLISHER/NAME',
location char(20) xpath='PUBLISHER/PLACE',
year int(4) xpath='DATEPUB')
engine=CONNECT table_type=XML file_name='Xsample.xml'
tablename='BIBLIO' option_list='rownode=BOOK';
```

Before Connect 1.7.0002

```
create table xsampall (
isbn char(15) field_format='@ISBN',
language char(2) field_format='@LANG',
subject char(32) field_format='@SUBJECT',
authorfn char(20) field_format='AUTHOR/FIRSTNAME',
authorln char(20) field_format='AUTHOR/LASTNAME',
title char(32) field_format='TITLE',
translated char(32) field_format='TRANSLATOR/@PREFIX',
tranfn char(20) field_format='TRANSLATOR/FIRSTNAME',
tranln char(20) field_format='TRANSLATOR/LASTNAME',
publisher char(20) field_format='PUBLISHER/NAME',
location char(20) field_format='PUBLISHER/PLACE',
year int(4) field_format='DATEPUB')
engine=CONNECT table_type=XML file_name='Xsample.xml'
tablename='BIBLIO' option_list='rownode=BOOK';
```

This very flexible column parameter serves several purposes:

- To specify the tag name, or the attribute name if different from the column name.
- To specify the type (tag or attribute) by a prefix of '@' for attributes.
- To specify the path for sub-tags using the '/' character.

This path is always relative to the current context (the column top node) and cannot be specified as an absolute path from the document root, therefore a leading '/' cannot be used. The path cannot be variable in node names or depth, therefore using '

//  
' is not allowed.

The query:

```
select isbn, title, translated, tranfn, tranln, location from
xsampall where translated is not null;
```

replies:

ISBN	TITLE	TRANSLATED	TRANFN	TRANLN	LOCATION
9782840825685	XML en Action	adapté de l'anglais par	James	Guerin	Paris

## Libxml2 default name space issue

An issue with libxml2 is that some files can declare a default name space in their root node. Because Xpath only searches in that name space, the nodes will not be found if they are not prefixed. If this happens, specify the tablename option as an Xpath ignoring the current name space:

```
TABNAME="//*[local-name()='BIBLIO']"
```

This must also be done for the default of specified Xpath of the not attribute columns. For instance:

```
title char(32) field_format="*[local-name()='TITLE']",
```

Note: This raises an error (and is useless anyway) with DOMDOC.

## Direct access on XML tables

Direct access is available on XML tables. This means that XML tables can be sorted and used in joins, even in the one-side of the join.

However, building a permanent index is not yet implemented. It is unclear whether this can be useful. Indeed, the DOM implementation that is used to access these tables firstly parses the whole file and constructs a node tree in memory. This may often be the longest part of the process, so the use of an index would not be of great value. Note also that this limits the XML files to a reasonable size. Anyway, when speed is important, this table type is not the best to use. Therefore, in these cases, it is probably better to convert the file to another type by inserting the XML table into another table of a more appropriate type for performance.

## Accessing tags with namespaces

With the Windows DOMDOC support, this can be done using the prefix in the tablename column option and/or xpath column option. For instance, given the file gns.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx xmlns:gns="http:dummy">
<gnss:trkseg>
<trkpt lon="-121.9822235107421875" lat="37.3884925842285156">
<gnss:ele>6.610851287841797</gnss:ele>
<time>2014-04-01T14:54:05.000Z</time>
</trkpt>
<trkpt lon="-121.9821929931640625" lat="37.3885803222656250">
<ele>6.787827968597412</ele>
<time>2014-04-01T14:54:08.000Z</time>
</trkpt>
<trkpt lon="-121.9821624755859375" lat="37.3886299133300781">
<ele>6.771987438201904</ele>
<time>2014-04-01T14:54:10.000Z</time>
</trkpt>
</gnss:trkseg>
</gpx>
```

and the defined CONNECT table:

```

CREATE TABLE xgns (
`lon` double(21,16) NOT NULL `xpath`='@',
`lat` double(20,16) NOT NULL `xpath`='@',
`ele` double(21,16) NOT NULL `xpath`='gns:ele',
`time` datetime date_format="YYYY-MM-DD 'T' hh:mm:ss '.000Z'"
)
ENGINE=CONNECT DEFAULT CHARSET=latin1 `table_type`=XML
`file_name`='gns.xml' tabname='gns:trkseg' option_list='xmlsup=domdoc';

```

```
select * from xgns;
```

Displays:

lon	lat	ele	time
-121,982223510742	37,3884925842285	6,6108512878418	01/04/2014 14:54:05
-121,982192993164	37,3885803222656	0	01/04/2014 14:54:08
-121,982162475586	37,3886299133301	0	01/04/2014 14:54:10

Only the prefixed 'ele' tag is recognized.

However, this does not work with the libxml2 support. The solution is then to use a function ignoring the name space:

```

CREATE TABLE xgns2 (
`lon` double(21,16) NOT NULL `xpath`='@',
`lat` double(20,16) NOT NULL `xpath`='@',
`ele` double(21,16) NOT NULL `xpath`="*[local-name()='ele']",
`time` datetime date_format="YYYY-MM-DD 'T' hh:mm:ss '.000Z'"
)
ENGINE=CONNECT DEFAULT CHARSET=latin1 `table_type`=XML
`file_name`='gns.xml' tabname="*[local-name()='trkseg']" option_list='xmlsup=libxml2';

```

Then :

```
select * from xgns2;
```

Displays:

lon	lat	ele	time
-121,982223510742	37,3884925842285	6,6108512878418	01/04/2014 14:54:05
-121,982192993164	37,3885803222656	6.7878279685974	01/04/2014 14:54:08
-121,982162475586	37,3886299133301	6.7719874382019	01/04/2014 14:54:10

This time, all 'ele' tags are recognized. This solution does not work with DOMDOC.

## Having Columns defined by Discovery

It is possible to let the MariaDB discovery process do the job of column specification. When columns are not defined in the `CREATE TABLE` statement, CONNECT endeavours to analyze the XML file and to provide the column specifications. This is possible only for true XML tables, but not for HTML tables.

For instance, the `xsamp` table could have been created specifying:

```

create table xsamp
engine=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK';

```

Let's check how it was actually specified using the `SHOW CREATE TABLE` statement:

```

CREATE TABLE `xsamp` (
  `ISBN` char(13) NOT NULL `FIELD_FORMAT`='@',
  `LANG` char(2) NOT NULL `FIELD_FORMAT`='@',
  `SUBJECT` char(12) NOT NULL `FIELD_FORMAT`='@',
  `AUTHOR` char(24) NOT NULL,
  `TRANSLATOR` char(12) DEFAULT NULL,
  `TITLE` char(30) NOT NULL,
  `PUBLISHER` char(21) NOT NULL,
  `DATEPUB` char(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='XML'
`FILE_NAME`='E:/Data/Xml/Xsample.xml' `TABNAME`='BIBLIO' `OPTION_LIST`='rownode=BOOK';

```

It is equivalent except for the column sizes that have been calculated from the file as the maximum length of the corresponding column when it was a normal value. Also, all columns are specified as type **CHAR** because XML does not provide information about the node content data type. Nullable is set to true if the column is missing in some rows.

If a more complex definition is desired, you can ask CONNECT to analyse the XPATH up to a given level using the level option in the option list. The level value is the number of nodes that are taken in the XPATH. For instance:

```

create table xsampall
engine=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK,Level=1';

```

This will define the table as:

From Connect 1.7.0002

```

CREATE TABLE `xsampall` (
  `ISBN` char(13) NOT NULL `XPATH`='@',
  `LANG` char(2) NOT NULL `XPATH`='@',
  `SUBJECT` char(12) NOT NULL `XPATH`='@',
  `AUTHOR_FIRSTNAME` char(15) NOT NULL `XPATH`='AUTHOR/FIRSTNAME',
  `AUTHOR_LASTNAME` char(8) NOT NULL `XPATH`='AUTHOR/LASTNAME',
  `TRANSLATOR_PREFIX` char(24) DEFAULT NULL `XPATH`='TRANSLATOR/@PREFIX',
  `TRANSLATOR_FIRSTNAME` char(7) DEFAULT NULL `XPATH`='TRANSLATOR/FIRSTNAME',
  `TRANSLATOR_LASTNAME` char(6) DEFAULT NULL `XPATH`='TRANSLATOR/LASTNAME',
  `TITLE` char(30) NOT NULL,
  `PUBLISHER_NAME` char(15) NOT NULL `XPATH`='PUBLISHER/NAME',
  `PUBLISHER_PLACE` char(5) NOT NULL `XPATH`='PUBLISHER/PLACE',
  `DATEPUB` char(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='XML' `FILE_NAME`='Xsample.xml' `TABNAME`='BIBLIO' `OPTION_LIST`='rownode=B

```

Before Connect 1.7.0002

```

CREATE TABLE `xsampall` (
  `ISBN` char(13) NOT NULL `FIELD_FORMAT`='@',
  `LANG` char(2) NOT NULL `FIELD_FORMAT`='@',
  `SUBJECT` char(12) NOT NULL `FIELD_FORMAT`='@',
  `AUTHOR_FIRSTNAME` char(15) NOT NULL `FIELD_FORMAT`='AUTHOR/FIRSTNAME',
  `AUTHOR_LASTNAME` char(8) NOT NULL `FIELD_FORMAT`='AUTHOR/LASTNAME',
  `TRANSLATOR_PREFIX` char(24) DEFAULT NULL `FIELD_FORMAT`='TRANSLATOR/@PREFIX',
  `TRANSLATOR_FIRSTNAME` char(7) DEFAULT NULL `FIELD_FORMAT`='TRANSLATOR/FIRSTNAME',
  `TRANSLATOR_LASTNAME` char(6) DEFAULT NULL `FIELD_FORMAT`='TRANSLATOR/LASTNAME',
  `TITLE` char(30) NOT NULL,
  `PUBLISHER_NAME` char(15) NOT NULL `FIELD_FORMAT`='PUBLISHER/NAME',
  `PUBLISHER_PLACE` char(5) NOT NULL `FIELD_FORMAT`='PUBLISHER/PLACE',
  `DATEPUB` char(4) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='XML' `FILE_NAME`='Xsample.xml'
`TABNAME`='BIBLIO' `OPTION_LIST`='rownode=BOOK,Level=1';

```

This method can be used as a quick way to make a “template” table definition that can later be edited to make the desired definition. In particular, column names are constructed from all the nodes of their path in order to have distinct column names. This can be manually edited to have the desired names, provided their XPATH is not modified.

To have a preview of how columns will be defined, you can use a catalog table like this:

```

create table xsacol
engine=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK,Level=1' catfunc=col;

```

And when asking:

```

select column_name Name, type_name Type, column_size Size, nullable, xpath from xsacol;

```

You get the description of what the table columns will be:

Name	Type	Size	nullable	xpath
ISBN	CHAR	13	0	@
LANG	CHAR	2	0	@
SUBJECT	CHAR	12	0	@
AUTHOR_FIRSTNAME	CHAR	15	0	AUTHOR/FIRSTNAME
AUTHOR_LASTNAME	CHAR	8	0	AUTHOR/LASTNAME
TRANSLATOR_PREFIX	CHAR	24	1	TRANSLATOR/@PREFIX
TRANSLATOR_FIRSTNAME	CHAR	7	1	TRANSLATOR/FIRSTNAME
TRANSLATOR_LASTNAME	CHAR	6	1	TRANSLATOR/LASTNAME
TITLE	CHAR	30	0	
PUBLISHER_NAME	CHAR	15	0	PUBLISHER/NAME
PUBLISHER_PLACE	CHAR	5	0	PUBLISHER/PLACE
DATEPUB	CHAR	4	0	

## Write operations on XML tables

You can freely use the Update, Delete and Insert commands with XML tables. However, you must understand that the format of the updated or inserted data follows the specifications of the table you created, not the ones of the original source file. For instance, let us suppose we insert a new book using the `xsamp` table (not the `xsampall` table) with the command:

```
insert into xsamp
(isbn, lang, subject, author, title, publisher,datepub)
values ('9782212090529','fr','général','Alain Michard',
'XML, Langage et Applications','Eyrolles Paris',1998);
```

Then if we ask:

```
select subject, author, title, translator, publisher from xsamp;
```

Everything seems correct when we get the result:

SUBJECT	AUTHOR	TITLE	TRANSLATOR	PUBLISHER
applications	Jean-Christophe Bernadac	Construire une application XML		Eyrolles Paris
applications	William J. Pardi	XML en Action	James Guerin	Microsoft Press Paris
général	Alain Michard	XML, Langage et Applications		Eyrolles Paris

However if we enter the apparently equivalent query on the `xsampall` table, based on the same file:

```
select subject,
concat(authorfn, ' ', authorln) author , title,
concat(tranfn, ' ', tranln) translator,
concat(publisher, ' ', location) publisher from xsampall;
```

this returns an apparently wrong answer:

SUBJECT	AUTHOR	TITLE	TRANSLATOR	PUBLISHER
applications	Jean-Christophe Bernadac	Construire une application XML		Eyrolles Paris
applications	William J. Pardi	XML en Action	James Guerin	Microsoft Press Paris
général		XML, Langage et Applications		

What happened here? Simply, because we used the `xsamp` table to do the Insert, what has been inserted within the XML file had the structure described for `xsamp`:

```

<BOOK ISBN="9782212090529" LANG="fr" SUBJECT="général">
  <AUTHOR>Alain Michard</AUTHOR>
  <TITLE>XML, Langage et Applications</TITLE>
  <TRANSLATOR></TRANSLATOR>
  <PUBLISHER>Eyrolles Paris</PUBLISHER>
  <DATEPUB>1998</DATEPUB>
</BOOK>

```

CONNECT cannot "invent" sub-tags that are not part of the xsamp table. Because these sub-tags do not exist, the xsampall table cannot retrieve the information that should be attached to them. If we want to be able to query the XML file by all the defined tables, the correct way to insert a new book to the file is to use the xsampall table, the only one that addresses all the components of the original document:

```

delete from xsamp where isbn = '9782212090529';

insert into xsampall (isbn, language, subject, authorfn, authorln,
                      title, publisher, location, year)
values('9782212090529', 'fr', 'général', 'Alain', 'Michard',
      'XML, Langage et Applications', 'Eyrolles', 'Paris', 1998);

```

Now the added book, in the XML file, will have the required structure:

```

<BOOK ISBN="9782212090529" LANG="fr" SUBJECT="général">
  <AUTHOR>
    <FIRSTNAME>Alain</FIRSTNAME>
    <LASTNAME>Michard</LASTNAME>
  </AUTHOR>
  <TITLE>XML, Langage et Applications</TITLE>
  <PUBLISHER>
    <NAME>Eyrolles</NAME>
    <PLACE>Paris</PLACE>
  </PUBLISHER>
  <DATEPUB>1998</DATEPUB>
</BOOK>

```

**Note:** We used a column list in the Insert statements when creating the table to avoid generating a <TRANSLATOR> node with sub-nodes, all containing null values (this works on Windows only).

## Multiple nodes in the XML document

Let us come back to the above example XML file. We have seen that the author node can be "multiple" meaning that there can be more than one author of a book. What can we do to get the complete information fitting the relational model? CONNECT provides you with two possibilities, but is restricted to only one such multiple node per table.

The first and most challenging one is to return as many rows than there are authors, the other columns being repeated as if we had make a join between the author column and the rest of the table. To achieve this, simply specify the "multiple" node name and the "expand" option when creating the table. For instance, we can create the xsamp2 table like this:

```

create table xsamp2 (
  ISBN char(15) field_format='@',
  LANG char(2) field_format='@',
  SUBJECT char(32) field_format='@',
  AUTHOR char(40),
  TITLE char(32),
  TRANSLATOR char(32),
  PUBLISHER char(32),
  DATEPUB int(4))
engine=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO'
option_list='rownode=BOOK,Expand=1,Mulnode=AUTHOR,Limit=2';

```

In this statement, the Limit option specifies the maximum number of values that will be expanded. If not specified, it defaults to

10

. Any values above the limit will be ignored and a warning message issued [3]. Now you can enter a query such as:

```
select isbn, subject, author, title from xsamp2;
```

This will retrieve and display the following result:

ISBN	SUBJECT	AUTHOR	TITLE
------	---------	--------	-------

9782212090819	applications	Jean-Christophe Bernadac	Construire une application XML
9782212090819	applications	François Knab	Construire une application XML
9782840825685	applications	William J. Pardi	XML en Action
9782212090529	général	Alain Michard	XML, Langage et Applications

In this case, this is as if the table had four rows. However if we enter the query:

```
select isbn, subject, title, publisher from xsamp2;
```

this time the result will be:

ISBN	SUBJECT	TITLE	PUBLISHER
9782212090819	applications	Construire une application XML	Eyrolles Paris
9782840825685	applications	XML en Action	Microsoft Press Paris
9782212090529	général	XML, Langage et Applications	Eyrolles Paris

Because the author column does not appear in the query, the corresponding row was not expanded. This is somewhat strange because this would have been different if we had been working on a table of a different type. However, it is closer to the relational model for which there should not be two identical rows (tuples) in a table. Nevertheless, you should be aware of this somewhat erratic behavior. For instance:

```
select count(*) from xsamp2;          /* Replies 3 */
select count(author) from xsamp2;      /* Replies 4 */
select count(isbn) from xsamp2;        /* Replies 3 */
select isbn, subject, title, publisher from xsamp2 where author <> '';
```

This last query replies:

ISBN	SUBJECT	TITLE	PUBLISHER
9782212090819	applications	Construire une application XML	Eyrolles Paris
9782212090819	applications	Construire une application XML	Eyrolles Paris
9782840825685	applications	XML en Action	Microsoft Press Paris
9782212090529	général	XML, Langage et Applications	Eyrolles Paris

Even though the author column does not appear in the result, the corresponding row was expanded because the multiple column was used in the where clause.

## Intermediate multiple node

The "multiple" node can be an intermediate node. If we want to do the same expanding with the xsampall table, there will be nothing more to do. The xsampall2 table can be created with:

From Connect 1.7.0002

```
create table xsampall2 (
isbn char(15) xpath='@ISBN',
language char(2) xpath='@LANG',
subject char(32) xpath='@SUBJECT',
authorfn char(20) xpath='AUTHOR/FIRSTNAME',
authorln char(20) xpath='AUTHOR/LASTNAME',
title char(32) xpath='TITLE',
translated char(32) xpath='TRANSLATOR/@PREFIX',
tranfn char(20) xpath='TRANSLATOR/FIRSTNAME',
tranln char(20) xpath='TRANSLATOR/LASTNAME',
publisher char(20) xpath='PUBLISHER/NAME',
location char(20) xpath='PUBLISHER/PLACE',
year int(4) xpath='DATEPUB')
engine=CONNECT table_type=XML file_name='Xsample.xml'
tabname='BIBLIO' option_list='rownode=BOOK,Expand=1,Mulnode=AUTHOR,Limit=2';
```

Before Connect 1.7.0002

```

create table xsampall2 (
    isbn char(15) field_format='@ISBN',
    language char(2) field_format='@LANG',
    subject char(32) field_format='@SUBJECT',
    authorfn char(20) field_format='AUTHOR/FIRSTNAME',
    authorln char(20) field_format='AUTHOR/LASTNAME',
    title char(32) field_format='TITLE',
    translated char(32) field_format='TRANSLATOR/@PREFIX',
    tranfn char(20) field_format='TRANSLATOR/FIRSTNAME',
    tranln char(20) field_format='TRANSLATOR/LASTNAME',
    publisher char(20) field_format='PUBLISHER/NAME',
    location char(20) field_format='PUBLISHER/PLACE',
    year int(4) field_format='DATEPUB'
engine=CONNECT table_type=XML file_name='Xsample.xml'
tablename='BIBLIO'
option_list='rownode=BOOK,Expand=1,Mulnode=AUTHOR,Limit=2';

```

The only difference is that the "multiple" node is an intermediate node in the path. The resulting table can be seen with a query such as:

```

select subject, language lang, title, authorfn first, authorln
last, year from xsampall2;

```

This query displays:

SUBJECT	LANG	TITLE	FIRST	LAST	YEAR
applications	fr	Construire une application XML	Jean-Christophe	Bernadac	1999
applications	fr	Construire une application XML	François	Knab	1999
applications	fr	XML en Action	William J.	Pardi	1999
général	fr	XML, Langage et Applications	Alain	Michard	1998

These composite tables, half array half tree, reserve some surprises for us when updating, deleting from or inserting into them. Insert just cannot generate this structure; if two rows are inserted with just a different author, two book nodes will be generated in the XML file. Delete always deletes one book node and all its children nodes even if specified against only one author. Update is more complicated:

```

update xsampall2 set authorfn = 'Simon' where authorln = 'Knab';
update xsampall2 set year = 2002 where authorln = 'Bernadac';
update xsampall2 set authorln = 'Mercier' where year = 2002;

```

After these three updates, the first two responding "Affected rows: 1" and the last one responding "Affected rows: 2", the last query answers:

subject	lang	title	first	last	year
applications	fr	Construire une application XML	Jean-Christophe	Mercier	2002
applications	fr	Construire une application XML	François	Knab	2002
applications	fr	XML en Action	William J.	Pardi	1999
général	fr	XML, Langage et Applications	Alain	Michard	1998

What must be understood here is that the Update modifies node values in the XML file, not cell values in the relational table. The first update worked normally. The second update changed the year value of the book and this shows for the two expanded rows because there is only one DATEPUB node for that book. Because the third update applies to a row having a certain date value, both author names were updated.

## Making a List of Multiple Values

Another way to see multiple values is to ask CONNECT to make a comma separated list of the multiple node values. This time, it can only be done if the "multiple" node is not intermediate. For example, we can modify the xsamp2 table definition by:

```

alter table xsamp2 option_list='rownode=BOOK,Mulnode=AUTHOR,Limit=3';

```

This time 'Expand' is not specified, and Limit gives the maximum number of items in the list. Now if we enter the query:

```

select isbn, subject, author "AUTHOR(S)", title from xsamp2;

```

We will get the following result:

ISBN	SUBJECT	AUTHOR(S)	TITLE
9782212090819	applications	Jean-Christophe Bernadac, François Knab	Construire une application XML

9782840825685	applications	William J. Pardi	XML en Action
9782212090529	général	Alain Michard	XML, Langage et Applications

Note that updating the "multiple" column is not possible because CONNECT does not know which of the nodes to update.

This could not have been done with the xsampall2 table because the author node is intermediate in the path, and making two lists, one of first names and another one of last names would not make sense anyway.

## What if a table contains several multiple nodes

This can be handled by creating several tables on the same file, each containing only one multiple node and constructing the desired result using joins.

## Support of HTML Tables

Most tables included in HTML documents cannot be processed by CONNECT because the HTML language is often not compatible with the syntax of XML. In particular, XML requires all open tags to be matched by a closing tag while it is sometimes optional in HTML. This is often the case concerning column tags.

However, you can meet tables that respect the XML syntax but have some of the features of HTML tables. For instance:

```
<?xml version="1.0"?>
<Beers>
  <table>
    <th><td>Name</td><td>Origin</td><td>Description</td></th>
    <tr>
      <td><brandName>Huntsman</brandName></td>
      <td><origin>Bath, UK</origin></td>
      <td><details>Wonderful hop, light alcohol</details></td>
    </tr>
    <tr>
      <td><brandName>Tuborg</brandName></td>
      <td><origin>Danmark</origin></td>
      <td><details>In small bottles</details></td>
    </tr>
  </table>
</Beers>
```

Here the different column tags are included in

<td></td>

tags as for HTML tables. You cannot just add this tag in the Xpath of the columns, because the search is done on the first occurrence of each tag, and this would cause this search to fail for all columns except the first one. This case is handled by specifying the *Colnode* table option that gives the name of these column tags, for example:

From Connect 1.7.0002

```
create table beers (
`Name` char(16) xpath='brandName',
`Origin` char(16) xpath='origin',
`Description` char(32) xpath='details')
engine=CONNECT table_type=XML file_name='beers.xml'
tabname='table' option_list='rownode=tr,colnode=td';
```

Before Connect 1.7.0002

```
create table beers (
`Name` char(16) field_format='brandName',
`Origin` char(16) field_format='origin',
`Description` char(32) field_format='details')
engine=CONNECT table_type=XML file_name='beers.xml'
tabname='table' option_list='rownode=tr,colnode=td';
```

The table will be displayed as:

Name	Origin	Description
Huntsman	Bath, UK	Wonderful hop, light alcohol
Tuborg	Danmark	In small bottles

However, you can deal with tables even closer to the HTML model. For example the *coffee.htm* file:

```

<TABLE summary="This table charts the number of cups of coffee consumed by each senator, the type of coffee (decaf or regular), and whether taken with sugar.">
  <CAPTION>Cups of coffee consumed by each senator</CAPTION>
  <TR>
    <TH>Name</TH>
    <TH>Cups</TH>
    <TH>Type of Coffee</TH>
    <TH>Sugar?</TH>
  </TR>
  <TR>
    <TD>T. Sexton</TD>
    <TD>10</TD>
    <TD>Espresso</TD>
    <TD>No</TD>
  </TR>
  <TR>
    <TD>J. Dinnen</TD>
    <TD>5</TD>
    <TD>Decaf</TD>
    <TD>Yes</TD>
  </TR>
</TABLE>

```

Here column values are directly represented by the TD tag text. You cannot declare them as tags nor as attributes. In addition, they are not located using their name but by their position within the row. Here is how to declare such a table to CONNECT:

```

create table coffee (
  `Name` char(16),
  `Cups` int(8),
  `Type` char(16),
  `Sugar` char(4))
engine=connect table_type=XML file_name='coffee.htm'
tabname='TABLE' header=1 option_list='Coltype=HTML';

```

You specify the fact that columns are located by position by setting the *Coltype* option to 'HTML'. Each column position (0 based) will be the value of the *flag* column parameter that is set by default in sequence. Now we are able to display the table:

Name	Cups	Type	Sugar
T. Sexton	10	Espresso	No
J. Dinnen	5	Decaf	Yes

**Note 1:** We specified '

```

header=1
' in the create statement to indicate that the first n rows of the table are not data rows and should be skipped.
```

**Note 2:** In this last example, we did not specify the node names using the Rownode and Colnode options because when *Coltype* is set to 'HTML' they default to '

```

Rownode=TR
' and '
Colnode=TD
'.
```

**Note 3:** The *Coltype* option is a word only the first character of which is significant. Recognized values are:

T(ag) or N(ode) Column names match a tag name (the default).

A(tribute) or @ Column names match an attribute name.

H(tml) or C(ol) or P(os) Column are retrieved by their position.

## New file setting

Some create options are used only when creating a table on a new file, i. e. when inserting into a file that does not exist yet. When specified, the 'Header' option will create a header row with the name of the table columns. This is chiefly useful for HTML tables to be displayed on a web browser.

Some new list-options are used in this context:

**Encoding** The encoding of the new document, defaulting to UTF-8.

**Attribute** A list of 'atname=attvalue' separated by ';' to add to the table node.

**HeadAttr** An attribute list to be added to the header row node.

Let us see for instance, the following create statement:

```

create table handlers (
  handler char(64),
  version char(20),
  author char(64),
  description char(255),
  maturity char(12))
engine=CONNECT table_type=XML file_name='handlers.htm'
tabname='TABLE' header=yes
option_list='coltype=HTML,encoding=ISO-8859-1,
attribute=border=1;cellpadding=5,headattr=bgcolor=yellow';

```

Supposing the table file does not exist yet, the first insert into that table, for instance by the following statement:

```

insert into handlers select plugin_name, plugin_version,
  plugin_author, plugin_description, plugin_maturity from
information_schema.plugins where plugin_type = 'DAEMON';

```

will generate the following file:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Created by CONNECT Version 3.05.0005 August 17, 2012 -->
<TABLE border="1" cellpadding="5">
  <TR bgcolor="yellow">
    <TH>handler</TH>
    <TH>version</TH>
    <TH>author</TH>
    <TH>description</TH>
    <TH>maturity</TH>
  </TR>
  <TR>
    <TD>Maria</TD>
    <TD>1.5</TD>
    <TD>Monty Program Ab</TD>
    <TD>Compatibility aliases for the Aria engine</TD>
    <TD>Gamma</TD>
  </TR>
</TABLE>

```

This file can be used to display the table on a web browser (encoding should be

```

ISO-8859-x
)
```

handler	version	author	description	maturity
Maria	1.5	Monty Program Ab	Compatibility aliases for the Aria engine	Gamma

**Note:** The XML document encoding is generally specified in the XML header node and can be different from the DATA\_CHARSET, which is always UTF-8 for XML tables. Therefore the table DATA\_CHARSET character set should be unspecified, or specified as UTF8. The Encoding specification is useful only for new XML files and ignored for existing files having their encoding already specified in the header node.

## Notes

- ↑ CONNECT does not claim to be able to deal with any XML document. Besides, those that can usefully be processed for data analysis are likely to have a structure that can easily be transformed into a table.
- ↑ With libxml2, sub tags text can be separated by 0 or several blanks depending on the structure and indentation of the data file.
- ↑ This may cause some rows to be lost because an eventual where clause on the “multiple” column is applied only on the limited number of retrieved rows.

## 4.3.7.6.14 CONNECT INI Table Type

### Contents

1. [Overview](#)
2. [Column layout](#)
3. [Row layout](#)

## Overview

The INI type is one of the configuration or initialization files often found on Windows machines. For instance, let us suppose you have the following contact file *contact.ini*:

```

[BER]
name=Bertrand
forename=Olivier
address=21 rue Ferdinand Buisson
city=Issy-les-Mlx
zipcode=92130
tel=09.54.36.29.60
cell=06.70.06.04.16

[WEL]
name=Schmitt
forename=Bernard
hired=19/02/1985
address=64 tiergarten strasse
city=Berlin
zipcode=95013
tel=03.43.377.360

[UK1]
name=Smith
forename=Henry
hired=08/11/2003
address=143 Blum Rd.
city=London
zipcode=NW1 2BP

```

CONNECT lets you view it as a table in two different ways.

## Column layout

The first way is to regard it as a table having one line per section, the columns being the keys you want to display. In this case, the CREATE statement could be:

```

create table contact (
    contact char(16) flag=1,
    name char(20),
    forename char(32),
    hired date date_format='DD/MM/YYYY',
    address char(64),
    city char(20),
    zipcode char(8),
    tel char(16))
engine=CONNECT table_type=INI file_name='contact.ini';

```

The column that will contain the section name can have any name but must specify

flag=1

. All other columns must have the names of the keys we want to display (case insensitive). The type can be character or numeric depending on the key value type, and the length is the maximum expected length for the key value. Once done, the statement:

```
select contact, name, hired, city, tel from contact;
```

This statement will display the file in tabular format.

contact	name	hired	city	tel
BER	Bertrand	1970-01-01	Issy-les-Mlx	09.54.36.29.60
WEL	Schmitt	1985-02-19	Berlin	03.43.377.360
UK1	Smith	2003-11-08	London	NULL

Only the keys defined in the create statements are visible; keys that do not exist in a section are displayed as null or pseudo null (blank for character, 1/1/70 for dates, and 0 for numeric) for columns declared NOT NULL.

All relational operations can be applied to this table. The table (and the file) can be updated, inserted and conditionally deleted. The only constraint is that when inserting values, the section name must be the first in the list of values.

**Note 1:** When inserting, if a section already exists, no new section will be created but the new values will be added or replace those of the existing section. Thus, the following two commands are equivalent:

```

update contact set forename = 'Harry' where contact = 'UK1';
insert into contact (contact,forename) values('UK1','Harry');

```

**Note 2:** Because sections represent one line, a DELETE statement on a section key will delete the whole section.

## Row layout

To be a good candidate for tabular representation, an INI file should have often the same keys in all sections. In practice, many files commonly found on computers, such as the *win.ini* file of the Windows directory or the *my.ini* file cannot be viewed that way because each section has different keys. In this case, a second way is to regard the file as a table having one row per section key and whose columns can be the section name, the key name, and the key value.

For instance, let us define the table:

```
create table xcont (
  section char(16) flag=1,
  keyname char(16) flag=2,
  value char(32))
engine=CONNECT table_type=INI file_name='contact.ini'
option_list='Layout=Row';
```

In this statement, the "Layout" option sets the display format, Column by default or anything else not beginning by 'C' for row layout display. The names of the three columns can be freely chosen. The Flag option gives the meaning of the column. Specify

flag=1  
for the section name and  
flag=2  
for the key name. Otherwise, the column will contain the key value.

Once done, the command:

```
select * from xcont;
```

Will display the following result:

section	keyname	value
BER	name	Bertrand
BER	forename	Olivier
BER	address	21 rue Ferdinand Buisson
BER	city	Issy-les-Mix
BER	zipcode	92130
BER	tel	09.54.36.29.60
BER	cell	06.70.06.04.16
WEL	name	Schmitt
WEL	forename	Bernard
WEL	hired	19/02/1985
WEL	address	64 tiergarten strasse
WEL	city	Berlin
WEL	zipcode	95013
WEL	tel	03.43.377.360
UK1	name	Smith
UK1	forename	Henry
UK1	hired	08/11/2003
UK1	address	143 Blum Rd.
UK1	city	London
UK1	zipcode	NW1 2BP

**Note:** When processing an INI table, all section names are retrieved in a buffer of 8K bytes (2048 bytes before 10.0.17). For a big file having many sections, this size can be increased using for example:

```
option_list='seclen=16K';
```

## 4.3.7.6.15 CONNECT - External Table Types

Because so many ODBC and JDBC drivers exist and only the main ones have been heavily tested, these table types cannot be ranked as stable. Use

them with care in production applications.

These types can be used to access tables belonging to the current or another database server. Six types are currently provided:

**ODBC** : To be used to access tables from a database management system providing an ODBC connector. ODBC is a standard of Microsoft and is currently available on Windows. On Linux, it can also be used provided a specific application emulating ODBC is installed. Currently only unixODBC is supported.

**JDBC** : To be used to access tables from a database management system providing a JDBC connector. JDBC is an Oracle standard implemented in Java and principally meant to be used by Java applications. Using it directly from C or C++ application seems to be almost impossible due to an Oracle bug still not fixed. However, this can be achieved using a Java wrapper class used as an interface between C++ and JDBC. On another hand, JDBC is available on all platforms and operating systems.

**Mongo** : To access MongoDB collections as tables via their MongoDB C Driver. Because this requires both MongoDB and the C Driver to be installed and operational, this table type is not currently available in binary distributions but only when compiling MariaDB from source.

**MySQL** : This type is the preferred way to access tables belonging to another MySQL or MariaDB server. It uses the MySQL API to access the external table. Even though this can be obtained using the FEDERATED(X) plugin, this specific type is used internally by CONNECT because it also makes it possible to access tables belonging to the current server.

**PROXY** : Internally used by some table types to access other tables from one table.

## External Table Specification

The four main external table types – odbc, jdbc, mongo and mysql – are specified giving the following information:

1. The data source. This is specified in the connection option.
2. The remote table or view to access. This can be specified within the connection string or using specific CONNECT options.
3. The column definitions. This can be also left to CONNECT to find them using the discovery MariaDB feature.
4. The optional Quoted option. Can be set to 1 to quote the identifiers in the query sent to the remote server. This is required if columns or table names can contain blanks.

The way this works is by establishing a connection to the external data source and by sending it an SQL statement (or its equivalent using API functions for MONGO) enabling it to execute the original query. To enhance performance, it is necessary to have the remote data source do the maximum processing. This is needed in particular to reduce the amount of data returned by the data source.

This is why, for SELECT queries, CONNECT uses the `cond_push` MariaDB feature to retrieve the maximum of the where clause of the original query that can be added to the query sent to the data source. This is automatic and does not require anything to be done by the user.

However, more can be done. In addition to accessing a remote table, CONNECT offers the possibility to specify what the remote server must do. This is done by specifying it as a view in the srcdef option. For example:

```
CREATE TABLE custnum ENGINE=CONNECT TABLE_TYPE=XXX
  CONNECTION='connection string'
  SRCDEF='select pays as country, count(*) as customers from custnum group by pays';
```

Doing so, the group by clause will be done by the remote server considerably reducing the amount of data sent back on the connection.

This may even be increased by adding to the srcdef part of the “compatible” part of the query where clauses like this are done for table-based tables. Note that for MariaDB, this table has two columns, country and customers. Supposing the original query is:

```
SELECT * FROM custnum WHERE (country = 'UK' OR country = 'USA') AND customers > 5;
```

How can we make the where clause be added to the sent srcdef? There are many problems:

1. Where to include the additional information.
2. What about the use of alias.
3. How to know what will be a where clause or a having clause.

The first problem is solved by preparing the srcdef view to receive clauses. The above example srcdef becomes:

```
SRCDEF='select pays as country, count(*) as customers from custnum where %s group by pays having %s';
```

The %s in the srcdef are place holders for eventual compatible parts of the original query where clause. If the select query does not specify a where clause, or gives an unacceptable where clause, place holders will be filled by dummy clauses (1=1).

The other problems must be solved by adding to the create table a list of columns that must be translated because they are aliases or/and aliases on aggregate functions that must become a having clause. For example, in this case:

```
CREATE TABLE custnum ENGINE=CONNECT TABLE_TYPE=XXX
  CONNECTION='connection string'
  SRCDEF='select pays as country, count(*) as customers from custnum where %s group by pays having %s'
  OPTION_LIST='Alias=customers=*count(*);country=pays';
```

This is specified by the alias option, to be used in the option list. It is made of a semi-colon separated list of items containing:

1. The local column name (alias in the remote server)
2. An equal sign.

3. An eventual '\*' indicating this is column correspond to an aggregate function.
4. The remote column name.

With this information, CONNECT will be able to make the query sent to the remote data source:

```
select pays as country, count(*) as customers from custnum where (pays = 'UK' OR pays = 'USA') group by country having count(*)
```

Note: Some data sources, including MySQL and MariaDB, accept aliases in the having clause. In that case, the alias option could have been specified as:

```
OPTION_LIST='Alias=customers=*;country=pays';
```

Another option exists, phpos, enabling to specify what place holders are present and in what order. To be specified as "W", "WH", "H", or "HW". It is rarely used because by default CONNECT can set it from the srcdef content. The only cases it is needed is when the srcdef contains only a having place holder or when the having place holder occurs before the where place holder, which can occur on queries containing joins. CONNECT cannot handle more than one place holder of each type.

SRCDEF is not available for MONGO tables, but other ways of achieving this exist and are described in the MONGO table type chapter.

## 4.3.7.6.16 CONNECT ODBC Table Type: Accessing Tables From Another DBMS

### Contents

1. [Random Access of ODBC Tables](#)
2. [Retrieving data from a spreadsheet](#)
3. [Multiple ODBC tables](#)
4. [Performance consideration](#)
5. [Using ODBC Tables inside correlated sub-queries](#)
6. [Accessing specified views](#)
7. [Data Modifying Operations](#)
  1. [INSERT Command](#)
  2. [UPDATE and DELETE Commands](#)
8. [Sending commands to a Data Source](#)
  1. [Sending several commands together](#)
9. [Connecting to a Data Source](#)
  1. [Defining the Connection String](#)
  2. [ODBC Defined Connection Attributes](#)
  3. [Using a Predefined DSN](#)
10. [ODBC Tables on Linux/Unix](#)
  1. [SELinux](#)
11. [ODBC Catalog Information](#)
  1. [Table name case](#)
12. [Non-ASCII Character Sets with Oracle](#)
  1. [Using systemd](#)
  2. [Using Windows](#)

ODBC (Open Database Connectivity) is a standard API for accessing database management systems (DBMS). CONNECT uses this API to access data contained in other DBMS without having to implement a specific application for each one. An exception is the access to MySQL that should be done using the [MYSQL table type](#).

**Note:** On Linux, unixODBC must be installed.

These tables are given the type ODBC. For example, if a "Customers" table is contained in an Access ™ database you can define it with a command such as:

```

create table Customer (
    CustomerID varchar(5),
    CompanyName varchar(40),
    ContactName varchar(30),
    ContactTitle varchar(30),
    Address varchar(60),
    City varchar(15),
    Region varchar(15),
    PostalCode varchar(10),
    Country varchar(15),
    Phone varchar(24),
    Fax varchar(24))
engine=connect table_type=ODBC block_size=10
tabname='Customers'
Connection='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB';

```

Tabname option defaults to the table name. It is required if the source table name is different from the name of the CONNECT table. Note also that for some data sources this name is case sensitive.

Often, because CONNECT can retrieve the table description using ODBC catalog functions, the column definitions can be unspecified. For instance this table can be simply created as:

```

create table Customer engine=connect table_type=ODBC
block_size=10 tabname='Customers'
Connection='DSN=MS Access Database;DBQ=C:/Program Files/Microsoft Office/Office/1033/FPNWIND.MDB';

```

The

BLOCK\_SIZE

specification will be used later to set the RowsetSize when retrieving rows from the ODBC table. A reasonably large RowsetSize can greatly accelerate the fetching process.

If you specify the column description, the column names of your table must exist in the data source table. However, you are not obliged to define all the data source columns and you can change the order of the columns. Some type conversion can also be done if appropriate. For instance, to access the FireBird sample table EMPLOYEE, you could define your table as:

```

create table empodbc (
    EMP_NO smallint(5) not null,
    FULL_NAME varchar(37) not null,
    PHONE_EXT varchar(4) not null,
    HIRE_DATE date,
    DEPT_NO smallint(3) not null,
    JOB_COUNTRY varchar(15),
    SALARY double(12,2) not null)
engine=CONNECT table_type=ODBC tabname='EMPLOYEE'
connection='DSN=firebird';

```

This definition ignores the FIRST\_NAME, LAST\_NAME, JOB\_CODE, and JOB\_GRADE columns. It places the FULL\_NAME last column of the original table in second position. The type of the HIRE\_DATE column was changed from *timestamp* to *date* and the type of the DEPT\_NO column was changed from *char* to *integer*.

Currently, some restrictions apply to ODBC tables:

1. Cursor type is forward only (sequential reading).
2. No indexing of ODBC tables (do not specify any columns as key). However, because CONNECT can often add a where clause to the query sent to the data source, indexing will be used by the data source if it supports it. (Remote indexing is available with version 1.04, released with [MariaDB 10.1.6](#) )
3. CONNECT ODBC supports [SELECT](#) and [INSERT](#) . [UPDATE](#) and [DELETE](#) are also supported in a somewhat restricted way (see below). For other operations, use an ODBC table with the EXECSRC option (see below) to directly send proper commands to the data source.

## Random Access of ODBC Tables

In CONNECT version 1.03 (until [MariaDB 10.1.5](#) ) ODBC tables are not indexable. Version 1.04 (from [MariaDB 10.1.6](#) ) adds remote indexing facility to the ODBC table type.

However, some queries require random access to an ODBC table; for instance when it is joined to another table or used in an order by queries applied to a long column or large tables.

There are several ways to enable random (position) access to a CONNECT ODBC table. They are dependant on the following table options:

Option	Type	Used For
Block_Size	Integer	Specifying the rowset size.
Memory*	Integer	Storing the result set in memory.

Scorable*	Boolean	Using a scrollable cursor.
-----------	---------	----------------------------

\*

- To be specified in the option\_list.

When dealing with small tables, the simpler way to enable random access is to specify a rowset size equal or larger than the table size (or the result set size if a push down where clause is used). This means that the whole result is in memory on the first fetch and CONNECT will use it for further positional accesses.

Another way to have the result set in memory is to use the memory option. This option can be set to the following values:

0. No memory used (the default). Best when the table is read sequentially as in SELECT statements with only eventual WHERE clauses.
1. Memory size required is calculated during the first sequential table read. The allocated memory is filled during the second sequential read. Then the table rows are retrieved from the memory. This should be used when the table will be accessed several times randomly, such as in sub-selects or being the target table of a join.
2. A first query is executed to get the result set size and the needed memory is allocated. It is filled on the first sequential reading. Then random access of the table is possible. This can be used in the case of ORDER BY clauses, when MariaDB uses position reading.

Note that the best way to handle ORDER BY is to set the max\_length\_for\_sort\_data variable to a larger value (its default value is 1024 that is pretty small). Indeed, it requires less memory to be used, particularly when a WHERE clause limits the retrieved data set. This is because in the case of an order by query, MariaDB firstly retrieves the sequentially the result set and the position of each records. Often the sort can be done from the result set if it is not too big. But if too big, or if it implies some "long" columns, only the positions are sorted and MariaDB retrieves the final result from the table read in random order. If setting the max\_length\_for\_sort\_data variable is not feasible or does not work, to be able to retrieve table data from memory after the first sequential read, the memory option must be set to 2.

For tables too large to be stored in memory another possibility is to make your table to use a scrollable cursor. In this case each randomly accessed row can be retrieved from the data source specifying its cursor position, which is reasonably fast. However, scrollable cursors are not supported by all data sources.

With CONNECT version 1.04 (from [MariaDB 10.1.6](#) ), another way to provide random access is to specify some columns to be indexed. This should be done only when the corresponding column of the source table is also indexed. This should be used for tables too large to be stored in memory and is similar to the remote indexing used by the [MYSQL table type](#) and by the [FEDERATED engine](#) .

There remains the possibility to extract data from the external table and to construct another table of any file format from the data source. For instance to construct a fixed formatted DOS table containing the CUSTOMER table data, create the table as

```
create table Custfix engine=connect File_name='customer.txt'
table_type=fix block_size=20 as select * from customer;
```

Now you can use *custfix* for fast database operations on the copied *customer* table data.

## Retrieving data from a spreadsheet

ODBC can also be used to create tables based on tabular data belonging to an Excel spreadsheet:

```
create table XLCNT
engine=CONNECT table_type=ODBC tablename='CONTACT'
Connection='DSN=Excel Files;DBQ=D:/Ber/Doc/Contact_BP.xls';
```

This supposes that a tabular zone of the sheet including column headers is defined as a table named CONTACT or using a "named reference". Refer to the Excel documentation for how to specify tables inside sheets. Once done, you can ask:

```
select * from xlcont;
```

This will extract the data from Excel and display:

Nom	Fonction	Societe
Boisseau Frederic		9 Telecom
Martelliere Nicolas		Vidal SA (Groupe UBM)
Remy Agathe		Price Minister
Du Halgouet Tanguy		Danone
Vandamme Anna		GDF
Thomas Willy		Europ Assistance France
Thomas Dominique		Acoss (DG des URSSAF)
Thomas Berengere	Responsable SI Decisionnel	DEXIA Credit Local
Husy Frederic	Responsable Decisionnel	Neuf Cegetel

Lemonnier Nathalie	Directeur Marketing Client	Louis Vuitton
Louis Loic	Reporting International Decisionnel	Accor
Menseau Eric		Orange France

Here again, the columns description was left to CONNECT when creating the table.

## Multiple ODBC tables

The concept of multiple tables can be extended to ODBC tables when they are physically represented by files, for instance to Excel or Access tables. The condition is that the connect string for the table must contain a field DBQ= *filename* , in which wildcard characters can be included as for multiple=1 tables in their filename. For instance, a table contained in several Excel files CA200401.xls, CA200402.xls, ...CA200412.xls can be created by a command such as:

```
create table ca04mul (Date char(19), Operation varchar(64),
  Debit double(15,2), Credit double(15,2)
engine=CONNECT table_type=ODBC multiple=1
qchar= '"' tabname='bank account'
connection='DSN=Excel Files;DBQ=D:/Ber/CA/CA2004*.xls';'
```

Providing that in each file the applying information is internally set for Excel as a table named "bank account". This extension to ODBC does not support *multiple* =2. The *qchar* option was specified to make the identifiers quoted in the select statement sent to ODBC, in particular the when the table or column names contain blanks, to avoid SQL syntax errors.

**Caution:** Avoid accessing tables belonging to the currently running MariaDB server via the MySQL ODBC connector. This may not work and may cause the server to be restarted.

## Performance consideration

To avoid extracting entire tables from an ODBC source, which can be a lengthy process, CONNECT extracts the "compatible" part of query WHERE clauses and adds it to the ODBC query. Compatible means that it must be understood by the data source. In particular, clauses involving scalar functions are not kept because the data source may have different functions than MariaDB or use a different syntax. Of course, clauses involving sub-select are also skipped. This will transfer eventual indexing to the data source.

Take care with clauses involving string items because you may not know whether they are treated by the data source as case sensitive or case insensitive. If in doubt, make your queries as if the data source was processing strings as case sensitive to avoid incomplete results.

## Using ODBC Tables inside correlated sub-queries

Unlike not correlated subqueries that are executed only once, correlated subqueries are executed many times. It is what ODBC calls a "requery". Several methods can be used by CONNECT to deal with this depending on the setting of the MEMORY or SCROLLABLE Boolean options:

Option	Description
Default	Implementing "requery" by discarding the current result set and re submitting the query (as MFC does)
Memory=1 or 2	Storing the result set in memory as MYSQL tables do.
Scrollable=Yes	Using a scrollable cursor.

Note: the MEMORY and SCROLLABLE options must be specified in the OPTION \_ LIST.

Because the table is accessed several times, this can make queries last very long except for small tables and is almost unacceptable for big tables. However, if it cannot be avoided, using the memory method is the best choice and can be more than four times faster than the default method. If it is supported by the driver, using a scrollable cursor is slightly slower than using memory but can be an alternative to avoid memory problems when the sub-query returns a huge result set.

If the result set is of reasonable size, it is also possible to specify the block\_size option equal or slightly larger than the result set. The whole result set being read on the first fetch, can be accessed many times without having to do anything else.

Another good workaround is to replace within the correlated sub-query the ODBC table by a local copy of it because MariaDB is often able to optimize the query and to provide a very fast execution.

## Accessing specified views

Instead of specifying a source table name via the TABNAME option, it is possible to retrieve data from a "view" whose definition is given in a new option SRCDEF. For instance:

```

CREATE TABLE custnum (
    country varchar(15) NOT NULL,
    customers int(6) NOT NULL
)
ENGINE=CONNECT TABLE_TYPE=ODBC BLOCK_SIZE=10
CONNECTION='DSN=MS Access Database;DBQ=C:/Program Files/Microsoft Office/Office/1033/FPNWIND.MDB;'
SRCDEF='select country, count(*) as customers from customers group by country';

```

Or simply, because CONNECT can retrieve the returned column definition:

```

CREATE TABLE custnum ENGINE=CONNECT TABLE_TYPE=ODBC BLOCK_SIZE=10
CONNECTION='DSN=MS Access Database;DBQ=C:/Program Files/Microsoft Office/Office/1033/FPNWIND.MDB;'
SRCDEF='select country, count(*) as customers from customers group by country';

```

Then, when executing for instance:

```
select * from custnum where customers > 3;
```

The processing of the group by is done by the data source, which returns only the generated result set on which only the where clause is performed locally. The result:

country	customers
Brazil	9
France	11
Germany	11
Mexico	5
Spain	5
UK	7
USA	13
Venezuela	4

This makes possible to let the data source do complicated operations, such as joining several tables or executing procedures returning a result set. This minimizes the data transfer through ODBC.

## Data Modifying Operations

The only data modifying operations are the [INSERT](#), [UPDATE](#) and [DELETE](#) commands. They can be executed successfully only if the data source database or tables are not read-only.

### INSERT Command

When inserting values to an ODBC table, local values are used and sent to the ODBC table. This does not make any difference when the values are constant but in a query such as:

```
insert into t1 select * from t2;
```

Where t1 is an ODBC table, t2 is a locally defined table that must exist on the local server. Besides, it is a good way to create a distant ODBC table from local data.

CONNECT does not directly support INSERT commands such as:

```
insert into t1 values(2,'Deux') on duplicate key update msg = 'Two';
```

Sure enough, the "on duplicate key update" part of it is ignored, and will result in error if the key value is duplicated.

### UPDATE and DELETE Commands

Unlike the [INSERT](#) command, [UPDATE](#) and [DELETE](#) are supported in a simplified way. Only simple table commands are supported; CONNECT does not support multi-table commands, commands sent from a procedure, or issued via a trigger. These commands are just rephrased to correspond to the data source syntax and sent to the data source for execution. Let us suppose we created the table:

```

create table tolite (
    id int(9) not null,
    nom varchar(12) not null,
    nais date default null,
    rem varchar(32) default null)
ENGINE=CONNECT TABLE_TYPE=ODBC tablename='lite'
CONNECTION='DSN=SQLite3 Datasource;Database=test.sqlite3'
CHARSET=utf8 DATA_CHARSET=utf8;

```

We can populate it by:

```

insert into tolite values(1,'Toto',now(),'First'),
(2,'Foo','2012-07-14','Second'),(4,'Machin','1968-05-30','Third');

```

The function

```
now()
```

will be executed by MariaDB and it returned value sent to the ODBC table.

Let us see what happens when updating the table. If we use the query:

```
update tolite set nom = 'Gillespie' where id = 10;
```

CONNECT will rephrase the command as:

```
update lite set nom = 'Gillespie' where id = 10;
```

What it did is just to replace the local table name with the remote table name and change all the back ticks to blanks or to the data source identifier quoting characters if QUOTED is specified. Then this command will be sent to the data source to be executed by it.

This is simpler and can be faster than doing a positional update using a cursor and commands such as “select ... for update of ...” that are not supported by all data sources. However, there are some restrictions that must be understood due to the way it is handled by MariaDB.

1. MariaDB does not know about all the above. The command will be parsed as if it were to be executed locally. Therefore, it must respect the MariaDB syntax.
2. Being executed by the data source, the (rephrased) command must also respect the data source syntax.
3. All data referenced in the SET and WHERE clause belongs to the data source.

This is possible because both MariaDB and the data source are using the SQL language. But you must use only the basic features that are part of the core SQL language. For instance, keywords like IGNORE or LOW\_PRIORITY will cause syntax error with many data source.

Scalar function names also can be different, which severely restrict the use of them. For instance:

```
update tolite set nais = now() where id = 2;
```

This will not work with SQLite3, the data source returning an “unknown scalar function” error message. Note that in this particular case, you can rephrase it to:

```
update tolite set nais = date('now') where id = 2;
```

This understood by both parsers, and even if this function would return NULL executed by MariaDB, it does return the current date when executed by SQLite3. But this begins to become too trickery so to overcome all these restrictions, and permit to have all types of commands executed by the data source, CONNECT provides a specific ODBC table subtype described now.

## Sending commands to a Data Source

This can be done using a special subtype of ODBC table. Let us see this in an example:

```

create table crlrite (
    command varchar(128) not null,
    number int(5) not null flag=1,
    message varchar(255) flag=2)
engine=connect table_type=odbc
connection='Driver=SQLite3 ODBC Driver;Database=test.sqlite3;NoWCHAR=yes'
option_list='Execsrc=1';

```

The key points in this create statement are the EXECSRC option and the column definition.

The EXECSRC option tells that this table will be used to send a command to the data source. Most of the sent commands do not return result set. Therefore, the table columns are used to specify the command to be executed and to get the result of the execution. The name of these columns can be chosen arbitrarily, their function coming from the FLAG value:

Flag=0: The command to execute.

Flag=1: The affected rows, or -1 in case of error, or the result number of column if the command returns a result set.

Flag=2: The returned (eventually error) message.

How to use this table and specify the command to send? By executing a command such as:

```
select * from crlite where command = 'a command';
```

This will send the command specified in the WHERE clause to the data source and return the result of its execution. The syntax of the WHERE clause must be exactly as shown above. For instance:

```
select * from crlite where command =
'CREATE TABLE lite (
ID integer primary key autoincrement,
name char(12) not null,
birth date,
rem varchar(32))';
```

This command returns:

command	number	message
CREATE TABLE lite (ID integer primary key autoincrement, name... 0	0	Affected rows

Now we can create a standard ODBC table on the newly created table:

```
CREATE TABLE tlite
ENGINE=CONNECT TABLE_TYPE=ODBC tablename='lite'
CONNECTION='Driver=SQLite3 ODBC Driver;Database=test.sqlite3;NoWCHAR=yes'
CHARSET=utf8 DATA_CHARSET=utf8;
```

We can populate it directly using the supported [INSERT](#) statement:

```
insert into tlite(name,birth) values('Toto','2005-06-12');
insert into tlite(name,birth,rem) values('Foo',NULL,'No ID');
insert into tlite(name,birth) values('Truc','1998-10-27');
insert into tlite(name,birth,rem) values('John','1968-05-30','Last');
```

And see the result:

```
select * from tlite;
```

ID	name	birth	rem
1	Toto	2005-06-12	NULL
2	Foo	NULL	No ID
3	Truc	1998-10-27	NULL
4	John	1968-05-30	Last

Any command, for instance [UPDATE](#), can be executed from the *crlite* table:

```
select * from crlite where command =
'update lite set birth = ''2012-07-14'' where ID = 2';
```

This command returns:

command	number	message
update lite set birth = '2012-07-15' where ID = 2	1	Affected rows

Let us verify it:

```
select * from tlite where ID = 2;
```

ID	name	birth	rem
2		2012-07-15	

2	Foo	2012-07-15	No ID
---	-----	------------	-------

The syntax to send a command is rather strange and may seem unnatural. It is possible to use an easier syntax by defining a stored procedure such as:

```
create procedure send_cmd(cmd varchar(255))
MODIFIES SQL DATA
select * from crlite where command = cmd;
```

Now you can send commands like this:

```
call send_cmd('drop tlite');
```

This is possible only when sending one single command.

## Sending several commands together

Grouping commands uses an easier syntax and is faster because only one connection is made for all of them. To send several commands in one call, use the following syntax:

```
select * from crlite where command in (
  'update lite set birth = ''2012-07-14'' where ID = 2',
  'update lite set birth = ''2009-08-10'' where ID = 3');
```

When several commands are sent, the execution stops at the end of them or after a command that is in error. To continue after *n* errors, set the option maxerr= *n* (0 by default) in the option list.

**Note 1:** It is possible to specify the SRCDEF option when creating an EXECSRC table. It will be the command sent by default when a WHERE clause is not specified.

**Note 2:** Most data sources do not allow sending several commands separated by semi-colons.

**Note 3:** Quotes inside commands must be escaped. This can be avoided by using a different quoting character than the one used in the command

**Note 4:** The sent command must obey the data source syntax.

**Note 5:** Sent commands apply in the specified database. However, they can address any table within this database, or belonging to another database using the name syntax *schema.tablename*.

## Connecting to a Data Source

There are two ways to establish a connection to a data source:

1. Using SQLDriverConnect and a Connection String
2. Using SQLConnect and a Data Source Name (DSN)

The first way uses a Connection String whose components describe what is needed to establish the connection. It is the most complete way to do it and by default CONNECT uses it.

The second way is a simplified way in which ODBC is just given the name of a DSN that must have been defined to ODBC or UnixOdbc and that contains the necessary information to establish the connection. Only the user name and password can be specified out of the DSN specification.

## Defining the Connection String

Using the first way, the connection string must be specified. This is sometimes the most difficult task when creating ODBC tables because, depending on the operating system and the data source, this string can widely differ.

The format of the ODBC Connection String is:

```
connection-string ::= empty-string[] | attribute[] | attribute; connection-string
empty-string ::=
attribute ::= attribute-keyword=attribute-value | DRIVER=[{}attribute-value[]]
attribute-keyword ::= DSN | UID | PWD | driver-defined-attribute-keyword
attribute-value ::= character-string
driver-defined-attribute-keyword = identifier
```

Where character-string has zero or more characters; identifier has one or more characters; attribute-keyword is not case-sensitive; attribute-value may be case-sensitive; and the value of the DSN keyword does not consist solely of blanks. Due to the connection string grammar, keywords and attribute values that contain the characters

[{}();,;?\*=!@]

should be avoided. The value of the DSN keyword cannot consist only of blanks, and should not contain leading blanks. Because of the grammar of the system information, keywords and data source names cannot contain the backslash (\) character. Applications do not have to add braces around the attribute value after the DRIVER keyword unless the attribute contains a semicolon (;), in which case the braces are required. If the attribute value that the driver receives includes the braces, the driver should not remove them, but they should be part of the returned connection string.

## ODBC Defined Connection Attributes

The ODBC defined attributes are:

- DSN - the name of the data source to connect to. You must create this before attempting to refer to it. You create new DSNs through the ODBC Administrator (Windows), ODBCAdmin (unixODBC's GUI manager) or in the odbc.ini file.
- DRIVER - the name of the driver to connect to. You can use this in DSN-less connections.
- FILEDSN - the name of a file containing the connection attributes.
- UID/PWD - any username and password the database requires for authentication.
- SAVEFILE - request the DSN attributes are saved in this file.

Other attributes are DSN dependent attributes. The connection string can give the name of the driver in the DRIVER field or the data source in the DSN field (attention! meet the spelling and case) and has other fields that depend on the data source. When specifying a file, the DBQ field must give the full path and name of the file containing the table. Refer to the specific ODBC connector documentation for the exact syntax of the connection string.

## Using a Predefined DSN

This is done by specifying in the option list the Boolean option "UseDSN" as yes or 1. In addition, string options "user" and "password" can be optionally specified in the option list.

When doing so, the connection string just contains the name of the predefined Data Source. For instance:

```
CREATE TABLE tlite ENGINE=CONNECT TABLE_TYPE=ODBC tablename='lite'  
CONNECTION='SQLite3 Datasource'  
OPTION_LIST='UseDSN=Yes,User=me,Password=mypass';
```

Note: the connection data source name (limited to 32 characters) should not be preceded by "DSN=".

## ODBC Tables on Linux/Unix

In order to use ODBC tables, you will need to have unixODBC installed. Additionally, you will need the ODBC driver for your foreign server's protocol. For example, for MS SQL Server or Sybase, you will need to have FreeTDS installed.

Make sure the user running mysqld (usually the mysql user) has permission to the ODBC data source configuration and the ODBC drivers. If you get an error on Linux/Unix when using TABLE\_TYPE=ODBC:

```
Error Code: 1105 [unixODBC][Driver Manager]Can't open lib  
'/usr/cachesys/bin/libcacheodbc.so' : file not found
```

You must make sure that the user running mysqld (usually "mysql") has enough permission to load the ODBC driver library. It can happen that the driver file does not have enough read privileges (use chmod to fix this), or loading is prevented by SELinux configuration (see below).

Try this command in a shell to check if the driver had enough permission:

```
sudo -u mysql ldd /usr/cachesys/bin/libcacheodbc.so
```

### SELinux

SELinux can cause various problems. If you think SELinux is causing problems, check the system log (e.g. /var/log/messages) or the audit log (e.g. /var/log/audit/audit.log).

**mysqld can't load some executable code, so it can't use the ODBC driver.**

Example error:

```
Error Code: 1105 [unixODBC][Driver Manager]Can't open lib  
'/usr/cachesys/bin/libcacheodbc.so' : file not found
```

Audit log:

```
type=AVC msg=audit(1384890085.406:76): avc: denied { execute }  
for pid=1433 comm="mysqld"  
path="/usr/cachesys/bin/libcacheodbc.so" dev=dm-0 ino=3279212  
scontext=unconfined_u:system_r:mysqld_t:s0  
tcontext=unconfined_u:object_r:usr_t:s0 tclass=file
```

**mysqld can't open TCP sockets on some ports, so it can't connect to the foreign server.**

Example error:

```
ERROR 1296 (HY000): Got error 174 '[unixODBC][FreeTDS][SQL Server]Unable to connect to data source' from CONNECT
```

Audit log:

```
type=AVC msg=audit(1423094175.109:433): avc: denied { name_connect } for pid=3193 comm="mysqld" dest=1433 scontext=system_u:object_r:mysqld_db_t:s0 tcontext=unconfined_u:object_r:mysqld_db_t:s0
```

## ODBC Catalog Information

Depending on the version of the used ODBC driver, some additional information on the tables are existing, such as table QUALIFIER or OWNER for old versions, now named CATALOG or SCHEMA since version 3.

CATALOG is apparently rarely used by most data sources, but SCHEMA (formerly OWNER) is and corresponds to the DATABASE information of MySQL.

The issue is that if no schema name is specified, some data sources return information for all schemas while some others only return the information of the "default" schema. In addition, the used "schema" or "database" is sometimes implied by the connection string and sometimes is not. Sometimes, it also can be included in a data source definition.

CONNECT offers two ways to specify this information:

1. When specified, the DBNAME create table option is regarded by ODBC tables as the SCHEMA name.
2. Table names can be specified as "*cat.sch.tab*" allowing to set the catalog and schema info.

When both are used, the qualified table name has precedence over DBNAME . For instance:

Tabname	DBname	Description
test.t1		The t1 table of the test schema.
test.t1	mydb	The t1 table of the test schema (test has precedence)
t1	mydb	The t1 table of the mydb schema
%.%.%		All tables in all catalogs and all schemas
t1		The t1 table in the default or all schema depending on the DSN
%.t1		The t1 table in all schemas for all DSN
test.%		All tables in the test schema

When creating a standard ODBC table, you should make sure only one source table is specified. Specifying more than one source table must be done only for CONNECT catalog tables (with CATFUNC=tables or columns).

In particular, when column definition is left to the Discovery feature, if tables with the same name are present in several schemas and the schema name is not specified, several columns with the same name will be generated. This will make the creation fail with a not very explicit error message.

Note: With some ODBC drivers, the DBNAME option or qualified table name is useless because the schema implied by the connection string or the definition of the data source has priority over the specified DBNAME .

## Table name case

Another issue when dealing with ODBC tables is the way table and column names are handled regarding of the case.

For instance, Oracle follows to the SQL standard here. It converts non-quoted identifiers to upper case. This is correct and expected. PostgreSQL is not standard. It converts identifiers to lower case. MySQL/MariaDB is not standard. They preserve identifiers on Linux, and convert to lower case on Windows.

Think about that if you fail to see a table or a column on an ODBC data source.

## Non-ASCII Character Sets with Oracle

When connecting through ODBC, the MariaDB Server operates as a client to the foreign database management system. As such, it requires that you configure MariaDB as you would configure native clients for the given database server.

In the case of connecting to Oracle, when using non-ASCII character sets, you need to properly set the NLS\_LANG environment variable before starting the MariaDB Server.

For instance, to test this on Oracle, create a table that contains a series of special characters:

```

CREATE TABLE t1 (letter VARCHAR(4000));

INSERT INTO t1 VALUES
(UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('C4'))),
(UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('C5'))),
(UTL_RAW.CAST_TO_VARCHAR2(HEXTORAW('C6')));

SELECT letter, RAWTOHEX(letter) FROM t1;

letter | RAWTOHEX(letter)
-----+-----
A      | C4
A      | C5
A      | C6

```

Then create a connecting table on MariaDB and attempt the same query:

```

CREATE TABLE t1 (
    letter VARCHAR(4000)
ENGINE=CONNECT
DEFAULT CHARSET=utf8mb4
CONNECTION='DSN=YOUR_DSN'
TABLE_TYPE = 'ODBC'
DATA_CHARSET = latin1
TABNAME = 'YOUR_SCHEMA.T1';

SELECT letter, HEX(letter) FROM t1;

+-----+-----+
| letter | HEX(letter) |
+-----+-----+
| A      |      41 |
| ?      |      3F |
| ?      |      3F |
+-----+-----+

```

While the character set is defined in a way that satisfies MariaDB, it has not been defined for Oracle, (that is, setting the NLS\_LANG environment variable). As a result, Oracle is not providing the characters you want to MariaDB and Connect. The specific method of setting the NLS\_LANG variable can vary depending on your operating system or distribution. If you're experiencing this issue, check your OS documentation for more details on how to properly set environment variables.

## Using systemd

With Linux distributions that use `systemd`, you need to set the environment variable in the service file, (systemd doesn't read from the `/etc/environment` file).

This is done by setting the Environment variable in the [Service] unit. For instance,

```
# vi /usr/lib/systemd/system/mariadb.service

[Service]
...
Environment=NLS_LANG=GERMAN_GERMANY.WE8ISO8859P1
```

Once this is done, reload the systemd units:

```
# systemctl daemon-reload
```

Then restart MariaDB,

```
# systemctl restart mariadb.service
```

You can now retrieve the appropriate characters from Oracle tables:

```
SELECT letter, HEX(letter) FROM t1;
```

letter	HEX(letter)
À	C384
Á	C385
È	C386

## Using Windows

Microsoft Windows doesn't ignore environment variables the way systemd does on Linux, but it does require that you set the NLS\_LANG environment variable on your system. In order to do so, you need to open an elevated command-prompt, (that is, Cmd.exe with administrative privileges).

From here, you can use the Setx command to set the variable. For instance,

```
Setx NLS_LANG GERMAN_GERMANY.WE8ISO8859P1 /m
```

Note: For more detail about this, see [MDEV-17501](#).

## 4.3.7.6.17 CONNECT JDBC Table Type: Accessing Tables from Another DBMS

### Contents

- 1. [Compiling From Source Distribution](#)
  - 1. [Compiling the Java source files](#)
- 2. [Setting the Required Information](#)
  - 1. [JVM Library Location](#)
  - 2. [Java Class Path](#)
- 3. [CONNECT JDBC Tables](#)
  - 1. [Using a Federated Server](#)
- 4. [Connecting to a JDBC driver](#)
  - 1. [Fetch Size](#)
  - 2. [Connection to a Data Source](#)
- 5. [Random Access to JDBC Tables](#)
- 6. [Other Operations with JDBC Tables](#)
- 7. [JDBC Specific Restrictions](#)
- 8. [Handling the UUID Data Type](#)
- 9. [Executing the JDBC tests](#)
- 10. [Fixing Problem With mysqldump](#)

The JDBC table type should be distributed with all recent versions of MariaDB. However, if the automatic compilation of it is possible after the java JDK was installed, the complete distribution of it is not fully implemented in older versions. The distributed JdbcInterface.jar file contains the JdbcInterface wrapper only. New versions distribute a JavaWrappers.jar that contains all currently existing wrappers.

This will require that:

1. The Java SDK is installed on your system.
2. The java wrapper class files are available on your system.
3. And of course, some JDBC drivers exist to be used with the matching DBMS.

Point 2 was made automatic in the newest versions of MariaDB.

## Compiling From Source Distribution

Even when the Java JDK has been installed, CMake sometimes cannot find the location where it stands. For instance on Linux the Oracle Java JDK package might be installed in a path not known by the CMake lookup functions causing error message such as:

```
CMake Error at /usr/share/cmake/Modules/FindPackageHandleStandardArgs.cmake:148 (message):  
Could NOT find Java (missing: Java_JAR_EXECUTABLE Java_JAVAC_EXECUTABLE  
Java_JAVAH_EXECUTABLE Java_JAVADOC_EXECUTABLE)
```

When this happen, provide a Java prefix as a hint on where the package was loaded. For instance on Ubuntu I was obliged to enter:

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

After that, the compilation of the CONNECT JDBC type was completed successfully.

## Compiling the Java source files

They are the source of the java wrapper classes used to access JDBC drivers. In the source distribution, they are located in the CONNECT source directory.

The default wrapper, JdbcInterface, is the only one distributed with binary distribution. It uses the standard way to get a connection to the drivers via the DriverManager.getConnection method. Other wrappers, only available with source distribution, enable connection to a Data Source, eventually implementing pooling. However, they must be compiled and installed manually.

The available wrappers are:

Wrapper	Description
JdbcInterface	Used to make the connection with available drivers the standard way.
ApacheInterface	Based on the Apache common-dbcp2 package this interface enables making connections to DBCP data sources with any JDBC drivers.
MariadbInterface	Makes connection to a MariaDB data source.
MysqlInterface	Makes connection to a Mysql data source. Must be used with a MySQL driver that implements data sources.
OracleInterface	Makes connection to an Oracle data source.
PostgresqlInterface	Makes connection to a Postgresql data source.

The wrapper used by default is specified by the `connect_java_wrapper` session variable and is initially set to

`wrappers/JdbcInterface`

. The wrapper to use for a table can also be specified in the option list as a wrapper option of the "create table" statements.

Note: Conforming java naming usage, class names are preceded by the java package name with a slash separator. However, this is not mandatory for CONNECT which adds the package name if it is missing.

The JdbcInterface wrapper is always usable when Java is present on your machine. Binary distributions have this wrapper already compiled as a JdbcInterface.jar file installed in the plugin directory whose path is automatically included in the class path of the JVM. Recent versions also add a JavaWrappers.jar that contains all these wrappers. Therefore there is no need to worry about its path.

Compiling the ApacheInterface wrapper requires that the Apache common-DBCP2 package be installed. Other wrappers are to be used only with the matching JDBC drivers that must be available when compiling them.

Installing the jar file in the plugin directory is the best place because it is part of the class path. Depending on what is installed on your system, the source files can be reduced accordingly. To compile only the JdbcInterface.java file the CMAKE\_JAVA\_INCLUDE\_PATH is not required. Here the paths are the ones existing on my Windows 7 machine and should be localized.

## Setting the Required Information

Before any operation with a JDBC driver can be made, CONNECT must initialize the environment that will make working with Java possible. This will consist of:

1. Loading dynamically the JVM library module.
2. Creating the Java Virtual Machine.
3. Establishing contact with the java wrapper class.
4. Connecting to the used JDBC driver.

Indeed, the JVM library module is not statically linked to the CONNECT plugin. This is to make it possible to use a CONNECT plugin that has been compiled with the JDBC table type on a machine where the Java SDK is not installed. Otherwise, users not interested in the JDBC table type would be obliged to install the Java SDK on their machine to be able to load the CONNECT storage engine.

## JVM Library Location

If the JVM library (jvm.dll on Windows, libjvm.so on Linux) was not placed in the standard library load path, CONNECT cannot find it and must be told where to search for it. This happens in particular on Linux when the Oracle Javapackage was installed in a private location.

If the JAVA\_HOME variable was exported as explained above, CONNECT can sometimes find it using this information. Otherwise, its search path can be added to the LD\_LIBRARY\_PATH environment variable. But all this is complicated because making environment variables permanent on Linux is painful (many different methods must be used depending on the Linux version and the used shell).

This is why CONNECT introduced a new global variable `connect_jvm_path` to store this information. It can be set when starting the server as a command line option or even afterwards before the first use of the JDBC table type. For example:

```
set global connect_jvm_path="/usr/lib/jvm/java-8-oracle/jre/lib/i386/client"
```

or

```
set global connect_jvm_path="/usr/lib/jvm/java-8-oracle/jre/lib/i386/server"
```

The client library is smaller and faster for connection. The server library is more optimized and can be used in case of heavy load usage.

Note that this may not be required on Windows because the path to the JVM library can sometimes be found in the registry.

Once this library is loaded, CONNECT can create the required Java Virtual Machine.

## Java Class Path

This is the list of paths Java searches when loading classes. With CONNECT, the classes to load will be the java wrapper classes used to communicate with the drivers , and the used JDBC driver classes that are grouped inside jar files. If the ApacheInterface wrapper must be used, the class path must also include all three jars used by the Apache package.

**Caution:** This class path is passed as a parameter to the Java Virtual Machine (JVM) when creating it and cannot be modified as it is a read only property. In addition, because MariaDB is a multi-threading application, this JVM cannot be destroyed and will be used throughout the entire life of the MariaDB server. Therefore, be sure it is correctly set before you use the JDBC table type for the first time. Otherwise, there will be practically no alternative than to shut down the server and restart it.

The path to the wrapper classes must point to the directory containing the wrappers sub-directory. If a JdbcInterface.jar file was made, its path is the directory where it is located followed by the jar file name. It is unclear where because this will depend on the installation process. If you start from a source distribution, it can be in the storage/connect directory where the CONNECT source files are or where you moved them or compiled the JdbcInterface.jar file.

For binary distributions, there is nothing to do because the jar file has been installed in the mysql share directory whose path is always automatically included in the class path available to the JVM.

Remaining are the paths of all the installed JDBC drivers that you intend to use. Remember that their path must include the jar file itself. Some applications use an environment variable CLASSPATH to contain them. Paths are separated by ':' on Linux and by ';' on Windows.

If the CLASSPATH variable actually exists and if it is available inside MariaDB, so far so good. You can check this using an UDF function provided by CONNECT that returns environment variable values:

```
create function envar returns string soname 'ha_connect.so';
select envar('CLASSPATH');
```

Most of the time, this will return null or some required files are missing. This is why CONNECT introduced a global variable to store this information. The paths specified in this variable will be added and have precedence to the ones, if any, of the CLASSPATH environment variable. As for the jvm path, this variable connect\_class\_path should be specified when starting the server but can also be set before using the JDBC table type for the first time.

The current directory (sql/data) is also placed by CONNECT at the beginning of the class path.

As an example, here is how I start MariaDB when doing tests on Linux:

```
olivier@olivier-Aspire-8920:~$ sudo /usr/local/mysql/bin/mysqld -u root --console --default-storage-engine=myisam --skip-innodb
```

## CONNECT JDBC Tables

These tables are given the type JDBC. For instance, supposing you want to access the boys table located on an external local or remote database management system providing a JDBC connector:

```
create table boys (
  name char(12),
  city char(12),
  birth date,
  hired date);
```

To access this table via JDBC you can create a table such as:

```
create table jboys engine=connect table_type=JDBC tablename=boys
connection='jdbc:mysql://localhost/dbname?user=root';
```

The CONNECTION option is the URL used to establish the connection with the remote server. Its syntax depends on the external DBMS and in this example is the one used to connect as root to a MySQL or MariaDB local database using the MySQL JDBC connector.

As for ODBC, the columns definition can be omitted and will be retrieved by the discovery process. The restrictions concerning column definitions are the same as for ODBC.

Note: The dbname indicated in the URL corresponds for many DBMS to the catalog information. For MySQL and MariaDB it is the schema (often called database) of the connection.

## Using a Federated Server

Alternatively, a JDBC table can specify its connection options via a Federated server. For instance, supposing you have a table accessing an external Postgresql table defined as:

```
create table juuid engine=connect table_type=JDBC tablename=testuuid
connection='jdbc:postgresql:test?user=postgres&password=pwd';
```

You can create a Federated server:

```
create server 'post1' foreign data wrapper 'postgresql' options (
HOST 'localhost',
DATABASE 'test',
USER 'postgres',
PASSWORD 'pwd',
PORT 0,
SOCKET '',
OWNER 'postgres');
```

Now the JDBC table can be created by:

```
create table juuid engine=connect table_type=JDBC connection='post1' tablename=testuuid;
```

or by:

```
create table juuid engine=connect table_type=JDBC connection='post1/testuuid';
```

In any case, the location of the remote table can be changed in the Federated server without having to alter all the tables using this server.

JDBC needs a URL to establish a connection. CONNECT was able to construct that URL from the information contained in such Federated server definition when the URL syntax is similar to the one of MySQL, MariaDB or Postgresql. However, other DBMSs such as Oracle use a different URL syntax. In this case, simply replace the HOST information by the required URL in the Federated server definition. For instance:

```
create server 'oracle' foreign data wrapper 'oracle' options (
HOST 'jdbc:oracle:thin:@localhost:1521:xe',
DATABASE 'SYSTEM',
USER 'system',
PASSWORD 'manager',
PORT 0,
SOCKET '',
OWNER 'SYSTEM');
```

Now you can create an Oracle table with something like this:

```
create table empor engine=connect table_type=JDBC connection='oracle/HR.EMPLOYEES';
```

Note: Oracle, as Postgresql, does not seem to understand the DATABASE setting as the table schema that must be specified in the Create Table statement.

## Connecting to a JDBC driver

When the connection to the driver is established by the JdbcInterface wrapper class, it uses the options that are provided when creating the CONNECT JDBC tables. Inside the default Java wrapper, the driver's main class is loaded by the DriverManager.getConnection function that takes three arguments:

<b>URL</b>	That is the URL that you specified in the CONNECTION option.
<b>User</b>	As specified in the OPTION_LIST or NULL if not specified.
<b>Password</b>	As specified in the OPTION_LIST or NULL if not specified.

The URL varies depending on the connected DBMS. Refer to the documentation of the specific JDBC driver for a description of the syntax to use. User and password can also be specified in the option list.

Beware that the database name in the URL can be interpreted differently depending on the DBMS. For MySQL this is the schema in which the tables are found. However, for Postgresql, this is the catalog and the schema must be specified using the CONNECT dbname option.

For instance a table accessing a Postgresql table via JDBC can be created with a create statement such as:

```
create table jt1 engine=connect table_type=JDBC
connection='jdbc:postgresql://localhost/mtr' dbname=public tablename=t1
option_list='User=mtr,Password=mtr';
```

Note: In previous versions of JDBC, to obtain a connection, java first had to initialize the JDBC driver by calling the method Class.forName. In this case, see the documentation of your DBMS driver to obtain the name of the class that implements the interface java.sql.Driver. This name can be specified as an option DRIVER to be put in the option list. However, most modern JDBC drivers since version 4 are self-loading and do not require this option to be specified.

The wrapper class also creates some required items and, in particular, a statement class. Some characteristics of this statement will depend on the options specified when creating the table:

<b>Scrollable</b>	To be specified in the option list. Determines the cursor type: no= forward_only or yes=scroll_insensitive.
<b>Block_size</b>	Will be used to set the statement fetch size.

## Fetch Size

The fetch size determines the number of rows that are internally retrieved by the driver on each interaction with the DBMS. Its default value depends on the JDBC driver. It is equal to 10 for some drivers but not for the MySQL or MariaDB connectors.

The MySQL/MariaDB connectors retrieve all the rows returned by one query and keep them in a memory cache. This is generally fine in most cases, but not when retrieving a large result set that can make the query fail with a memory exhausted exception.

To avoid this, when accessing a big table and expecting large result sets, you should specify the BLOCK\_SIZE option to 1 (the only acceptable value). However a problem remains:

Suppose you execute a query such as:

```
select id, name, phone from jbig limit 10;
```

Not knowing the limit clause, CONNECT sends to the remote DBMS the query:

```
SELECT id, name, phone FROM big;
```

In this query big can be a huge table having million rows. Having correctly specified the block size as 1 when creating the table, the wrapper just reads the 10 first rows and stops. However, when closing the statement, these MySQL/MariaDB drivers must still retrieve all the rows returned by the query. This is why, the wrapper class when closing the statement also cancels the query to stop that extra reading.

The bad news is that if it works all right for some previous versions of the MySQL driver, it does not work for new versions as well as for the MariaDB driver that apparently ignores the cancel command. The good news is that you can use an old MySQL driver to access MariaDB databases. It is also possible that this bug will be fixed in future versions of the drivers.

## Connection to a Data Source

This is the java preferred way to establish a connection because a data source can keep a pool of connections that can be re-used when necessary. This makes establishing connections much faster once it was done for the first time.

CONNECT provide additional wrappers whose files are located in the CONNECT source directory. The wrapper to use can be specified in the global variable connect\_java\_wrapper, which defaults to "JdbcInterface".

It can also be specified for a table in the option list by setting the option wrapper to its name. For instance:

```
create table jboys
engine=CONNECT table_type=JDBC tablename='boys'
connection='jdbc:mariadb://localhost/connect?user=root&useSSL=false'
option_list='Wrapper=MariadbInterface,Scrollable=1';
```

They can be used instead of the standard JdbcInterface and are using created data sources.

The Apache one uses data sources implemented by the Apache-commons-dbc package and can be used with all drivers including those not implementing data sources. However, the Apache package must be installed and its three required jar files accessible via the class path.

1. commons-dbc-2.1.1.jar
2. commons-pool2-2.4.2.jar
3. commons-logging-1.2.jar

Note: the versions numbers can be different on your installation.

The other ones use data sources provided by the matching JDBC driver. There are currently four wrappers to be used with mysql-6.0.2, mariadb, oracle and postgresql.

Unlike the class path, the used wrapper can be changed even after the JVM machine was created.

## Random Access to JDBC Tables

The same methods described for ODBC tables can be used with JDBC tables.

Note that in the case of the MySQL or MariaDB connectors, because they internally read the whole result set in memory, using the MEMORY option would be a waste of memory. It is much better to specify the use of a scrollable cursor when needed.

## Other Operations with JDBC Tables

Except for the way the connection string is specified and the table type set to JDBC, all operations with ODBC tables are done for JDBC tables the same

way. Refer to the ODBC chapter to know about:

- Accessing specified views (SRCDEF)
- Data modifying operations.
- Sending commands to a data source.
- JDBC catalog information.

Note: Some JDBC drivers fail when the global time\_zone variable is ambiguous, which sometimes happens when it is set to SYSTEM. If so, reset it to a not ambiguous value, for instance:

```
set global time_zone = '+2:00';
```

## JDBC Specific Restrictions

Connecting via data sources created externally (for instance using Tomcat) is not supported yet.

Other restrictions are the same as for the ODBC table type.

## Handling the UUID Data Type

PostgreSQL has a native UUID data type, internally stored as BIN(16). This is neither an SQL nor a MariaDB data type. The best we can do is to handle it by its character representation.

UUID will be translated to CHAR(36) when column definitions are set using discovery. Locally a PostgreSQL UUID column will be handled like a CHAR or VARCHAR column. Example:

Using the PostgreSQL table testuuid in the text database:

```
Table « public.testuuid »
Column | Type | Default
-----+-----+
id     | uuid | uuid_generate_v4()
msg    | text |
```

Its column definitions can be queried by:

```
create or replace table juuidcol engine=connect table_type=JDBC tablename=testuuid catfunc=columns
connection='jdbc:postgresql:test?user=postgres&password=pwd';
```

```
select table_name "Table", column_name "Column", data_type "Type", type_name "Name", column_size "Size" from juuidcol;
```

This query returns:

Table	Column	Type	Name	Size
testuuid	id	1111	uuid	2147483647
testuuid	msg	12	text	2147483647

Note: PostgreSQL, when a column size is undefined, returns 2147483647, which is not acceptable for MariaDB. CONNECT change it to the value of the connect\_conv\_size session variable. Also, for TEXT columns the data type returned is 12 (SQL\_VARCHAR) instead of -1 the SQL\_TEXT value.

Accessing this table via JDBC by:

```
CREATE TABLE juuid ENGINE=connect TABLE_TYPE=JDBC TABNAME=testuuid
CONNECTION='jdbc:postgresql:test?user=postgres&password=pwd';
```

it will be created by discovery as:

```
CREATE TABLE `juuid` (
  `id` char(36) DEFAULT NULL,
  `msg` varchar(8192) DEFAULT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 CONNECTION='jdbc:postgresql:test?user=postgres&password=pwd' `TABLE_TYPE`='JDBC' `TABNAME`='juuid'
```

Note: 8192 being here the \_connect\_conv\_size\_ value.

Let's populate it:

```
insert into juuid(msg) values('First');
insert into juuid(msg) values('Second');
select * from juuid;
```

Result:

id	msg
4b173ee1-1488-4355-a7ed-62ba59c2b3e7	First
6859f850-94a7-4903-8d3c-fc3c874fc274	Second

Here the id column values come from the DEFAULT of the PostgreSQL column that was specified as `uuid_generate_v4()`.

It can be set from MariaDB. For instance:

```
insert into juuid
  values('2f835fb8-73b0-42f3-a1d3-8a532b38feca','inserted');
insert into juuid values(NULL,'null');
insert into juuid values('','random');
select * from juuid;
```

Result:

id	msg
4b173ee1-1488-4355-a7ed-62ba59c2b3e7	First
6859f850-94a7-4903-8d3c-fc3c874fc274	Second
2f835fb8-73b0-42f3-a1d3-8a532b38feca	inserted
<null>	null
8fc0a30e-dc66-4b95-ba57-497a161f4180	random

The first insert specifies a valid UUID character representation. The second one set it to NULL. The third one (a void string) generates a Java random UUID. UPDATE commands obey the same specification.

These commands both work:

```
select * from juuid where id = '2f835fb8-73b0-42f3-a1d3-8a532b38feca';
delete from juuid where id = '2f835fb8-73b0-42f3-a1d3-8a532b38feca';
```

However, this one fails:

```
select * from juuid where id like '%42f3%';
```

Returning:

1296: Got error 174 'ExecuteQuery: org.postgresql.util.PSQLException: ERROR: operator does not exist: uuid ~ unknown hint: no operator corresponds to the data name and to the argument types.'

because CONNECT cond\_push feature added the WHERE clause to the query sent to PostgreSQL:

```
SELECT id, msg FROM testuuid WHERE id LIKE '%42f3%'
```

and the LIKE operator does not apply to UUID in PostgreSQL.

To handle this, a new session variable was added to CONNECT: connect\_cond\_push. It permits to specify if cond\_push is enabled or not for CONNECT and defaults to 1 (enabled). In this case, you can execute:

```
set connect_cond_push=0;
```

Doing so, the where clause will be executed by MariaDB only and the query will not fail anymore.

## Executing the JDBC tests

Four tests exist but they are disabled because requiring some work to localized them according to the operating system and available java package and JDBC drivers and DBMS.

Two of them, jdbc.test and jdbc\_new.test, are accessing MariaDB via JDBC drivers that are contained in a fat jar file that is part of the test. They should be executable without anything to do on Windows; simply adding the option --enable-disabled when running the tests.

However, on Linux these tests can fail to locate the JVM library. Before executing them, you should export the JAVA\_HOME environment variable set to the prefix of the java installation or export the LD\_LIBRARY\_PATH containing the path to the JVM lib.

## Fixing Problem With mysqldump

In some case or some platform, when CONNECT is set up for use with JDBC table types, this causes `mysqldump` with the option --all-databases to fail.

## 4.3.7.6.18 CONNECT MONGO Table Type: Accessing Collections from MongoDB

### Contents

#### 1. Accessing MongoDB from CONNECT

- 1. [Using the MongoDB C Driver](#)
- 2. [Using the Mongo Java Driver](#)
- 3. [Using JDBC](#)
- 4. [Using JSON](#)

#### 2. CONNECT MONGO Tables

- 1. [Fixing Problems With mysqldump](#)
- 2. [MongoDB Dot Notation](#)

#### 3. MONGO Specific Options

- 1. [Colist Option](#)
  - 2. [Filter Option](#)
  - 3. [Pipeline Option](#)
  - 4. [Fullarray Option](#)
4. [Create, Read, Update and Delete Operations](#)
5. [Status of MONGO Table Type](#)
6. [Current Restrictions](#)

Classified as a NoSQL database program, MongoDB uses JSON-like documents (BSON) grouped in collections. The MONGO type is used to directly access MongoDB collections as tables.

## Accessing MongoDB from CONNECT

Accessing MongoDB from CONNECT can be done in different ways:

1. As a MONGO table via the MongoDB C Driver.
2. As a MONGO table via the MongoDB Java Driver.
3. As a JDBC table using some commercially available MongoDB JDBC drivers.
4. As a JSON table via the MongoDB C or Java Driver.

### Using the MongoDB C Driver

This is currently not available from binary distributions but only for versions compiled from source. The preferred version of the MongoDB C Driver is 1.7, because they provide package recognition. What must be done is:

1. Install libbson and the MongoDB C Driver 1.7.
2. Configure, compile and install MariaDB.

With earlier versions of the Mongo C Driver, the additional include directories and libraries will have to be specified manually when compiling.

When possible, this is the preferred means of access because it does not require all the Java path settings etc. and is faster than using the Java driver.

### Using the Mongo Java Driver

This is possible with all distributions including JDBC support, or compiling from source. With a binary distribution that does not enable the MONGO table type, it is possible to access MongoDB using an OEM module. See [CONNECT OEM Table Example](#) for details. The only additional things to do are:

1. Install the MongoDB Java Driver by downloading its jar file. Several versions are available. If possible use the latest version 3 one.
2. Add the path to it in the CLASSPATH environment variable or in the connect\_class\_path variable. This is like what is done to declare JDBC drivers.

Connection is established by new Java wrappers Mongo3Interface and Mongo2Interface. They are available in a JDBC distribution in the Mongo2.jar and Mongo3.jar files (previously JavaWrappers.jar). If version 2 of the Java Driver is used, specify "Version=2" in the option list when creating tables.

### Using JDBC

See the documentation of the existing commercial JDBC Mongo drivers.

### Using JSON

See the specific chapter of the JSON Table Type.

The following describes the MONGO table type.

## CONNECT MONGO Tables

Creating and running MONGO tables requires a connection to a running local or remote MongoDB server.

A MONGO table is defined to access a MongoDB collection. The table rows will be the collection documents. For instance, to create a table based on the MongoDB sample collection restaurants, you can do something such as the following:

```
create table resto (
  _id varchar(24) not null,
  name varchar(64) not null,
  cuisine char(200) not null,
  borough char(16) not null,
  restaurant_id varchar(12) not null)
engine=connect table_type=MONGO tablename='restaurants'
data_charset=utf8 connection='mongodb://localhost:27017';
```

Note: The used driver is by default the C driver if only the MongoDB C Driver is installed and the Java driver if only the MongoDB Java Driver is installed. If both are available, it can be specified by the DRIVER option to be specified in the option list and defaults to C.

Here we did not define all the items of the collection documents but only those that are JSON values. The database is test by default. The connection value is the URI used to establish a connection to a local or remote MongoDB server. The value shown in this example corresponds to a local server started with its default port. It is the default connection value for MONGO tables so we could have omit specifying it.

Using discovery is available. This table could have been created by:

```
create table resto
engine=connect table_type=MONGO tablename='restaurants'
data_charset=utf8 option_list='level=-1';
```

Here "depth=-1" is used to create only columns that are simple values (no array or object). Without this, with the default value "depth=0" the table had been created as:

```
CREATE TABLE `resto` (
  `_id` char(24) NOT NULL,
  `address` varchar(136) NOT NULL,
  `borough` char(13) NOT NULL,
  `cuisine` char(64) NOT NULL,
  `grades` varchar(638) NOT NULL,
  `name` char(98) NOT NULL,
  `restaurant_id` char(8) NOT NULL
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='MONGO' `TABNAME`='restaurants' `DATA_CHARSET`='utf8';
```

## Fixing Problems With mysqldump

In some case or some platforms, when CONNECT is set up for use with JDBC table types, this causes `mysqldump` with the `--all-databases` option to fail.

This was reported by Robert Dyas who found the cause of it and how to fix it (see [MDEV-11238](#) ).

This occurs when the Java JRE "Usage Tracker" is enabled. In that case, Java creates a directory #mysql50#.oracle\_jre\_usage in the mysql data directory that shows up as a database but cannot be accessed via MySQL Workbench nor apparently backed up by `mysqldump --all-databases`.

Per the Oracle documentation (<https://docs.oracle.com/javacomponents/usage-tracker/overview/>) the "Usage Tracker" is disabled by default. It is enabled only when creating the properties file <JRE directory>/lib/management/usagetracker.properties. This turns out to be WRONG on some platforms as the file does exist by default on a new installation, and the existence of this file enables the usage tracker.

The solution on CentOS 7 with the Oracle JVM is to rename or delete the usagetracker.properties file (to disable it) and then delete the bogus folder it created in the mysql database directory, then restart.

For example, the following works:

```
sudo mv /usr/java/default/jre/lib/management/management.properties /usr/java/default/jre/lib/management/management.properties.TR
sudo reboot
sudo rm -rf /var/lib/mysql/.oracle_jre_usage
sudo reboot
```

In this collection, the address column is a JSON object and the column grades is a JSON array. Unlike the JSON table, just specifying the column name with no Jpath result in displaying the JSON representation of them. For instance:

```
select name, address from resto limit 3;
```

name	address
Morris Park Bake Shop	{"building":"1007","coord":[-73.8561,40.8484], "street":"Morris ParkAve", "zipcode":"10462"}
Wendy'S	{"building":"469","coord":[-73.9617,40.6629], "street":"Flatbush Avenue", "zipcode":"11225"}
Reynolds Restaurant	{"building":"351","coord":[-73.9851,40.7677], "street":"West 57Street", "zipcode":"10019"}

## MongoDB Dot Notation

To address the items inside object or arrays, specify the Jpath in MongoDB syntax (if using Discovery, specify the Depth option accordingly):

From Connect 1.7.0002

```
create table newresto (
  _id varchar(24) not null,
  name varchar(64) not null,
  cuisine char(200) not null,
  borough char(16) not null,
  street varchar(65) jpath='address.street',
  building char(16) jpath='address.building',
  zipcode char(5) jpath='address.zipcode',
  grade char(1) jpath='grades.0.grade',
  score int(4) not null jpath='grades.0.score',
  `date` date jpath='grades.0.date',
  restaurant_id varchar(255) not null
engine=connect table_type=MONGO tablename='restaurants'
data_charset=utf8 connection='mongodb://localhost:27017';
```

Before Connect 1.7.0002

```
create table newresto (
  _id varchar(24) not null,
  name varchar(64) not null,
  cuisine char(200) not null,
  borough char(16) not null,
  street varchar(65) field_format='address.street',
  building char(16) field_format='address.building',
  zipcode char(5) field_format='address.zipcode',
  grade char(1) field_format='grades.0.grade',
  score int(4) not null field_format='grades.0.score',
  `date` date field_format='grades.0.date',
  restaurant_id varchar(255) not null
engine=connect table_type=MONGO tablename='restaurants'
data_charset=utf8 connection='mongodb://localhost:27017';
```

If this is not done, the Oracle JVM will start the usage tracker, which will create the hidden folder .oracle\_jre\_usage in the mysql home directory, which will cause a mysql dump of the server to fail.

```
select name, street, score, date from newresto limit 5;
```

name	street	score	date
Morris Park Bake Shop	Morris Park Ave	2	03/03/2014
Wendy'S	Flatbush Avenue	8	30/12/2014
Dj Reynolds Pub And Restaurant	West 57 Street	2	06/09/2014
Riviera Caterer	Stillwell Avenue	5	10/06/2014
Tov Kosher Kitchen	63 Road	20	24/11/2014

## MONGO Specific Options

The MongoDB syntax for Jpath does not allow the CONNECT specific items on arrays. The same effect can still be obtained by a different way. For this, additional options are used when creating MONGO tables.

Option	Type	Description
Colist	String	Options to pass to the MongoDB cursor.
Filter	String	Query used by the MongoDB cursor.
Pipeline*	Boolean	If True, Colist is a pipeline.
Fullarray*	Boolean	Used when creating with Discovery.
Driver*	String	C or Java.
Version*	Integer	The Java Driver version (defaults to 3)

\* : To be specified in the option list.

Note: For the content of these options, refer to the MongoDB documentation.

## Colist Option

Used to pass different options when making the MongoDB cursor used to retrieve the collation documents. One of them is the projection, allowing to limit the items retrieved in documents. It is hardly useful because this limitation is made automatically by CONNECT. However, it can be used when using discovery to eliminate the \_id (or another) column when you are not willing to keep it:

```
create table restest
engine=connect table_type=MONGO tablename='restaurants'
data_charset=utf8 option_list='depth=-1'
colist='{"projection":{"_id":0},"limit":5}';
```

In this example, we added another cursor option, the limit option that works like the limit SQL clause.

This additional option works only with the C driver. When using the Java driver, colist should be:

```
colist='{"_id":0}';
```

And limit would be specified with select statements.

Note: When used with a JSON table, to specify the projection list (or 'all' to get all columns) makes JPATH to be Connect Json paths, not MongoDB ones, allowing JPATH options not available to MongoDB.

## Filter Option

This option is used to specify a "filter" that works as a where clause on the table. Supposing we want to create a table restricted to the restaurant making English cuisine that are not located in the Manhattan borough, we can do it by:

```
create table english
engine=connect table_type=MONGO tablename='restaurants'
data_charset=utf8
colist='{"projection":{"cuisine":0}}'
filter='{"cuisine":"English","borough":{"$ne":"Manhattan"}}'
option_list='Depth=-1';
```

And if we ask:

```
select * from english;
```

This query will return:

_id	borough	name	restaurant_id
58ada47de5a51ddfc55ee1f3	Brooklyn	The Park Slope Chipshop	40816202
58ada47de5a51ddfc55ee999	Brooklyn	Chip Shop	41076583
58ada47ee5a51ddfc5f13d5	Brooklyn	The Monro	41660253
58ada47ee5a51ddfc5f176e	Brooklyn	Dear Bushwick	41690534
58ada47ee5a51ddfc5f1e91	Queens	Snowdonia Pub	50000290

## Pipeline Option

When this option is specified as true (by YES or 1) the Colist option contains a MongoDB pipeline applying to the table collation. This is a powerful mean for doing things such as expanding arrays like we do with JSON tables. For instance:

```
create table resto2 (
name varchar(64) not null,
borough char(16) not null,
date datetime not null,
grade char(1) not null,
score int(4) not null
engine=connect table_type=MONGO tablename='restaurants' data_charset=utf8
colist='{"pipeline":[{"$match":{"cuisine":"French"}},{"$unwind":"$grades"}, {"$project":{"_id":0,"name":1,"borough":1,"date":1,"grades":1}}]
option_list='Pipeline=1';
```

In this pipeline "\$match" is an early filter, "\$unwind" means that the grades array will be expanded (one Document for each array values) and "\$project" eliminates the \_id and cuisine columns and gives the Jpath for the date, grade and score columns.

```
select name, grade, score, date from resto2
where borough = 'Bronx';
```

This query replies:

name	grade	score	date
Bistro Sk	A	10	21/11/2014 01:00:00
Bistro Sk	A	12	19/02/2014 01:00:00
Bistro Sk	B	18	12/06/2013 02:00:00

This make possible to get things like we do with JSON tables:

```
select name, avg(score) average from resto2
group by name having average >= 25;
```

Can be used to get the average score inside the grades array.

name	average
Bouley Botanical	25,0000
Cheri	46,0000
Graine De Paris	30,0000
Le Pescadeux	29,7500

## Fullarray Option

This option, like the Depth option, is only interpreted when creating a table with Discovery (meaning not specifying the columns). It tells CONNECT to generate a column for all existing values in the array. For instance, let us see the MongoDB collection tar by:

From Connect 1.7.0002

```
create table seetar (
Collection varchar(300) not null jpath='*'
engine=CONNECT table_type=MONGO tablename=tar;
```

Before Connect 1.7.0002

```
create table seetar (
Collection varchar(300) not null field_format='*'
engine=CONNECT table_type=MONGO tablename=tar;
```

The format '\*' indicates we want to see the Json documents. This small collection is:

Collection
{"_id": {"\$oid": "58f63a5099b37d9c930f9f3b"}, "item": "journal", "prices": [87.0, 45.0, 63.0, 12.0, 78.0]}
{"_id": {"\$oid": "58f63a5099b37d9c930f9f3c"}, "item": "notebook", "prices": [123.0, 456.0, 789.0]}

The Fullarray option can be used here to generate enough columns to see all the prices of the document prices array.

```
create table tar
engine=connect table_type=MONGO
colist='{"projection":{"_id":0}}'
option_list='depth=1,Fullarray=YES';
```

The table has been created as:

From Connect 1.7.0002

```
CREATE TABLE `tar` (
`item` char(8) NOT NULL,
`prices_0` double(12,6) NOT NULL `JPATH`='prices.0',
`prices_1` double(12,6) NOT NULL `JPATH`='prices.1',
`prices_2` double(12,6) NOT NULL `JPATH`='prices.2',
`prices_3` double(12,6) DEFAULT NULL `JPATH`='prices.3',
`prices_4` double(12,6) DEFAULT NULL `JPATH`='prices.4'
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='MONGO' `COLIST`='{"projection":{"_id":0}}' `OPTION_LIST`='depth=1,Fullarra
```

Before Connect 1.7.0002

```
CREATE TABLE `tar` (
  `item` char(8) NOT NULL,
  `prices_0` double(12,6) NOT NULL `FIELD_FORMAT`='prices.0',
  `prices_1` double(12,6) NOT NULL `FIELD_FORMAT`='prices.1',
  `prices_2` double(12,6) NOT NULL `FIELD_FORMAT`='prices.2',
  `prices_3` double(12,6) DEFAULT NULL `FIELD_FORMAT`='prices.3',
  `prices_4` double(12,6) DEFAULT NULL `FIELD_FORMAT`='prices.4'
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='MONGO' `COLIST`='{"projection":{"_id":0}}' `OPTION_LIST`='level=1,Fullarray'
```

And is displayed as:

item	prices_0	prices_1	prices_2	prices_3	prices_4
journal	87.00	45.00	63.00	12.00	78.00
notebook	123.00	456.00	789.00	NULL	NULL

## Create, Read, Update and Delete Operations

All modifying operations are supported. However, inserting into arrays must be done in a specific way. Like with the Fullarray option, we must have enough columns to specify the array values. For instance, we can create a new table by:

From Connect 1.7.0002

```
create table testin (
n int not null,
m char(12) not null,
surname char(16) not null jpath='person.name.first',
name char(16) not null jpath='person.name.last',
age int(3) not null jpath='person.age',
price_1 double(8,2) jpath='d.0',
price_2 double(8,2) jpath='d.1',
price_3 double(8,2) jpath='d.2')
engine=connect table_type=MONGO tabname='tin'
connection='mongodb://localhost:27017';
```

Before Connect 1.7.0002

```
create table testin (
n int not null,
m char(12) not null,
surname char(16) not null field_format='person.name.first',
name char(16) not null field_format='person.name.last',
age int(3) not null field_format='person.age',
price_1 double(8,2) field_format='d.0',
price_2 double(8,2) field_format='d.1',
price_3 double(8,2) field_format='d.2')
engine=connect table_type=MONGO tabname='tin'
connection='mongodb://localhost:27017';
```

Now it is possible to populate it by:

```
insert into testin values
(1789, 'Welcome', 'Olivier', 'Bertrand', 56, 3.14, 2.36, 8.45),
(1515, 'Hello', 'John', 'Smith', 32, 65.17, 98.12, NULL),
(2014, 'Coucou', 'Foo', 'Bar', 20, -1.0, 74, 81356);
```

The result will be:

n	m	surname	name	age	price_1	price_2	price_3
1789	Welcome	Olivier	Bertrand	56	3,14	2,36	8,45
1515	Hello	John	Smith	32	65,17	98,12	NULL
2014	Coucou	Foo	Bar	20	-1	74	81356

Note: If the collection does not exist yet when creating the table and inserting in it, MongoDB creates it automatically.

It can be updated by queries such as:

```
update tintin set price_3 = 83.36 where n = 2014;
```

To look how the array is generated, let us create another table:

From Connect 1.7.0002

```
create table tintin (
n int not null,
name char(16) not null jpath='person.name.first',
prices varchar(255) jpath='d')
engine=connect table_type=MONGO tablename='tin';
```

Before Connect 1.7.002

```
create table tintin (
n int not null,
name char(16) not null field_format='person.name.first',
prices varchar(255) field_format='d')
engine=connect table_type=MONGO tablename='in';
```

This table is displayed as:

From Connect 1.7.0002

n	name	prices
1789	Olivier	[3.1400000000000001243,2.35999999999998757,8.44999999999992895]
1515	John	[65.17000000000001705,98.12000000000004547,null]
2014	Foo	[null,74.0,83.3599999999999432]

Before Connect 1.7.002

n	name	prices
1789	Olivier	[3.14, 2.36, 8.45]
1515	John	[65.17, 98.12]
2014	Foo	[<null>, 74.0, 83.36]

Note: This last table can be used to make array calculations like with JSON tables using the JSON UDF functions. For instance:

```
select name, jsonget_real(prices,'[+]') sum_prices, jsonget_real(prices,'[!]') avg_prices from tintin;
```

This query returns:

name	sum_prices	avg_prices
Olivier	13.95	4.65
John	163.29	81.64
Foo	157,36	78.68

Note: When calculating on arrays, null values are ignored.

## Status of MONGO Table Type

This table type is still under development. It has significant advantages over the JSON type to access MongoDB collections. Firstly, the access being direct, tables are always up to date whether the collection has been modified by another application. Performance wise, it can be faster than JSON, because most processing is done by MongoDB on BSON, its internal representation of JSON data, which is designed to optimize all operations. Note that using the MongoDB C Driver can be faster than using the MongoDB Java Driver.

## Current Restrictions

- Option “CATFUNC=tables” is not implemented yet.
- Options SRCDEF and EXECSRC do not apply to MONGO tables.

## 4.3.7.6.19 CONNECT MYSQL Table Type: Accessing MySQL/MariaDB Tables

## Contents

1. Charset Specification
2. Indexing of MySQL tables
3. Data Modifying Operations
4. Sending commands to a MariaDB Server
  1. Sending several commands in one call
  2. Retrieving Warnings and Notes
5. Connection Engine Limitations
  1. Data types
  2. SQL Limitations
6. CONNECT MySQL versus FEDERATED
7. See also

This table type uses libmysql API to access a MySQL or MariaDB table or view. This table must be created on the current server or on another local or remote server. This is similar to what the [FederatedX](#) storage engine provides with some differences.

Currently the Federated-like syntax can be used to create such a table, for instance:

```
create table essai (
    num integer(4) not null,
    line char(15) not null)
engine=CONNECT table_type=MYSQL
connection='mysql://root@localhost/test/people';
```

The connection string can have the same syntax as that used by FEDERATED

```
scheme://username:password@hostname:port/database tablename
scheme://username@hostname/database tablename
scheme://username:password@hostname/database tablename
scheme://username:password@hostname/database tablename
```

However, it can also be mixed with connect standard options. For instance:

```
create table essai (
    num integer(4) not null,
    line char(15) not null)
engine=CONNECT table_type=MYSQL dbname=test tabname=people
connection='mysql://root@localhost';
```

It can also be specified as a reference to a federated server:

```
connection="connection_one"
connection="connection_one/table_foo"
```

The pure (deprecated) CONNECT syntax is also accepted:

```
create table essai (
    num integer(4) not null,
    line char(15) not null)
engine=CONNECT table_type=MYSQL dbname=test tabname=people
option_list='user=root,host=localhost';
```

The specific connection items are:

Option	Default value	Description
Table	The table name	The name of the table to access.
Database	The current DB name	The database where the table is located.
Host	localhost*	The host of the server, a name or an IP address.
User	The current user	The connection user name.
Password	No password	An optional user password.
Port	The currently used port	The port of the server.
Quoted	0	1 if remote Tabname must be quoted.

- - When the host is specified as "localhost", the connection is established on Linux using Linux sockets. On Windows, the connection is established by default using shared memory if it is enabled. If not, the TCP protocol is used. An alternative is to specify the host as "." to use a named pipe connection (if it is enabled). This makes possible to use these table types with server skipping networking.

**Caution:** Take care not to refer to the MYSQL table itself to avoid an infinite loop!

MYSQL table can refer to the current server as well as to another server. Views can be referred by name or directly giving a source definition, for instance:

```
create table grp engine=connect table_type=mysql  
CONNECTION='mysql://root@localhost/test/people'  
SRCDEF='select title, count(*) as cnt from employees group by title';
```

When specified, the columns of the mysql table must exist in the accessed table with the same name, but can be only a subset of them and specified in a different order. Their type must be a type supported by CONNECT and, if it is not identical to the type of the accessed table matching column, a conversion can be done according to the rules given in [Data type conversion](#).

Note: For columns prone to be targeted by a where clause, keep the column type compatible with the source table column type (numeric or character) to have a correct rephrasing of the where clause.

If you do not want to restrict or change the column definition, do not provide it and leave CONNECT get the column definition from the remote server. For instance:

```
create table essai engine=CONNECT table_type=MYSQL  
connection='mysql://root@localhost/test/people';
```

This will create the `essai` table with the same columns than the `people` table. If the target table contains CONNECT incompatible type columns, see [Data type conversion](#) to know how these columns can be converted or skipped.

## Charset Specification

When accessing the remote table, CONNECT sets the connection charset set to the default local table charset as the FEDERATED engine does.

Do not specify a column character set if it is different from the table default character set even when it is the case on the remote table. This is because the remote column is translated to the local table character set when reading it. This is the default but it can be modified by the setting the `character_set_results` variable of the target server. If it must keep its setting, for instance to UTF8 when containing Unicode characters, specify the local default charset to its character set.

This means that it is not possible to correctly retrieve a remote table if it contains columns having different character sets. A solution is to retrieve it by several local tables, each accessing only columns with the same character set.

## Indexing of MYSQL tables

Indexes are rarely useful with MYSQL tables. This is because CONNECT tries to access only the requested rows. For instance if you ask:

```
select * from essai where num = 23;
```

CONNECT will construct and send to the server the query:

```
SELECT num, line FROM people WHERE num = 23
```

If the `people` table is indexed on `num`, indexing will be used on the remote server. This, in all cases, will limit the amount of data to retrieve on the network.

However, an index can be specified for columns that are prone to be used to join another table to the MYSQL table. For instance:

```
select d.id, d.name, f.dept, f.salary  
from loc_tab d straight_join cnc_tab f on d.id = f.id  
where f.salary > 10000;
```

If the `id` column of the remote table addressed by the `cnc_tab` MYSQL table is indexed (which is likely if it is a key) you should also index the `id` column of the MYSQL `cnc_tab` table. If so, using “remote” indexing as does FEDERATED, only the useful rows of the remote table will be retrieved during the join process. However, because these rows are retrieved by separate `SELECT` statements, this will be useful only when retrieving a few rows of a big table.

In particular, you should not specify an index for columns not used for joining and above all DO NOT index a joined column if it is not indexed in the remote table. This would cause multiple scans of the remote table to retrieve the joined rows one by one.

## Data Modifying Operations

The CONNECT MYSQL type supports `SELECT` and `INSERT` and a somewhat limited form of `UPDATE` and `DELETE`. These are described below.

The MYSQL type uses similar methods than the ODBC type to implement the `INSERT`, `UPDATE` and `DELETE` commands. Refer to the ODBC chapter for the restrictions concerning them.

For the **UPDATE** and **DELETE** commands, there are fewer restrictions because the remote server being a MySQL server, the syntax of the command will be always acceptable by the remote server.

For instance, you can freely use keywords like IGNORE or LOW\_PRIORITY as well as scalar functions in the SET and WHERE clauses.

However, there is still an issue on multi-table statements. Let us suppose you have a *t1* table on the remote server and want to execute a query such as:

```
update essai as x set line = (select msg from t1 where id = x.num)
where num = 2;
```

When parsed locally, you will have errors if no *t1* table exists or if it does not have the referenced columns. When *t1* does not exist, you can overcome this issue by creating a local dummy *t1* table:

```
create table t1 (id int, msg char(1)) engine=BLACKHOLE;
```

This will make the local parser happy and permit to execute the command on the remote server. Note however that having a local MySQL table defined on the remote *t1* table does not solve the problem unless it is also names *t1* locally.

This is why, to permit to have all types of commands executed by the data source without any restriction, CONNECT provides a specific MySQL table subtype described now.

## Sending commands to a MariaDB Server

This can be done like for ODBC or JDBC tables by defining a specific table that will be used to send commands and get the result of their execution..

```
create table send (
  command varchar(128) not null,
  warnings int(4) not null flag=3,
  number int(5) not null flag=1,
  message varchar(255) flag=2)
engine=connect table_type=mysql
connection='mysql://user@host/database'
option_list='Execsrc=1,Maxerr=2';
```

The key points in this create statement are the EXECSRC option and the column definition.

The EXECSRC option tells that this table will be used to send commands to the MariaDB server. Most of the sent commands do not return result set. Therefore, the table columns are used to specify the command to be executed and to get the result of the execution. The name of these columns can be chosen arbitrarily, their function coming from the FLAG value:

**Flag=0:** The command to execute (the default)

**Flag=1:** The number of affected rows, or the result number of columns if the command would return a result set.

**Flag=2:** The returned (eventually error) message.

**Flag=3:** The number of warnings.

How to use this table and specify the command to send? By executing a command such as:

```
select * from send where command = 'a command';
```

This will send the command specified in the WHERE clause to the data source and return the result of its execution. The syntax of the WHERE clause must be exactly as shown above. For instance:

```
select * from send where command =
'CREATE TABLE people (
  num integer(4) primary key autoincrement,
  line char(15) not null';
```

This command returns:

command	warnings	number	message
CREATE TABLE people (num integer(4) primary key aut...	0	0	Affected rows

## Sending several commands in one call

It can be faster to execute because there will be only one connection for all of them. To send several commands in one call, use the following syntax:

```
select * from send where command in (
"update people set line = 'Two' where id = 2",
"update people set line = 'Three' where id = 3");
```

When several commands are sent, the execution stops at the end of them or after a command that is in error. To continue after n errors, set the option maxerr= n (0 by default) in the option list.

**Note 1:** It is possible to specify the SRCDEF option when creating an EXECSRC table. It will be the command sent by default when a WHERE clause is not specified.

**Note 2:** Backslashes inside commands must be escaped. Simple quotes must be escaped if the command is specified between simple quotes, and double quotes if it is specified between double quotes.

**Note 3:** Sent commands apply in the specified database. However, they can address any table within this database.

**Note 4:** Currently, all commands are executed in mode AUTOCOMMIT.

## Retrieving Warnings and Notes

If a sent command causes warnings to be issued, it is useless to resend a “show warnings” command because the MariaDB server is opened and closed when sending commands. Therefore, getting warnings requires a specific (and tricky) way.

To indicate that warning text must be added to the returned result, you must send a multi-command query containing “pseudo” commands that are not sent to the server but directly interpreted by the EXECSRC table. These “pseudo” commands are:

**Warning** To get warnings

**Note** To get notes

**Error** To get errors returned as warnings (?)

Note that they must be spelled (case insensitive) exactly as above, no final “s”. For instance:

```
select * from send where command in ('Warning','Note',
'drop table if exists try',
'create table try (id int key auto_increment, msg varchar(32) not
null) engine=aria',
"insert into try(msg) values('One'),(NULL),('Three') ",
"insert into try values(2,'Deux') on duplicate key update msg =
'Two'",
"insert into try(message) values('Four'),('Five'),('Six')",
'insert into try(id) values(NULL)',
"update try set msg = 'Four' where id = 4",
'select * from try');
```

This can return something like this:

command	warnings	number	message
drop table if exists try	1	0	Affected rows
Note	0	1051	Unknown table 'try'
create table try (id int key auto_increment, msg...	0	0	Affected rows
insert into try(msg) values('One'),(NULL),('Three')	1	3	Affected rows
Warning	0	1048	Column 'msg' cannot be null
insert into try values(2,'Deux') on duplicate key...	0	2	Affected rows
insert into try(msge) values('Four'),('Five'),('Six')	0	1054	Unknown column 'msge' in 'field list'
insert into try(id) values(NULL)	1	1	Affected rows
Warning	0	1364	Field 'msg' doesn't have a default value

update try set msg = 'Four' where id = 4	0	1	Affected rows
select * from try	0	2	Result set columns

The execution continued after the command in error because of the MAXERR option. Normally this would have stopped the execution.

Of course, the last “select” command is useless here because it cannot return the table contain. Another MYSQL table without the EXECSRC option and with proper column definition should be used instead.

## Connection Engine Limitations

### Data types

There is a maximum key.index length of 255 bytes. You may be able to declare the table without an index and rely on the engine condition pushdown and remote schema.

The following types can't be used:

- [BIT](#)
- [BINARY](#)
- [TINYBLOB , BLOB , MEDIUMBLOB , LONGBLOB](#)
- [TINYTEXT , MEDIUMTEXT , LONGTEXT](#)
- [ENUM](#)
- [SET](#)
- [Geometry types](#)

Note: [TEXT](#) is allowed. However, the handling depends on the values given to the [connect\\_type\\_conv](#) and [connect\\_conv\\_size](#) system variables, and by default no conversion of TEXT columns is permitted.

### SQL Limitations

The following SQL queries are not supported

- [REPLACE INTO](#)
- [INSERT ... ON DUPLICATE KEY UPDATE](#)

## CONNECT MYSQL versus FEDERATED

The CONNECT MYSQL table type should not be regarded as a replacement for the [FEDERATED\(X\)](#) engine. The main use of the MYSQL type is to access other engine tables as if they were CONNECT tables. This was necessary when accessing tables from some CONNECT table types such as [TBL](#) , [XCOL](#) , [OCCUR](#) , or [PIVOT](#) that are designed to access CONNECT tables only. When their target table is not a CONNECT table, these types are silently using internally an intermediate MYSQL table.

However, there are cases where you can use MYSQL CONNECT tables yourself, for instance:

1. When the table will be used by a [TBL](#) table. This enables you to specify the connection parameters for each sub-table and is more efficient than using a local FEDERATED sub-table.
2. When the desired returned data is directly specified by the SRCDEF option. This is great to let the remote server do most of the job, such as grouping and/or joining tables. This cannot be done with the FEDERATED engine.
3. To take advantage of the *push\_cond* facility that adds a where clause to the command sent to the remote table. This restricts the size of the result set and can be crucial for big tables.
4. For tables with the EXECSRC option on.
5. When doing tests. For instance to check a connection string.

If you need multi-table updating, deleting, or bulk inserting on a remote table, you can alternatively use the FEDERATED engine or a “send” table specifying the EXECSRC option on.

### See also

- [Using the TBL and MYSQL types together](#)

## 4.3.7.6.20 CONNECT PROXY Table Type

### Contents

1. [Proxy on non-CONNECT Tables](#)
2. [Using a PROXY Table as a View](#)
3. [Avoiding PROXY table loop](#)
4. [Modifying Operations](#)

A

PROXY

table is a table that accesses and reads the data of another table or view. For instance, to create a table based on the boys

FIX

table:

```
create table xboy engine=connect  
table_type=PROXY tabname=boys;
```

Simply,

PROXY

being the default type when

TABNAME

is specified:

```
create table xboy engine=connect tabname=boys;
```

Because the boys table can be directly used, what can be the use of a proxy table? Well, its main use is to be internally used by other table types such as **TBL** , **XCOL** , **OCCUR** , or **PIVOT** . Sure enough, PROXY tables are CONNECT tables, meaning that they can be based on tables of any engines and accessed by table types that need to access CONNECT tables.

## Proxy on non-CONNECT Tables

When the sub-table is a view or not a CONNECT table, CONNECT internally creates a temporary CONNECT table of **MYSQL** type to access it. This connection uses the same default parameters as for a

MYSQL

table. It is also possible to specify them to the

PROXY

table using in the

PROXY

declaration the same

OPTION\_LIST

options as for a

MYSQL

table. Of course, it is simpler and more natural to use directly the MYSQL type in this case.

Normally, the default parameters should enable the

PROXY

table to reconnect the server. However, an issue is when the current user was logged using a password. The security protocol prevents CONNECT to retrieve this password and requires it to be given in the

PROXY

table create statement. For instance adding to it:

```
... option_list='Password=mypass' ;
```

However, it is often not advisable to write in clear a password that can be seen by all user able to see the table declaration by show create table, in particular, if the table is used when the current user is root. To avoid this, a specific user should be created on the local host that will be used by proxy tables to retrieve local tables. This user can have minimum grant options, for instance SELECT on desired directories, and needs no password.

Supposing 'proxy' is such a user, the option list to add will be:

```
... option_list='user=proxy' ;
```

## Using a PROXY Table as a View

A

PROXY

table can also be used by itself to modify the way a table is viewed. For instance, a proxy table does not use the indexes of the object table. It is also possible to define its columns with different names or type, to use only some of them or to changes their order. For instance:

```
create table city (  
    city varchar(11),  
    boy char(12) flag=1,  
    birth date)  
engine=CONNECT tabname=boys;  
select * from city;
```

This will display:

city	boy	birth
Boston	John	1986-01-25
Boston	Henry	1987-06-07
San Jose	George	1981-08-10
Chicago	Sam	1979-11-22
Dallas	James	1992-05-13
Boston	Bill	1986-09-11

Here we did not have to specify column format or offset because data are retrieved from the boys table, not directly from the boys.txt file. The flag option of the *boy* column indicates that it correspond to the first column of the boys table, the *name* column.

## Avoiding PROXY table loop

CONNECT is able to test whether a

PROXY

, or

PROXY

-based, table refers directly or indirectly to itself. If a direct reference can be tested at table creation, an indirect reference can only be tested when executing a query on the table. However, this is possible only for local tables. When using remote tables or views, a problem can occur if the remote table or the view refers back to one of the local tables of the chain. The same caution should be used than when using

FEDERATEDX

tables.

**Note:** All

PROXY

or

PROXY

-based tables are read-only in this version.

## Modifying Operations

All **INSERT / UPDATE / DELETE** operations can be used with proxy tables. However, the same restrictions applying to the source table also apply to the proxy table.

Note: All PROXY and PROXY-based table types are not indexable.

## 4.3.7.6.21 CONNECT XCOL Table Type

### Contents

1. [Using Special Columns with XCOL](#)
2. [XCOL tables based on specified views](#)

XCOL

tables are based on another table or view, like

PROXY

tables. This type can be used when the object table has a column that contains a list of values.

Suppose we have a '*children*' table that can be displayed as:

name	childlist
Sophie	Vivian, Antony
Lisbeth	Lucy, Charles, Diana
Corinne	
Claude	Marc
Janet	Arthur, Sandra, Peter, John

We can have a different view on these data, where each child will be associated with his/her mother by creating an

XCOL

table by:

```
CREATE TABLE xchild (
    mother char(12) NOT NULL,
    child char(12) DEFAULT NULL flag=2
) ENGINE=CONNECT table_type=XCOL tablename='chlist'
option_list='colname=child';
```

The

COLNAME

option specifies the name of the column receiving the list items. This will return from:

```
select * from xchild;
```

The requested view:

mother	child
Sophia	Vivian
Sophia	Antony
Lisbeth	Lucy
Lisbeth	Charles
Lisbeth	Diana
Corinne	NULL
Claude	Marc
Janet	Arthur
Janet	Sandra
Janet	Peter
Janet	John

Several things should be noted here:

- When the original *children* field is void, what happens depends on the NULL specification of the "multiple" column. If it is nullable, like here, a void string will generate a NULL value. However, if the column is not nullable, no row will be generated at all.
- Blanks after the separator are ignored.
- No copy of the original data was done. Both tables use the same source data.
- Specifying the column definitions in the

```
CREATE TABLE  
statement is optional.
```

The "multiple" column *child* can be used as any other column. For instance:

```
select * from xchild where substr(child,1,1) = 'A';
```

This will return:

Mother	Child
Sophia	Antony
Janet	Arthur

If a query does not involve the "multiple" column, no row multiplication will be done. For instance:

```
select mother from xchild;
```

This will just return all the mothers:

mother
Sophia
Lisbeth
Corinne
Claude

Janet

The same occurs with other types of select statements, for instance:

```
select count(*) from xchild;      -- returns 5
select count(child) from xchild;  -- returns 10
select count(mother) from xchild; -- returns 5
```

Grouping also gives different result:

```
select mother, count(*) from xchild group by mother;
```

Replies:

mother	count(*)
Claude	1
Corinne	1
Janet	1
Lisbeth	1
Sophia	1

While the query:

```
select mother, count(child) from xchild group by mother;
```

Gives the more interesting result:

mother	count(child)
Claude	1
Corinne	0
Janet	4
Lisbeth	3
Sophia	2

Some more options are available for this table type:

Option	Description
Sep_char	The separator character used in the "multiple" column, defaults to the comma.
Mult	Indicates the max number of multiple items. It is used to internally calculate the max size of the table and defaults to 10. (To be specified in OPTION_LIST ).

## Using Special Columns with XCOL

Special columns can be used in XCOL tables. The mostly useful one is ROWNUM that gives the rank of the value in the list of values. For instance:

```
CREATE TABLE xchild2 (
rank int NOT NULL SPECIAL=ROWID,
mother char(12) NOT NULL,
child char(12) NOT NULL flag=2
) ENGINE=CONNECT table_type=XCOL tablename='chlist' option_list='colname=child';
```

This table will be displayed as:

rank	mother	child
1	Sophia	Vivian
2	Sophia	Antony
1	Lisbeth	Lucy
2	Lisbeth	Charles
3	Lisbeth	Diana

1	Claude	Marc
1	Janet	Arthur
2	Janet	Sandra
3	Janet	Peter
4	Janet	John

To list only the first child of each mother you can do:

```
SELECT mother, child FROM xchild2 WHERE rank = 1 ;
```

returning:

mother	child
Sophia	Vivian
Lisbeth	Lucy
Claude	Marc
Janet	Arthur

However, note the following pitfall: trying to get the names of all mothers having more than 2 children cannot be done by:

```
SELECT mother FROM xchild2 WHERE rank > 2;
```

This is because with no row multiplication being done, the rank value is always 1. The correct way to obtain this result is longer but cannot use the ROWNUM column:

```
SELECT mother FROM xchild2 GROUP BY mother HAVING COUNT(child) > 2;
```

## XCOL tables based on specified views

Instead of specifying a source table name via the TABNAME option, it is possible to retrieve data from a “view” whose definition is given in a new option SRCDEF . For instance:

```
CREATE TABLE xsvars ENGINE=CONNECT TABLE_TYPE=XCOL
SRCDEF='SHOW VARIABLES LIKE "optimizer_switch"'
OPTION_LIST='Colname=Value';
```

Then, for instance:

```
SELECT value FROM xsvars LIMIT 10;
```

This will display something like:

value
index_merge=on
index_merge_union=on
index_merge_sort_union=on
index_merge_intersection=on
index_merge_sort_intersection=off
engine_condition_pushdown=off
index_condition_pushdown=on
derived_merge=on
derived_with_keys=on
firstmatch=on

Note: All XCOL tables are read only.

## 4.3.7.6.22 CONNECT OCCUR Table Type

Similarly to the

## XCOL

table type,  
OCCUR  
is an extension to the

## PROXY

type when referring to a table or view having several columns containing the same kind of data. It enables having a different view of the table where the data from these columns are put in a single column, eventually causing several rows to be generated from one row of the object table. For example, supposing we have a *pets* table:

name	dog	cat	rabbit	bird	fish
John	2	0	0	0	0
Bill	0	1	0	0	0
Mary	1	1	0	0	0
Lisbeth	0	0	2	0	0
Kevin	0	2	0	6	0
Donald	1	0	0	0	3

We can create an occur table by:

```
create table xpet (
    name varchar(12) not null,
    race char(6) not null,
    number int not null
engine=connect table_type=occur tabname=pets
option_list='OccurCol=number,RankCol=race'
Colist='dog,cat,rabbit,bird,fish';
```

When displaying it by

```
select * from xpet;
```

We will get the result:

name	race	number
John	dog	2
Bill	cat	1
Mary	dog	1
Mary	cat	1
Lisbeth	rabbit	2
Kevin	cat	2
Kevin	bird	6
Donald	dog	1
Donald	fish	3

First of all, the values of the column listed in the Colist option have been put in a unique column whose name is given by the OccurCol option. When several columns have non null (or pseudo-null) values, several rows are generated, with the other normal columns values repeated.

In addition, an optional special column was added whose name is given by the RankCol option. This column contains the name of the source column from which the value of the OccurCol column comes from. It permits here to know the race of the pets whose number is given in *number*.

This table type permit to make queries that would be more complicated to make on the original tables. For instance to know who has more than 1 pet of a kind, you can simply ask:

```
select * from xpet where number > 1;
```

You will get the result:

name	race	number
John	dog	2

Lisbeth	rabbit	2
Kevin	cat	2
Kevin	bird	6
Donald	fish	3

**Note 1:** Like for [XCOL tables](#), no row multiplication for queries not implying the Occur column.

**Note 2:** Because the OccurCol was declared "not null" no rows were generated for null or pseudo-null values of the column list. If the OccurCol is declared as nullable, rows are also generated for columns containing null or pseudo-null values.

Occur tables can be also defined from views or source definition. Also, CONNECT is able to generate the column definitions if not specified. For example:

```
create table ocsrc engine=connect table_type=occur
colist='january,february,march,april,may,june,july,august,september,
october,november,december' option_list='rankcol=month,occurcol=day'
srcdef='select ''Foo'' name, 8 january, 7 february, 2 march, 1 april,
8 may, 14 june, 25 july, 10 august, 13 september, 22 october, 28
november, 14 december';
```

This table is displayed as:

name	month	day
Foo	january	8
Foo	february	7
Foo	march	2
Foo	april	1
Foo	may	8
Foo	june	14
Foo	july	25
Foo	august	10
Foo	september	13
Foo	october	22
Foo	november	28
Foo	december	14

## 4.3.7.6.23 CONNECT PIVOT Table Type

### Contents

1. [Using the PIVOT Tables Type](#)
2. [Restricting the Columns in a Pivot Table](#)
3. [PIVOT Create Table Syntax](#)
  1. [Additional Access Options](#)
4. [Defining a Pivot Table](#)
  1. [Defining a Pivot Table from a Source Table](#)
  2. [Directly Defining the Source of a Pivot Table in SQL](#)
5. [Specifying the Columns Corresponding to the Pivot Column](#)
6. [Pivoting Big Source Tables](#)

This table type can be used to transform the result of another table or view (called the source table) into a pivoted table along “pivot” and “facts” columns. A pivot table is a great reporting tool that sorts and sums (by default) independent of the original data layout in the source table.

For example, let us suppose you have the following “Expenses” table:

Who	Week	What	Amount
Joe	3	Beer	18.00
Beth	4	Food	17.00
Janet	5	Beer	14.00
Joe	3	Food	12.00

Joe	4	Beer	19.00
Janet	5	Car	12.00
Joe	3	Food	19.00
Beth	4	Beer	15.00
Janet	5	Beer	19.00
Joe	3	Car	20.00
Joe	4	Beer	16.00
Beth	5	Food	12.00
Beth	3	Beer	16.00
Joe	4	Food	17.00
Joe	5	Beer	14.00
Janet	3	Car	19.00
Joe	4	Food	17.00
Beth	5	Beer	20.00
Janet	3	Food	18.00
Joe	4	Beer	14.00
Joe	5	Food	12.00
Janet	3	Beer	18.00
Janet	4	Car	17.00
Janet	5	Food	12.00

Pivoting the table contents using the 'Who' and 'Week' fields for the left columns, and the 'What' field for the top heading and summing the 'Amount' fields for each cell in the new table, gives the following desired result:

Who	Week	Beer	Car	Food
Beth	3	16.00	0.00	0.00
Beth	4	15.00	0.00	17.00
Beth	5	20.00	0.00	12.00
Janet	3	18.00	19.00	18.00
Janet	4	0.00	17.00	0.00
Janet	5	33.00	12.00	12.00
Joe	3	18.00	20.00	31.00
Joe	4	49.00	0.00	34.00
Joe	5	14.00	0.00	12.00

Note that SQL enables you to get the same result presented differently by using the "group by" clause, namely:

```
select who, week, what, sum(amount) from expenses
group by who, week, what;
```

However there is no way to get the pivoted layout shown above just using SQL. Even using embedded SQL programming for some DBMS is not quite simple and automatic.

The Pivot table type of CONNECT makes doing this much simpler.

## Using the PIVOT Tables Type

To get the result shown in the example above, just define it as a new table with the statement:

```
create table pivex
engine=connect table_type=pivot tablename=expenses;
```

You can now use it as any other table, for instance to display the result shown above, just say:

```
select * from pivex;
```

The CONNECT implementation of the PIVOT table type does much of the work required to transform the source table:

1. Finding the “Facts” column, by default the last column of the source table. Finding “Facts” or “Pivot” columns work only for table based pivot tables. They do not for view or srcdef based pivot tables, for which they must be explicitly specified.
2. Finding the “Pivot” column, by default the last remaining column.
3. Choosing the aggregate function to use, “SUM” by default.
4. Constructing and executing the “Group By” on the “Facts” column, getting its result in memory.
5. Getting all the distinct values in the “Pivot” column and defining a “Data” column for each.
6. Spreading the result of the intermediate memory table into the final table.

The source table “Pivot” column must not be nullable (there are no such things as a “null” column) The creation will be refused even if this nullable column actually does not contain null values.

If a different result is desired, Create Table options are available to change the defaults used by Pivot. For instance if we want to display the average expense for each person and product, spread in columns for each week, use the following statement:

```
create table pivex2  
engine=connect table_type=pivot tablename=expenses  
option_list='PivotCol=Week,Function=AVG';
```

Now saying:

```
select * from pivex2;
```

Will display the resulting table:

Who	What	3	4	5
Beth	Beer	16.00	15.00	20.00
Beth	Food	0.00	17.00	12.00
Janet	Beer	18.00	0.00	16.50
Janet	Car	19.00	17.00	12.00
Janet	Food	18.00	0.00	12.00
Joe	Beer	18.00	16.33	14.00
Joe	Car	20.00	0.00	0.00
Joe	Food	15.50	17.00	12.00

## Restricting the Columns in a Pivot Table

Let us suppose that we want a Pivot table from expenses summing the expenses for all people and products whatever week it was bought. We can do this just by removing from the pivex table the week column from the column list.

```
alter table pivex drop column week;
```

The result we get from the new table is:

Who	Beer	Car	Food
Beth	51.00	0.00	29.00
Janet	51.00	48.00	30.00
Joe	81.00	20.00	77.00

Note: Restricting columns is also needed when the source table contains extra columns that should not be part of the pivot table. This is true in particular for key columns that prevent a proper grouping.

## PIVOT Create Table Syntax

The Create Table statement for PIVOT tables uses the following syntax:

```

create table pivot_table_name
[(column_definition)]
engine=CONNECT table_type=PIVOT
{tablename='source_table_name' | srcdef='source_table_def'}
[option_list='pivot_table_option_list'];

```

The column definition has two sets of columns:

1. A set of columns belonging to the source table, not including the “facts” and “pivot” columns.
2. “Data” columns receiving the values of the aggregated “facts” columns named from the values of the “pivot” column. They are indicated by the “flag” option.

The **options** and **sub-options** available for Pivot tables are:

Option	Type	Description
<b>Tabname</b>	<i>[DB.]Name</i>	The name of the table to “pivot”. If not set SrcDef must be specified.
<b>SrcDef</b>	<i>SQL_statement</i>	The statement used to generate the intermediate mysql table.
<b>DBname</b>	<i>name</i>	The name of the database containing the source table. Defaults to the current database.
<b>Function*</b>	<i>name</i>	The name of the aggregate function used for the data columns, SUM by default.
<b>PivotCol*</b>	<i>name</i>	Specifies the name of the Pivot column whose values are used to fill the “data” columns having the flag option.
<b>FncCol*</b>	<i>[func([name])]</i>	Specifies the name of the data “Facts” column. If the form func(name) is used, the aggregate function name is set to func.
<b>Groupby*</b>	<i>Boolean</i>	Set it to True (1 or Yes) if the table already has a GROUP BY format.
<b>Accept*</b>	<i>Boolean</i>	To accept non matching Pivot column values.

- : These options must be specified in the OPTION\_LIST.

## Additional Access Options

There are four cases where pivot must call the server containing the source table or on which the SrcDef statement must be executed:

1. The source table is not a CONNECT table.
2. The SrcDef option is specified.
3. The source table is on another server.
4. The columns are not specified.

By default, pivot tries to call the currently used server using host=localhost, user=root not using password, and port=3306. However, this may not be what is needed, in particular if the local root user has a password in which case you can get an “access denied” error message when creating or using the pivot table.

Specify the host, user, password and/or port options in the option\_list to override the default connection options used to access the source table, get column specifications, execute the generated group by or SrcDef query.

## Defining a Pivot Table

There are principally two ways to define a PIVOT table:

1. From an existing table or view.
2. Directly giving the SQL statement returning the result to pivot.

### Defining a Pivot Table from a Source Table

The **tablename** standard table option is used to give the name of the source table or view.

For tables, the internal Group By will be internally generated, except when the GROUPBY option is specified as true. Do it only when the table or view has a valid GROUP BY format.

### Directly Defining the Source of a Pivot Table in SQL

Alternatively, the internal source can be directly defined using the **SrcDef** option that must have the proper group by format.

As we have seen above, a proper Pivot Table is made from an internal intermediate table resulting from the execution of a

GROUP BY

statement. In many cases, it is simpler or desirable to directly specify this when creating the pivot table. This may be because the source is the result of a complex process including filtering and/or joining tables.

To do this, use the **SrcDef** option, often replacing all other options. For instance, suppose that in the first example we are only interested in weeks 4 and 5. We could of course display it by:

```
select * from pivex where week in (4,5);
```

However, what if this table is a huge table? In this case, the correct way to do it is to define the pivot table as this:

```

create table pivex4
engine=connect table_type=pivot
option_list='PivotCol=what,FncCol=amount'
SrcDef='select who, week, what, sum(amount) from expenses
where week in (4,5) group by who, week, what';

```

If your source table has millions of records and you plan to pivot only a small subset of it, doing so will make a lot of a difference performance wise. In addition, you have entire liberty to use expressions, scalar functions, aliases, join, where and having clauses in your SQL statement. The only constraint is that you are responsible for the result of this statement to have the correct format for the pivot processing.

Using SrcDef also permits to use expressions and/or scalar functions. For instance:

```

create table xpivot (
Who char(10) not null,
What char(12) not null,
First double(8,2) flag=1,
Middle double(8,2) flag=1,
Last double(8,2) flag=1
engine=connect table_type=PIVOT
option_list='PivotCol=wk,FncCol=amnt'
Srcdef='select who, what, case when week=3 then ''First'' when
week=5 then ''Last'' else ''Middle'' end as wk, sum(amount) *
6.56 as amnt from expenses group by who, what, wk';

```

Now the statement:

```
select * from xpivot;
```

Will display the result:

Who	What	First	Middle	Last
Beth	Beer	104.96	98.40	131.20
Beth	Food	0.00	111.52	78.72
Janet	Beer	118.08	0.00	216.48
Janet	Car	124.64	111.52	78.72
Janet	Food	118.08	0.00	78.72
Joe	Beer	118.08	321.44	91.84
Joe	Car	131.20	0.00	0.00
Joe	Food	203.36	223.04	78.72

**Note 1:** to avoid multiple lines having the same fixed column values, it is mandatory in **SrcDef** to place the pivot column at the end of the group by list.

**Note 2:** in the create statement **SrcDef**, it is mandatory to give aliases **to** the columns containing expressions so they are recognized by the other options.

**Note 3:** in the **SrcDef** select statement, quotes must be escaped because the entire statement is passed to MariaDB between quotes. Alternatively, specify it between double quotes.

**Note 4:** We could have left CONNECT do the column definitions. However, because they are defined from the sorted names, the Middle column had been placed at the end of them.

## Specifying the Columns Corresponding to the Pivot Column

These columns must be named from the values existing in the “pivot” column. For instance, supposing we have the following *pet* table:

name	race	number
John	dog	2
Bill	cat	1
Mary	dog	1
Mary	cat	1
Lisbeth	rabbit	2
Kevin	cat	2
Kevin	bird	6

Donald	dog	1
Donald	fish	3

Pivoting it using `race` as the pivot column is done with:

```
create table pivot
engine=connect table_type=pivot tablename=pet
option_list='PivotCol=race,groupby=1';
```

This gives the result:

name	dog	cat	rabbit	bird	fish
John	2	0	0	0	0
Bill	0	1	0	0	0
Mary	1	1	0	0	0
Lisbeth	0	0	2	0	0
Kevin	0	2	0	6	0
Donald	1	0	0	0	3

By the way, does this ring a bell? It shows that in a way PIVOT tables are doing the opposite of what OCCUR tables do.

We can alternatively define specifically the table columns but what happens if the Pivot column contains values that is not matching a “data” column? There are three cases depending on the specified options and flags.

**First case:** If no specific options are specified, this is an error an when trying to display the table. The query will abort with an error message stating that a non-matching value was met. Note that because the column list is established when creating the table, this is prone to occur if some rows containing new values for the pivot column are inserted in the source table. If this happens, you should re-create the table or manually add the new columns to the pivot table.

**Second case:** The accept option was specified. For instance:

```
create table xpivot2 (
name varchar(12) not null,
dog int not null default 0 flag=1,
cat int not null default 0 flag=1)
engine=connect table_type=pivot tablename=pet
option_list='PivotCol=race,groupby=1,Accept=1';
```

No error will be raised and the non-matching values will be ignored. This table will be displayed as:

name	dog	cat
John	2	0
Bill	0	1
Mary	1	1
Lisbeth	0	0
Kevin	0	2
Donald	1	0

**Third case:** A “dump” column was specified with the flag value equal to 2. All non-matching values will be added in this column. For instance:

```
create table xpivot (
name varchar(12) not null,
dog int not null default 0 flag=1,
cat int not null default 0 flag=1,
other int not null default 0 flag=2)
engine=connect table_type=pivot tablename=pet
option_list='PivotCol=race,groupby=1';
```

This table will be displayed as:

name	dog	cat	other
John	2	0	0
Bill	0	1	0
Mary	1	1	0

Lisbeth	0	0	2
Kevin	0	2	6
Donald	1	0	3

It is a good idea to provide such a “dump” column if the source table is prone to be inserted new rows that can have a value for the pivot column that did not exist when the pivot table was created.

## Pivoting Big Source Tables

This may sometimes be risky. If the pivot column contains too many distinct values, the resulting table may have too many columns. In all cases the process involved, finding distinct values when creating the table or doing the group by when using it, can be very long and sometimes can fail because of exhausted memory.

Restrictions by a where clause should be applied to the source table when creating the pivot table rather than to the pivot table itself. This can be done by creating an intermediate table or using as source a view or a srcdef option.

All PIVOT tables are read only.

### 4.3.7.6.24 CONNECT TBL Table Type: Table List

#### Contents

1. Sub-tables of not CONNECT engines
2. Using the TABID special column
3. Parallel Execution

This type allows defining a table as a list of tables of any engine and type. This is more flexible than multiple tables that must be all of the same file type. This type does, but is more powerful than, what is done with the **MERGE** engine.

The list of the columns of the TBL table may not necessarily include all the columns of the tables of the list. If the name of some columns is different in the sub-tables, the column to use can be specified by its position given by the

FLAG

option of the column. If the

ACCEPT

option is set to true (Y or 1) columns that do not exist in some of the sub-tables are accepted and their value will be null or pseudo-null (this depends on the nullability of the column) for the tables not having this column. The column types can also be different and an automatic conversion will be done if necessary.

**Note:** If not specified, the column definitions are retrieved from the first table of the table list.

The default database of the sub-tables is the current database or if not, can be specified in the DBNAME option. For the tables that are not in the default database, this can be specified in the table list. For instance, to create a table based on the French table *employe* in the current database and on the English table *employee* of the *db2* database, the syntax of the create statement can be:

```
CREATE TABLE allemp (
  SERIALNO char(5) NOT NULL flag=1,
  NAME varchar(12) NOT NULL flag=2,
  SEX smallint(1),
  TITLE varchar(15) NOT NULL flag=3,
  MANAGER char(5) DEFAULT NULL flag=4,
  DEPARTMENT char(4) NOT NULL flag=5,
  SECRETARY char(5) DEFAULT NULL flag=6,
  SALARY double(8,2) NOT NULL flag=7)
ENGINE=CONNECT table_type=TBL
table_list='employe,db2.employee' option_list='Accept=1';
```

The search for columns in sub tables is done by name and, if they exist with a different name, by their position given by a not null

FLAG

option. Column sex exists only in the English table (

FLAG

is

0

). Its values will null value for the French table.

For instance, the query:

```
select name, sex, title, salary from allemp where department = 318;
```

Can reply:

NAME	SEX	TITLE	SALARY
BARBOUD	NULL	VENDEUR	9700.00
MARCHANT	NULL	VENDEUR	8800.00
MINIARD	NULL	ADMINISTRATIF	7500.00
POUPIN	NULL	INGENIEUR	7450.00
ANTERPE	NULL	INGENIEUR	6850.00
LOULOUTE	NULL	SECRETAIRE	4900.00
TARTINE	NULL	OPERATRICE	2800.00
WERTHER	NULL	DIRECTEUR	14500.00
VOITURIN	NULL	VENDEUR	10130.00
BANCROFT	2	SALESMAN	9600.00
MERCHANT	1	SALESMAN	8700.00
SHRINKY	2	ADMINISTRATOR	7500.00
WALTER	1	ENGINEER	7400.00
TONGHO	1	ENGINEER	6800.00
HONEY	2	SECRETARY	4900.00
PLUMHEAD	2	TYPIST	2800.00
WERTHER	1	DIRECTOR	14500.00
WHEELFOR	1	SALESMAN	10030.00

The first 9 rows, coming from the French table, have a null for the sex value. They would have 0 if the sex column had been created NOT NULL.

## Sub-tables of not CONNECT engines

Sub-tables are accessed as

PROXY

tables. For not CONNECT sub-tables that are accessed via the MySQL API, it is possible like with

PROXY

to change the MYSQL default options. Of course, this will apply to all not CONNECT tables of the list.

## Using the TABID special column

The TABID special column can be used to see from which table the rows come from and to restrict the access to only some of the sub-tables.

Let us see the following example where t1 and t2 are MyISAM tables similar to the ones given in the

MERGE

description:

```
create table xt1 (
  a int(11) not null,
  message char(20))
engine=CONNECT table_type=MYSQL tablename='t1'
option_list='database=test,user=root';

create table xt2 (
  a int(11) not null,
  message char(20))
engine=CONNECT table_type=MYSQL tablename='t2'
option_list='database=test,user=root';

create table toto (
  tablename char(8) not null special='TABID',
  a int(11) not null,
  message char(20))
engine=CONNECT table_type=TBL table_list='xt1,xt2';

select * from total;
```

The result returned by the SELECT statement is:

tablename	a	message
xt1	1	Testing
xt1	2	table
xt1	3	t1
xt2	1	Testing
xt2	2	table
xt2	3	t2

Now if you send the query:

```
select * from total where tablename = 'xt2';
```

CONNECT will analyze the where clause and only read the xt1 table. This can save time if you want to retrieve only a few sub-tables from a TBL table containing many sub-tables.

## Parallel Execution

Parallel Execution is currently unavailable until some bugs are fixed.

When the sub-tables are located on different servers, it is possible to execute the remote queries simultaneously instead of sequentially. To enable this, set the thread option to yes.

Additional options available for this table type:

Option	Description
Maxerr	The max number of missing tables in the table list before an error is raised. Defaults to 0.
Accept	If true, missing columns are accepted and return null values. Defaults to false.
Thread	If true, enables parallel execution of remote sub-tables.

These options can be specified in the

OPTION\_LIST

## 4.3.7.6.25 CONNECT - Using the TBL and MYSQL Table Types Together

### Contents

1. [Remotely executing complex queries](#)
2. [Providing a list of servers](#)

Used together, these types lift all the limitations of the [FEDERATED](#) and [MERGE](#) engines.

**MERGE:** Its limitation is obvious, the merged tables must be identical [MyISAM](#) tables, and MyISAM is not even the default engine for MariaDB. However, [TBL](#) accesses a collection of CONNECT tables, but because these tables can be user specified or internally created [MYSQL](#) tables, there is no limitation to the type of the tables that can be merged.

TBL is also much more flexible. The merged tables must not be "identical", they just should have the columns defined in the TBL table. If the type of one column in a merged table is not the one of the corresponding column of the TBL table, the column value will be converted. As we have seen, if one column of the TBL table of the TBL column does not exist in one of the merged table, the corresponding value will be set to null. If columns in a sub-table have a different name, they can be accessed by position using the FLAG column option of CONNECT.

However, one limitation of the TBL type regarding MERGE is that TBL tables are currently read-only; INSERT is not supported by TBL. Also, keep using MERGE to access a list of identical MyISAM tables because it will be faster, not passing by the MySQL API.

**FEDERATED(X):** The main limitation of FEDERATED is to access only MySQL/MariaDB tables. The MYSQL table type of CONNECT has the same limitation but CONNECT provides the [ODBC table type](#) and [JDBC table type](#) that can access tables of any RDBS providing an ODBC or JDBC driver (including MySQL even it is not really useful!)

Another major limitation of FEDERATED is to access only one table. By combining TBL and MYSQL tables, CONNECT enables to access a collection of local or remote tables as one table. Of course the sub-tables can be on different servers. With one SELECT statement, a company manager will be able to interrogate results coming from all of his subsidiary computers. This is great for distribution, banking, and many other industries.

### Remotely executing complex queries

Many companies or administrations must deal with distributed information. CONNECT enables to deal with it efficiently without having to copy it to a centralized database. Let us suppose we have on some remote network machines  $m_1, m_2, \dots, m_n$  some information contained in two tables  $t_1$  and  $t_2$ .

Suppose we want to execute on all servers a query such as:

```
select c1, sum(c2) from t1 a, t2 b where a.id = b.id group by c1;
```

This raises many problems. Returning the column values of the  $t_1$  and  $t_2$  tables from all servers can be a lot of network traffic. The group by on the possibly huge resulting tables can be a long process. In addition, the join on the  $t_1$  and  $t_2$  tables may be relevant only if the joined tuples belong to the same machine, obliging to add a condition on an additional tabid or servid special column.

All this can be avoided and optimized by forcing the query to be locally executed on each server and retrieving only the small results of the group by queries. Here is how to do it. For each remote machine, create a table that will retrieve the locally executed query. For instance for  $m_1$ :

```
create table rt1 engine=connect option_list='host=m1'
srcdef='select c1, sum(c2) as sc2 from t1 a, t2 b where a.id = b.id group by c1';
```

Note the alias for the functional column. An alias would be required for the  $c1$  column if its name was different on some machines. The  $t1$  and  $t2$  table names can also be eventually different on the remote machines. The true names must be used in the

SRCDEF

parameter. This will create a set of tables with two columns named  $c1$  and  $sc2$ <sup>[1]</sup>.

Then create the table that will retrieve the result of all these tables:

```
create table rtall engine=connect table_type=tbl
table_list='rt1,rt2,...,rtn' option_list='thread=yes';
```

Now you can retrieve the desired result by:

```
select c1, sum(sc2) from rtall;
```

Almost all the work will be done on the remote machines, simultaneously thanks to the thread option, making this query super-fast even on big tables placed on many remote machines.

Thread is currently experimental. Use it only for test and report any malfunction on [JIRA](#).

## Providing a list of servers

An interesting case is when the query to run on remote machines is the same for all of them. It is then possible to avoid declaring all sub-tables. In this case, the table list option will be used to specify the list of servers the

SRCDEF

query must be sent. This will be a list of URL's and/or Federated server names.

For instance, supposing that federated servers  $srv_1, srv_2, \dots, srv_n$  were created for all remote servers, it will be possible to create a  $tbl$  table allowing getting the result of a query executed on all of them by:

```
create table qall [column definition]
engine=connect table_type=TBL srcdef='a query'
table_list='srv1,srv2,...,srvn' [option_list='thread=yes'];
```

For instance:

```
create table verall engine=connect table_type=TBL srcdef='select @@version' table_list=',server_one';
select * from verall;
```

This reply:

@@version
10.0.3-MariaDB-debug
10.0.2-MariaDB

Here the server list specifies a void server corresponding to the local running MariaDB and a federated server named `server_one`.

1. ↑ To generate the columns from the

SRCDEF

query, CONNECT must execute it. This will make sure it is ok. However, if the remote server is not connected yet, or the remote table not existing yet, you can alternatively specify the columns in the create table statement.

## 4.3.7.6.26 CONNECT Table Types - Special "Virtual" Tables

### Contents

- 1. [DIR Type](#)
  - 1. [The Subdir option](#)
  - 2. [The Nodir option \(Windows\)](#)
- 2. [Windows Management Instrumentation](#)
  - Table Type "WMI"
    - 1. [Getting column information](#)
    - 2. [Performance Consideration](#)
    - 3. [Syntax of WMI queries](#)
  - 3. [MAC Address Table Type "MAC"](#)

The special table types supported by CONNECT are the Virtual table type ( VIR - introduced in [MariaDB 10.0.15](#) ), Directory Listing table type (DIR), the Windows Management Instrumentation Table Type (WMI), and the "Mac Address" type (MAC).

These tables are "virtual tables", meaning they have no physical data but rather produce result data using specific algorithms. Note that this is close to what Views are, so they could be regarded as special views.

### DIR Type

A table of type DIR returns a list of file name and description as a result set. To create a DIR table, use a Create Table statement such as:

```
create table source (
    DRIVE char(2) NOT NULL,
    PATH varchar(256) NOT NULL,
    FNAME varchar(256) NOT NULL,
    FTYPE char(4) NOT NULL,
    SIZE double(12,0) NOT NULL flag=5,
    MODIFIED datetime NOT NULL)
engine=CONNECT table_type=DIR file_name='..\*.cc';
```

When used in a query, the table returns the same file information listing than the system "DIR

```
*.cc
" statement would return if executed in the same current directory (here supposedly ..\)
```

For instance, the query:

```
select fname, size, modified from source
where fname like '%handler%';
```

Displays:

fname	size	modified
handler	152177	2011-06-13 18:08:29
sql_handler	25321	2011-06-13 18:08:31

**Note:** the important item in this table is the flag option value (set sequentially from 0 by default) because it determines which particular information item is returned in the column:

Flag value	Information
0	The disk drive (Windows)
1	The file path
2	The file name
3	The file type
4	The file attribute
5	The file size
6	The last write access date
7	The last read access date
8	The file creation date

## The Subdir option

When specified in the create table statement, the subdir option indicates to list, in addition to the files contained in the specified directory, all the files verifying the filename pattern that are contained in sub-directories of the specified directory. For instance, using:

```
create table data (
    PATH varchar(256) NOT NULL flag=1,
    FNAME varchar(256) NOT NULL,
    FTYPE char(4) NOT NULL,
    SIZE double(12,0) NOT NULL flag=5)
engine=CONNECT table_type=DIR file_name='*.frm'
option_list='subdir=1';

select path, count(*), sum(size) from data group by path;
```

You will get the following result set showing how many tables are created in the MariaDB databases and what is the total length of the FRM files:

path	count(*)	sum(size)
\CommonSource\mariadb-5.2.7\sql\data\connect\	30	264469
\CommonSource\mariadb-5.2.7\sql\data\mysql\	23	207168
\CommonSource\mariadb-5.2.7\sql\data\test\	22	196882

## The Nodir option (Windows)

The Boolean Nodir option can be set to false (0 or no) to add directories that match the file name pattern from the listed files (it is true by default). This is an addition to CONNECT version 1.6. Previously, directory names matching pattern were listed on Windows. Directories were and are never listed on Linux.

Note: The way file names are retrieved makes positional access to them impossible. Therefore, DIR tables cannot be indexed or sorted when it is done using positions.

Be aware, in particular when using the subdir option, that queries on DIR tables are slow and can last almost forever if made on a directory that contains a great number of files in it and its sub-directories.

dir tables can be used to populate a list of files used to create a multiple=2 table. However, this is not as useful as it was when the multiple 3 did not exist.

## Windows Management Instrumentation Table Type “WMI”

**Note:** This table type is available on Windows only.

WMI provides an operating system interface through which instrumented components provide information. Some Microsoft tools to retrieve information through WMI are the WMIC console command and the WMI CMI Studio application.

The CONNECT WMI table type enables administrators and operators not capable of scripting or programming on top of WMI to enjoy the benefit of WMI without even learning about it. It permits to present this information as tables that can be queried, transformed, copied in documents or other tables.

To create a WMI table displaying information coming from a WMI provider, you must provide the namespace and the class name that characterize the information you want to retrieve. The best way to find them is to use the WMI CIM Studio that have tools to browse namespaces and classes and that can display the names of the properties of that class.

The column names of the tables must be the names (case insensitive) of the properties you want to retrieve. For instance:

```
create table alias (
    friendlyname char(32) not null,
    target char(50) not null)
engine=CONNECT table_type='WMI'
option_list='Namespace=root\cli,Class=Msft_CliAlias';
```

WMI tables returns one row for each instance of the related information. The above example is handy to get the class equivalent of the alias of the WMIC command and also to have a list of many classes commonly used.

Because most of the useful classes belong to the 'root\cimv2' namespace, this is the default value for WMI tables when the namespace is not specified. Some classes have many properties whose name and type may not be known when creating the table. To find them, you can use the WMI CMI Studio application but this will be rarely required because CONNECT is able to retrieve them.

Actually, the class specification also has default values for some namespaces. For the 'root\cli' namespace the class name defaults to 'Msft\_CliAlias' and for the 'root\_cimv2' namespace the class default value is 'Win32\_ComputerSystemProduct'. Because many class names begin with 'Win32\_' it is not necessary to say it and specifying the class as 'Product' will effectively use class 'Win32\_Product'.

For example if you define a table as:

```
create table CSPROD engine=CONNECT table_type='WMI';
```

It will return the information on the current machine, using the class ComputerSystemProduct of the CIMV2 namespace. For instance:

```
select * from csprod;
```

Will return a result such as:

Column	Row 1
Caption	Computer system product
Description	Computer system product
IdentifyingNumber	LXAP50X32982327A922300
Name	Aspire 8920
SKUNumber	
UUID	00FC523D-B8F7-DC12-A70E-00B0D1A46136
Vendor	Acer
Version	Aspire 8920

**Note:** This is a transposed display that can be obtained with some GUI.

## Getting column information

An issue, when creating a WMI table, is to make its column definition. Indeed, even when you know the namespace and the class for the wanted information, it is not easy to find what are the names and types of its properties. However, because CONNECT can retrieve this information from the WMI provider, you can simply omit defining columns and CONNECT will do the job.

Alternatively, you can get this information using a catalog table (see below).

## Performance Consideration

Some WMI providers can be very slow to answer. This is not an issue for those that return few object instances, such as the ones returning computer, motherboard, or Bios information. They generally return only one row (instance). However, some can return many rows, in particular the "CIM\_DataFile" class. This is why care must be taken about them.

Firstly, it is possible to limit the allocated result size by using the 'Estimate' create table option. To avoid result truncation, CONNECT allocates a result of 100 rows that is enough for almost all tables. The 'Estimate' option permits to reduce this size for all classes that return only a few rows, and in some rare case to increase it to avoid truncation.

However, it is not possible to limit the time taken by some WMI providers to answer, in particular the CIM\_DATAFILE class. Indeed the Microsoft documentation says about it:

"Avoid enumerating or querying for all instances of CIM\_DataFile on a computer because the volume of data is likely to either affect performance or cause the computer to stop responding."

Sure enough, even a simple query such as:

```
select count(*) from cim where drive = 'D:' and path like '\\MariaDB\\%';
```

is prone to last almost forever (probably due to the LIKE clause). This is why, when not asking for some specific items, you should consider using the DIR table type instead.

## Syntax of WMI queries

Queries to WMI providers are done using the WQL language, not the SQL language. CONNECT does the job of making the WQL query. However, because of the restriction of the WQL syntax, the WHERE clause will be generated only when respecting the following restrictions:

1. No function.
2. No comparison between two columns.
3. No expression (currently a CONNECT restriction)
4. No BETWEEN and IN predicates.

Filtering with WHERE clauses not respecting these conditions will still be done by MariaDB only, except in the case of CIM\_Datafile class for the reason given above.

However, there is one point that is not covered yet, the syntax used to specify dates in queries. WQL does not recognize dates as number items but translates them to its internal format dates specified as text. Many formats are recognized as described in the Microsoft documentation but only one is useful because common to WQL and MariaDB SQL. Here is an example of a query on a table named "cim" created by:

```

create table cim (
    Name varchar(255) not null,
    LastModified datetime not null)
engine=CONNECT table_type='WMI'
option_list='class=CIM_DataFile,estimate=5000';

```

The date must be specified with the format in which CIM DATETIME values are stored (WMI uses the date and time formats defined by the Distributed Management Task Force).

```

select * from cim where drive = 'D:' and path = '\\PlugDB\\Bin\\'
    and lastmodified > '2012041500000.00000+120';

```

This syntax must be strictly respected. The text has the format:

```
yyyymmddHHMMSS.mmmmmmsUUU
```

It is: year, month, day, hour, minute, second, millisecond, and signed minute deviation from UTC . This format is locale-independent so you can write a query that runs on any machine.

**Note 1:** The WMI table type is available only in Windows versions of CONNECT.

**Note 2:** WMI tables are read only.

**Note 3:** WMI tables are not indexable.

**Note 4:** WMI consider all strings as case insensitive.

## MAC Address Table Type “MAC”

**Note:** This table type is available on Windows only.

This type is used to display various general information about the computer and, in particular, about its network cards. To create such a table, the syntax to use is:

```

create table tablename (column definition)
engine=CONNECT table_type=MAC;

```

Column names can be freely chosen because their signification, i.e. the values they will display, comes from the specified Flag option. The valid values for Flag are:

Flag	Valeur	Type
1	Host name	varchar(132)
2	Domain	varchar(132)
3	DNS address	varchar(24)
4	Node type	int(1)
5	Scope ID	varchar(256)
6	Routing	int(1)
7	Proxy	int(1)
8	DNS	int(1)
10	Name	varchar(260)
11	Description	varchar(132)
12	MAC address	char(24)
13	Type	int(3)
14	DHCP	int(1)
15	IP address	char(16)
16	SUBNET mask	char(16)
17	GATEWAY	char(16)
18	DHCP server	char(16)
19	Have WINS	int(1)

20	Primary WINS	char(16)
21	Secondary WINS	char(16)
22	Lease obtained	datetime
23	Lease expires	datetime

**Note:** The information of columns having a Flag value less than 10 are unique for the computer, the other ones are specific to the network cards of the computer.

For instance, you can define the table `macaddr` as:

```
create table macaddr (
  Host varchar(132) flag=1,
  Card varchar(132) flag=11,
  Address char(24) flag=12,
  IP char(16) flag=15,
  Gateway char(16) flag=17,
  Lease datetime flag=23)
engine=CONNECT table_type=MAC;
```

If you execute the query:

```
select host, address, ip, gateway, lease from macaddr;
```

It will return, for example:

Host	Address	IP	Gateway	Lease
OLIVIER	00-A0-D1-A4-61-36	0.0.0.0	0.0.0.0	1970-01-01 00:00:00
OLIVIER	00-1D-E0-9B-90-0B	192.168.0.10	192.168.0.254	2011-09-18 10:28:5

## 4.3.7.6.27 CONNECT Table Types - VIR

### Contents

1. [VIR Type](#)
2. [Displaying constants or expressions](#)
3. [Generating a Table filled with constant values](#)
4. [VIR tables vs. SEQUENCE tables](#)

## VIR Type

A VIR table is a virtual table having only Special or Virtual columns. Its only property is its “size”, or cardinality, meaning the number of virtual rows it contains. It is created using the syntax:

```
CREATE TABLE name [coldef] ENGINE=CONNECT TABLE_TYPE=VIR
[BLOCK_SIZE=n];
```

The optional

`BLOCK_SIZE`

option gives the size of the table, defaulting to 1 if not specified. When its columns are not specified, it is almost equivalent to a [SEQUENCE](#) table “seq\_1\_to\_Size”.

### Displaying constants or expressions

Many DBMS use a no-column one-line table to do this, often call “dual”. MySQL and MariaDB use syntax where no table is specified. With CONNECT, you can achieve the same purpose with a virtual table, with the noticeable advantage of being able to display several lines. For example:

```
create table virt engine=connect table_type=VIR block_size=10;
select concat('The square root of ', n, ' is') what,
round(sqrt(n),16) value from virt;
```

This will return:

what	value
The square root of 1 is	1.0000000000000000
The square root of 2 is	1.4142135623730951

The square root of 3 is	1.7320508075688772
The square root of 4 is	2.0000000000000000
The square root of 5 is	2.2360679774997898
The square root of 6 is	2.4494897427831779
The square root of 7 is	2.6457513110645907
The square root of 8 is	2.8284271247461903
The square root of 9 is	3.0000000000000000
The square root of 10 is	3.1622776601683795

What happened here? First of all, unlike Oracle “dual” tableS that have no columns, a MariaDB table must have at least one column. By default, CONNECT creates VIR tables with one special column. This can be seen with the SHOW CREATE TABLE statement:

```
CREATE TABLE `virt` (
`n` int(11) NOT NULL `SPECIAL`=ROWID,
PRIMARY KEY (`n`)
) ENGINE=CONNECT DEFAULT CHARSET=latin1 `TABLE_TYPE`='VIR'
`BLOCK_SIZE`=10
```

This special column is called “n” and its value is the row number starting from 1. It is purely a virtual table and no data file exists corresponding to it and to its index. It is possible to specify the columns of a VIR table but they must be CONNECT special columns or virtual columns. For instance:

```
create table virt2 (
n int key not null special=rowid,
sig1 bigint as ((n*(n+1))/2) virtual,
sig2 bigint as(((2*n+1)*(n+1)*n)/6) virtual)
engine=connect table_type=VIR block_size=10000000;
select * from virt2 limit 995, 5;
```

This table shows the sum and the sum of the square of the n first integers:

n	sig1	sig2
996	496506	329845486
997	497503	330839495
998	498501	331835499
999	499500	332833500
1000	500500	333833500

Note that the size of the table can be made very big as there no physical data. However, the result should be limited in the queries. For instance:

```
select * from virt2 where n = 1664510;
```

Such a query could last very long if the rowid column were not indexed. Note that by default, CONNECT declares the “n” column as a primary key. Actually, VIR tables can be indexed but only on the ROWID (or ROWNUM) columns of the table. This is a virtual index for which no data is stored.

## Generating a Table filled with constant values

An interesting use of virtual tables, which often cannot be achieved with a table of any other type, is to generate a table containing constant values. This is easily done with a virtual table. Let us define the table FILLER as:

```
create table filler engine=connect table_type=VIR block_size=5000000;
```

Here we choose a size larger than the biggest table we want to generate. Later if we need a table pre-filled with default and/or null values, we can do for example:

```
create table tp (
id int(6) key not null,
name char(16) not null,
salary float(8,2));
insert into tp select n, 'unknown', NULL from filler where n <= 10000;
```

This will generate a table having 10000 rows that can be updated later when needed. Note that a [SEQUENCE](#) table could have been used here instead of FILLING .

## VIR tables vs. SEQUENCE tables

With just its default column, a VIR table is almost equivalent to a [SEQUENCE](#) table. The syntax used is the main difference, for instance:

```
select * from seq_100_to_150_step_10;
```

can be obtained with a VIR table (of size  $\geq 15$ ) by:

```
select n*10 from vir where n between 10 and 15;
```

Therefore, the main difference is to be able to define the columns of VIR tables. Unfortunately, there are currently many limitations to virtual columns that hopefully should be removed in the future.

## 4.3.7.6.28 CONNECT Table Types - OEM: Implemented in an External LIB

### Contents

- 1. [An OEM Table Example](#)
- 1. [Some Currently Available OEM Table Modules and Subtypes](#)

Although CONNECT provides a rich set of table types, specific applications may need to access data organized in a way that is not handled by its existing foreign data wrappers (FDW). To handle these cases, CONNECT features an interface that enables developers to implement in C++ the required table wrapper and use it as if it were part of the standard CONNECT table type list. CONNECT can use these additional handlers providing the corresponding external module (dll or shared lib) be available.

To create such a table on an existing handler, use a Create Table statement as shown below.

```
create table xtab (column definitions)
engine=CONNECT table_type=OEM module='libname'
subtype='MYTYPE' [standard table options]
Option_list='Myopt=foo';
```

The option module gives the name of the DLL or shared library implementing the OEM wrapper for the table type. This library must be located in the plugin directory like all other plugins or UDF's.

This library must export a function `GetMYTYPE`. The option subtype enables CONNECT to have the name of the exported function and to use the new table type. Other options are interpreted by the OEM type and can also be specified within the `option_list` option.

Column definitions can be unspecified only if the external wrapper is able to return this information. For this it must export a function `ColMYTYPE` returning these definitions in a format acceptable by the CONNECT discovery function.

Which and how options must be specified and the way columns must be defined may vary depending on the OEM type used and should be documented by the OEM type implementer(s).

## An OEM Table Example

The OEM table REST described in [Adding the REST Feature as a Library Called by an OEM Table](#) permits using REST-like tables with MariaDB binary distributions containing but not enabling the [REST table type](#)

Of course, the mongo (dll or so) exporting the `GetREST` and `colREST` functions must be available in the plugin directory for all this to work.

## Some Currently Available OEM Table Modules and Subtypes

Module	Subtype	Description
libhello	HELLO	A sample OEM wrapper displaying a one line table saying "Hello world"
mongo	MONGO	Enables using tables based on MongoDB collections.
Tabfic	FIC	Handles files having the Windev HyperFile format.
Tabofx	OFC	Handles Open Financial Connectivity files.
Tabofx	QIF	Handles Quicken Interchange Format files.
Cirpack	CRPK	Handles CDR's from Cirpack UTP's.
Tabplg	PLG	Access tables from the PlugDB DBMS.

How to implement an OEM handler is out of the scope of this document.

## 4.3.7.6.29 CONNECT Table Types - Catalog Tables

A catalog table is one that returns information about another table, or data source. It is similar to what MariaDB commands such as

DESCRIBE  
or  
SHOW

do. Applied to local tables, this just duplicates what these commands do, with the noticeable difference that they are tables and can be used inside queries as joined tables or inside sub-selects.

But their main interest is to enable querying the structure of external tables that cannot be directly queried with description commands. Let's see an example:

Suppose we want to access the tables from a Microsoft Access database as an ODBC type table. The first information we must obtain is the list of tables existing in this data source. To get it, we will create a catalog table that will return it extracted from the result set of the SQLTables ODBC function:

```
create table tabinfo (
    table_name varchar(128) not null,
    table_type varchar(16) not null)
engine=connect table_type=ODBC catfunc=tables
Connection='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB';
```

The SQLTables function returns a result set having the following columns:

Field	Data Type	Null	Info Type	Flag Value
Table_Cat	char(128)	NO	FLD_CAT	17
Table_Name	char(128)	NO	FLD_SCHEM	18
Table_Name	char(128)	NO	FLD_NAME	1
Table_Type	char(16)	NO	FLD_TYPE	2
Remark	char(128)	NO	FLD_Rem	5

**Note:** The Info Type and Flag Value are CONNECT interpretations of this result.

Here we could have omitted the column definitions of the catalog table or, as in the above example, chose the columns returning the name and type of the tables. If specified, the columns must have the exact name of the corresponding SQLTables result set, or be given a different name with the matching flag value specification.

(The Table\_Type can be TABLE, SYSTEM TABLE, VIEW, etc.)

For instance, to get the tables we want to use we can ask:

```
select table_name from tabinfo where table_type = 'TABLE';
```

This will return:

table_name
Categories
Customers
Employees
Products
Shippers
Suppliers

Now we want to create the table to access the CUSTOMERS table. Because CONNECT can retrieve the column description of ODBC tables, it is not necessary to specify them in the create table statement:

```
create table Customers engine=connect table_type=ODBC
Connection='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB';
```

However, if we prefer to specify them (to eventually modify them) we must know what the column definitions of that table are. We can get this information with a catalog table. This is how to do it:

```
create table custinfo engine=connect table_type=ODBC
tabname=customers catfunc=columns
Connection='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB';
```

Alternatively it is possible to specify what columns of the catalog table we want:

```
create table custinfo (
    column_name char(128) not null,
    type_name char(20) not null,
    length int(10) not null flag=7,
    prec smallint(6) not null flag=9
    nullable smallint(6) not null)
engine=connect table_type=ODBC tablename=customers
catfunc=columns
Connection='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB';
```

To get the column info:

```
select * from custinfo;
```

which results in this table:

column_name	type_name	length	prec	nullable
CustomerID	VARCHAR	5	0	1
CompanyName	VARCHAR	40	0	1
ContactName	VARCHAR	30	0	1
ContactTitle	VARCHAR	30	0	1
Address	VARCHAR	60	0	1
City	VARCHAR	15	0	1
Region	VARCHAR	15	0	1
PostalCode	VARCHAR	10	0	1
Country	VARCHAR	15	0	1
Phone	VARCHAR	24	0	1
Fax	VARCHAR	24	0	1

Now you can create the CUSTOMERS table as:

```
create table Customers (
    CustomerID varchar(5),
    CompanyName varchar(40),
    ContactName varchar(30),
    ContactTitle varchar(30),
    Address varchar(60),
    City varchar(15),
    Region varchar(15),
    PostalCode varchar(10),
    Country varchar(15),
    Phone varchar(24),
    Fax varchar(24))
engine=connect table_type=ODBC block_size=10
Connection='DSN=MS Access Database;DBQ=C:/Program
Files/Microsoft Office/Office/1033/FPNWIND.MDB';
```

Let us explain what we did here: First of all, the creation of the catalog table. This table returns the result set of an ODBC SQLColumns function sent to the ODBC data source. Columns functions always return a data set having some of the following columns, depending on the table type:

Field	Data Type	Null	Info Type	Flag Value	Returned by
Table_Cat*	char(128)	NO	FLD_CAT	17	ODBC, JDBC
Table_Schema*	char(128)	NO	FLD_SCEM	18	ODBC, JDBC
Table_Name	char(128)	NO	FLD_TABNAME	19	ODBC, JDBC
Column_Name	char(128)	NO	FLD_NAME	1	ALL
Data_Type	smallint(6)	NO	FLD_TYPE	2	ALL
Type_Name	char(30)	NO	FLD_TYPENAME	3	ALL
Column_Size*	int(10)	NO	FLD_PREC	4	ALL

Buffer_Length*	int(10)	NO	FLD_LENGTH	5	ALL
Decimal_Digits*	smallint(6)	NO	FLD_SCALE	6	ALL
Radix	smallint(6)	NO	FLD_RADIX	7	ODBC, JDBC, MYSQL
Nullable	smallint(6)	NO	FLD_NULL	8	ODBC, JDBC, MYSQL
Remarks	char(255)	NO	FLD_REM	9	ODBC, JDBC, MYSQL
Collation	char(32)	NO	FLD_CHARSET	10	MYSQL
Key	char(4)	NO	FLD_KEY	11	MYSQL
Default_value	N.A.		FLD_DEFAULT	12	
Privilege	N.A.		FLD_PRIV	13	
Date_fmt	char(32)	NO	FLD_DATEFMT	15	MYSQL
Xpath/Jxpath	Varchar(256)	NO	FLD_FORMAT	16	XML/JSON

\*\*: These names have changed since earlier versions of CONNECT.

**Note:** ALL includes the ODBC, JDBC, MYSQL, DBF, CSV, PROXY, TBL, XML, JSON, XCOL, and WMI table types. More could be added later.

We chose among these columns the ones that were useful for our create statement, using the flag value when we gave them a different name (case insensitive).

The options used in this definition are the same as the one used later for the actual CUSTOMERS data tables except that:

1. The

TABNAME

option is mandatory here to specify what the queried table name is.

2. The

CATFUNC

option was added both to indicate that this is a catalog table, and to specify that we want column information.

**Note:** If the

TABNAME

option had not been specified, this table would have returned the columns of all the tables defined in the connected data source.

Currently the available

CATFUNC

are:

Function	Specified as:	Applies to table types:
FNC_TAB	<b>tab les</b>	ODBC, JDBC, MYSQL
FNC_COL	<b>col umns</b>	ODBC, JDBC, MYSQL, DBF, CSV, PROXY, XCOL, TBL, WMI
FNC_DSN	<b>datasource s</b> <b>dsn</b> <b>sqldatasource s</b>	ODBC
FNC_DRIVER	<b>driver s</b> <b>sqldrivers</b>	ODBC, JDBC

**Note:** Only the bold part of the function name specification is required.

The

DATASOURCE

and

DRIVERS

functions respectively return the list of available data sources and ODBC drivers available on the system.

The SQLDataSources function returns a result set having the following columns:

Field	Data Type	Null	Info Type	Flag value
Name	varchar(256)	NO	FLD_NAME	1
Description	varchar(256)	NO	FLD_REM	9

To get the data source, you can do for instance:

```
create table datasources (
  engine=CONNECT table_type=ODBC catfunc=DSN;
```

The SQLDrivers function returns a result set having the following columns:

Field	Type	Null	Info Type	Flag value
Description	varchar(128)	YES	FLD_NAME	1
Attributes	varchar(256)	YES	FLD_Rem	9

You can get the driver list with:

```
create table drivers
engine=CONNECT table_type=ODBC catfunc=drivers;
```

## Another example, WMI table

To create a catalog table returning the attribute names of a WMI class, use the same table options as the ones used with the normal WMI table plus the additional option 'catfunc=columns'. If specified, the columns of such a catalog table can be chosen among the following:

Name	Type	Flag	Description
Column_Name	CHAR	1	The name of the property
Data_Type	INT	2	The SQL data type
Type_Name	CHAR	3	The SQL type name
Column_Size	INT	4	The field length in characters
Buffer_Length	INT	5	Depends on the coding
Scale	INT	6	Depends on the type

If you wish to use a different name for a column, set the Flag column option.

For example, before creating the "csprod" table, you could have created the info table:

```
create table CSPRODCOL (
  Column_name char(64) not null,
  Data_Type int(3) not null,
  Type_name char(16) not null,
  Length int(6) not null,
  Prec int(2) not null flag=6
engine=CONNECT table_type='WMI' catfunc=col;
```

Now the query:

```
select * from csprodcol;
```

will display the result:

Column_name	Data_Type	Type_name	Length	Prec
Caption	1	CHAR	255	1
Description	1	CHAR	255	1
IdentifyingNumber	1	CHAR	255	1
Name	1	CHAR	255	1
SKUNumber	1	CHAR	255	1
UUID	1	CHAR	255	1
Vendor	1	CHAR	255	1
Version	1	CHAR	255	1

This can help to define the columns of the matching normal table.

**Note 1:** The column length, for the Info table as well as for the normal table, can be chosen arbitrarily, it just must be enough to contain the returned information.

**Note 2:** The Scale column returns 1 for text columns (meaning case insensitive); 2 for float and double columns; and 0 for other numeric columns.

## Catalog Table result size limit

Because catalog tables are processed like the information retrieved by "Discovery" when table columns are not specified in a Create Table statement, their result set is entirely retrieved and memory allocated.

By default, this allocation is done for a maximum return line number of:

Catfunc	Max lines
Drivers	256
Data Sources	512
Columns	20,000
Tables	10,000

When the number of lines retrieved for a table is more than this maximum, a warning is issued by CONNECT. This is mainly prone to occur with columns (and also tables) with some data sources having many tables when the table name is not specified.

If this happens, it is possible to increase the default limit using the MAXRES option, for instance:

```
create table allcols engine=connect table_type=odbc
connection='DSN=ORACLE_TEST;UID=system;PWD=manager'
option_list='Maxres=110000' catfunc=columns;
```

Indeed, because the entire table result is memorized before the query is executed; the returned value would be limited even on a query such as:

```
select count(*) from allcols;
```

## 4.3.7.7 CONNECT - Security

The use of the CONNECT engine requires the

FILE

privilege for "outward" tables. This should not be an important restriction. The use of CONNECT "outward" tables on a remote server seems of limited interest without knowing the files existing on it and must be protected anyway. On the other hand, using it on the local client machine is not an issue because it is always possible to create locally a user with the

FILE  
privilege.

## 4.3.7.8 CONNECT - OEM Table Example

This is an example showing how an OEM table can be implemented.

The header File

```
my_global.h
:
```

```

/***********************/
/* Definitions needed by the included files. */
/***********************/

#ifndef !defined(MY_GLOBAL_H)
#define MY_GLOBAL_H

typedef unsigned int uint;
typedef unsigned int uint32;
typedef unsigned short ushort;
typedef unsigned long ulong;
typedef unsigned long DWORD;
typedef char *LPSTR;
typedef const char *LPCSTR;
typedef int BOOL;
#if defined(_WIN_)
typedef void *HANDLE;
#else
typedef int HANDLE;
#endif
typedef char *PSZ;
typedef const char *PCSZ;
typedef unsigned char BYTE;
typedef unsigned char uchar;
typedef long long longlong;
typedef unsigned long long ulonglong;
typedef char my_bool;
struct charset_info_st {};
typedef const charset_info_st CHARSET_INFO;
#define FALSE 0
#define TRUE 1
#define Item char
#define MY_MAX(a,b) ((a>b)?(a):(b))
#define MY_MIN(a,b) ((a<b)?(a):(b))
#endif // MY_GLOBAL_H

```

Note: This is a fake

```

my_global.h
that just contains what is useful for the
jmgoem.cpp
source file.

```

The source File

```

jmgoem.cpp
:
```

```

/****************** jmgoem C++ Program Source Code File (.CPP) *****/
/* PROGRAM NAME: jmgoem Version 1.0 */
/* (C) Copyright to the author Olivier BERTRAND 2017 */
/* This program is the Java MONGO OEM module definition. */
/******************/

/******************/
/* Definitions needed by the included files. */
/******************/

#include "my_global.h"

/******************/
/* Include application header files: */
/* global.h is header containing all global declarations. */
/* plgdbsem.h is header containing the DB application declarations. */
/* (x)table.h is header containing the TDBASE declarations. */
/* tabext.h is header containing the TDBEXT declarations. */
/* mongo.h is header containing the MONGO declarations. */
/******************/

#include "global.h"
#include "plgdbsem.h"
#if defined(HAVE_JMGO)
#include "csort.h"
#include "javaconn.h"
#endif // HAVE_JMGO
#include "xtable.h"
#include "tabext.h"
#include "mongo.h"

/******************/
/* These functions are exported from the MONGO library. */
/*****************/

```

```

extern "C" {
    PTABDEF __stdcall GetMONGO(PGLOBAL, void*);
    PQRYRES __stdcall ColMONGO(PGLOBAL, PTOS, void*, char*, char*, bool);
} // extern "C"

/***/
/*  DB static variables.                                */
/***/
int TDB::Tnum;
int DTVAL::Shift;
#if defined(HAVE_JMGO)
int CSORT::Limit = 0;
double CSORT::Lg2 = log(2.0);
size_t CSORT::Cpn[1000] = {0}; /* Precalculated cmpnum values */
#endif defined(HAVE_JAVACONN)
char *JvmPath = NULL;
char *ClassPath = NULL;
char *GetPluginDir(void)
{return "C:/mongo-java-driver/mongo-java-driver-3.4.2.jar";
"C:/MariaDB-10.1/MariaDB/storage/connect/";}
char *GetJavaWrapper(void) {return (char*)"wrappers/Mongo3Interface";}
#else // !HAVE_JAVACONN
HANDLE JAVAConn::LibJvm;           // Handle to the jvm DLL
CRTJVM JAVAConn::CreateJavaVM;
GETJVM JAVAConn::GetCreatedJavaVMs;
#if defined(_DEBUG)
GETDEF JAVAConn::GetDefaultJavaVMIInitArgs;
#endif // _DEBUG
#endif // !HAVE_JAVACONN
#endif // HAVE_JMGO

/***/
/* This function returns a Mongo definition class.      */
/***/
PTABDEF __stdcall GetMONGO(PGLOBAL g, void *memp)
{
    return new(g, memp) MGODEF;
} // end of GetMONGO

#ifndef NOEXP
/***/
/* Functions to be defined if not exported by the CONNECT version. */
/***/
bool IsNum(PSZ s)
{
    for (char *p = s; *p; p++)
        if (*p == ']')
            break;
        else if (!isdigit(*p) || *p == '-')
            return false;

    return true;
} // end of IsNum
#endif

/***/
/* Return the columns definition to MariaDB.          */
/***/
PQRYRES __stdcall ColMONGO(PGLOBAL g, PTOS tp, char *tab,
                           char *db, bool info)
{
#ifdef NOMGOCOL
    // Cannot use discovery
    strcpy(g->Message, "No discovery, MGOColumns is not accessible");
    return NULL;
#else
    return MGOColumns(g, db, NULL, tp, info);
#endif
} // end of ColMONGO

```

The file

mongo.def  
:(required only on Windows)

```
LIBRARY      MONGO
EXPORTS
GetMONGO    @1
ColMONGO    @2
```

## Compiling this OEM

To compile this OEM module, first make the two or three required files by copy/pasting from the above listings.

Even if this module is to be used with a binary distribution, you need some source files in order to successfully compile it. At least the CONNECT header files that are included in

`jmgem.cpp`

and the ones they can include. This can be obtained by downloading the MariaDB source file tar.gz and extracting from it the CONNECT sources files in a directory that will be added to the additional source directories if it is not the directory containing the above files.

The module must be linked to the

`ha_connect.lib`

of the binary version it will be used with. Recent distributions add this lib in the plugin directory.

The resulting module, for instance

`mongo.so`

or

`mongo.dll`

, must be placed in the plugin directory of the MariaDB server. Then, you will be able to use MONGO like tables simply replacing in the CREATE TABLE statement the option

`TABLE_TYPE=MONGO`

with

`TABLE_TYPE=OEM SUBTYPE=MONGO MODULE='mongo.(so|dll)'`

. Actually, the module name, here supposedly 'mongo', can be anything you like.

This will work with the last (not yet) distributed versions of [MariaDB 10.0](#) and 10.1 because, even if it is not enabled, the MONGO type is included in them. This is also the case for [MariaDB 10.2.9](#) but then, on Windows, you will have to define NOEXP and NOMGOCOL because these functions are not exported by this version.

To implement for older versions that do not contain the MONGO type, you can add the corresponding source files, namely

`javaconn.cpp`

,

`jmgfam.cpp`

,

`jmgconn.cpp`

,

`mongo.cpp`

and

`tabjmg.cpp`

that you should find in the CONNECT extracted source files if you downloaded a recent version. As they include

`my_global.h`

, this is the reason why the included file was named this way. In addition, your compiling should define

`HAVE_JMGO`

and

`HAVE_JAVACONN`

. Of course, this is possible only if

`ha_connect.lib`

is available.

## 4.3.7.9 Using CONNECT



### Using CONNECT - General Information

[Using CONNECT - General Information](#)



### Using CONNECT - Virtual and Special Columns

[Virtual and special columns example usage](#)



### Using CONNECT - Importing File Data Into MariaDB Tables

[Directly using external \(file\) data has many advantages](#)



### Using CONNECT - Exporting Data From MariaDB

[Exporting data from MariaDB with CONNECT](#)



### Using CONNECT - Indexing

[Indexing with the CONNECT handler](#)



## Using CONNECT - Condition Pushdown

[Using CONNECT - Condition Pushdown.](#)



## USING CONNECT - Offline Documentation

[CONNECT Plugin User Manual.](#)



## Using CONNECT - Partitioning and Sharding

[Partitioning and Sharding with CONNECT](#)

# 4.3.7.9.1 Using CONNECT - General Information

## Contents

- 1. [Performance](#)
- 2. [Create Table statement](#)
- 3. [Drop Table statement](#)
- 4. [Alter Table statement](#)
- 5. [Update and Delete for File Tables](#)

The main characteristic of CONNECT is to enable accessing data scattered on a machine as if it was a centralized database. This, and the fact that locking is not used by connect (data files are open and closed for each query) makes CONNECT very useful for importing or exporting data into or from a MariaDB database and also for all types of Business Intelligence applications. However, it is not suited for transactional applications.

For instance, the index type used by CONNECT is closer to bitmap indexing than to B-trees. It is very fast for retrieving result but not when updating is done. In fact, even if only one indexed value is modified in a big table, the index is entirely remade (yet this being four to five times faster than for a b-tree index). But normally in Business Intelligence applications, files are not modified so often.

If you are using CONNECT to analyze files that can be modified by an external process, the indexes are of course not modified by it and become outdated. Use the OPTIMIZE TABLE command to update them before using the tables based on them.

This means also that CONNECT is not designed to be used by centralized servers, which are mostly used for transactions and often must run a long time without human intervening.

## Performance

Performances vary a great deal depending on the table type. For instance, ODBC tables are only retrieved as fast as the other DBMS can do. If you have a lot of queries to execute, the best way to optimize your work can be sometime to translate the data from one type to another. Fortunately this is very simple with CONNECT. Fixed formats like FIX, BIN or VEC tables can be created from slower ones by commands such as:

```
Create table fastable table_specs select * from slowtable;
```

FIX

and

BIN

are often the better choice because the I/O functions are done on blocks of

BLOCK\_SIZE

ROWS.

VEC

tables can be very efficient for tables having many columns only a few being used in each query. Furthermore, for tables of reasonable size, the

MAPPED

option can very often speed up many queries.

## Create Table statement

Be aware of the two broad kinds of CONNECT tables:

**Inward** They are table whose file name is not specified at create. An empty file will be given a default name ( `tablename.tabtype` ) and will be populated like for other engines. They do not require the FILE privilege and can be used for testing purpose.

They are all other

CONNECT

**Outward** tables and access external data sources or files. They are the true useful tables but require the FILE privilege.

## Drop Table statement

For outward tables, the `DROP TABLE` statement just removes the table definition but does not erase the table data. However, dropping an inward tables also erase the table data as well.

## Alter Table statement

Be careful using the `ALTER TABLE` statement. Currently the data compatibility is not tested and the modified definition can become incompatible with the data. In particular, Alter modifies the table definition only but does not modify the table data. Consequently, the table type should not be modified this way, except to correct an incorrect definition. Also adding, dropping or modifying columns may be wrong because the default offset values (when not explicitly given by the FLAG option) may be wrong when recompiled with missing columns.

Safe use of ALTER is for indexing, as we have seen earlier, and to change options such as MAPPED or HUGE those do not impact the data format but just the way the data file is accessed. Modifying the BLOCK\_SIZE option is all right with FIX, BIN, DBF, split VEC tables; however it is unsafe for VEC tables that are not split (only one data file) because at their creation the estimate size has been made a multiple of the block size. This can cause errors if this estimate is not a multiple of the new value of the block size.

In all cases, it is safer to drop and re-create the table (outward tables) or to make another one from the table that must be modified.

## Update and Delete for File Tables

CONNECT can execute these commands using two different algorithms:

- It can do it in place, directly modifying rows (update) or moving rows (delete) within the table file. This is a fast way to do it in particular when indexing is used.
- It can do it using a temporary file to make the changes. This is required when updating variable record length tables and is more secure in all cases.

The choice between these algorithms depends on the session variable `connect_use_tempfile`.

### 4.3.7.9.2 Using CONNECT - Virtual and Special Columns

CONNECT supports MariaDB [virtual and persistent columns](#). It is also possible to declare a column as being a CONNECT special column. Let us see on an example how this can be done. The boys table we have seen previously can be recreated as:

```
create table boys (
    linenum int(6) not null default 0 special=rowid,
    name char(12) not null,
    city char(12) not null,
    birth date not null date_format='DD/MM/YYYY',
    hired date not null date_format='DD/MM/YYYY' flag=36,
    agehired int(3) as (floor(datediff(hired,birth)/365.25))
    virtual,
    fn char(100) not null default '' special=FILEID)
engine=CONNECT table_type=FIX file_name='boys.txt' mapped=YES lrecl=47;
```

We have defined two CONNECT special columns. You can give them any name; it is the field SPECIAL option that specifies the special column functional name.

**Note:** the default values specified for the special columns do not mean anything. They are specified just to prevent getting warning messages when inserting new rows.

For the definition of the `agehired` virtual column, no CONNECT options can be specified as it has no offset or length, not being stored in the file.

The command:

```
select * from boys where city = 'boston';
```

will return:

linenum	name	city	birth	hired	agehired	fn
1	John	Boston	1986-01-25	2010-06-02	24	d:\mariadb\sql\data\boys.txt
2	Henry	Boston	1987-06-07	2008-04-01	20	d:\mariadb\sql\data\boys.txt
6	Bill	Boston	1986-09-11	2008-02-10	21	d:\mariadb\sql\data\boys.txt

Existing special columns are listed in the following table:

Special Name	Type	Description of the column value
ROWID	Integer	The row ordinal number in the table. This is not quite equivalent to a virtual column with an auto increment of 1 because rows are renumbered when deleting rows.
ROWNUM	Integer	The row ordinal number in the file. This is different from ROWID for multiple tables, TBL/XCOL/OCCUR/PIVOT tables, XML tables with a multiple column, and for DBF tables where ROWNUM includes soft deleted rows.
FILEID FDISK FPATH FNAME FTYPE	String	FILEID returns the full name of the file this row belongs to. Useful in particular for multiple tables represented by several files. The other special columns can be used to retrieve only one part of the full name.

TABID	String	The name of the table this row belongs to. Useful for TBL tables.
PARTID	String	The name of the partition this row belongs to. Specific to partitioned tables.
SERVID	String	The name of the federated server or server host used by a MYSQL table. "ODBC" for an ODBC table, "JDBC" for a JDBC table and "Current" for all other tables.

**Note:** CONNECT does not currently support auto incremented columns. However, a

ROWID

special column will do the job of a column auto incremented by 1.

### 4.3.7.9.3 Using CONNECT - Importing File Data Into MariaDB Tables

Directly using external (file) data has many advantages, such as to work on "fresh" data produced for instance by cash registers, telephone switches, or scientific apparatus. However, you may want in some case to import external data into your MariaDB database. This is extremely simple and flexible using the CONNECT handler. For instance, let us suppose you want to import the data of the xsample.xml XML file previously given in example into a MyISAM table called *biblio* belonging to the connect database. All you have to do is to create it by:

```
create table biblio engine=myisam select * from xsampall2;
```

This last statement creates the MyISAM table and inserts the original XML data, translated to tabular format by the xsampall2 CONNECT table, into the MariaDB *biblio* table. Note that further transformation on the data could have been achieved by using a more elaborate Select statement in the Create statement, for instance using filters, alias or applying functions to the data. However, because the Create Table process copies table data, later modifications of the xsample.xml file will not change the *biblio* table and changes to the *biblio* table will not modify the xsample.xml file.

All these can be combined or transformed by further SQL operations. This makes working with CONNECT much more flexible than just using the LOAD statement.

### 4.3.7.9.4 Using CONNECT - Exporting Data From MariaDB

Exporting data from MariaDB is obviously possible with CONNECT in particular for all formats not supported by the

SELECT INTO OUTFILE

statement. Let us consider the query:

```
select plugin_name handler, plugin_version version, plugin_author
author, plugin_description description, plugin_maturity maturity
from information_schema.plugins where plugin_type = 'STORAGE ENGINE';
```

Supposing you want to get the result of this query into a file handlers.htm in XML/HTML format, allowing displaying it on an Internet browser, this is how you can do it:

Just create the CONNECT table that will be used to make the file:

```
create table handout
engine=CONNECT table_type=XML file_name='handout.htm' header=yes
option_list='name=TABLE,coltype=HTML,attribute=border=1;cellpadding=5
,headattr=bgcolor=yellow'
select plugin_name handler, plugin_version version, plugin_author
author, plugin_description description, plugin_maturity maturity
from information_schema.plugins where plugin_type = 'STORAGE ENGINE';
```

Here the column definition is not given and will come from the Select statement following the Create. The CONNECT options are the same we have seen previously. This will do both actions, creating the matching *handlers* CONNECT table and 'filling' it with the query result.

**Note 1:** This could not be done in only one statement if the table type had required using explicit CONNECT column options. In this case, firstly create the table, and then populate it with an Insert statement.

**Note 2:** The source "plugins" table column "description" is a long text column, data type not supported for CONNECT tables. It has been silently internally replaced by varchar(256).

### 4.3.7.9.5 Using CONNECT - Indexing

## Contents

1. Standard Indexing
  1. Handling index errors
  2. Index file mapping
2. Block Indexing
  1. Difference between standard indexing and block indexing
  2. Notes for this Release:
3. Remote Indexing
4. Dynamic Indexing
5. Virtual Indexing

[Indexing](#) is one of the main ways to optimize queries. Key columns, in particular when they are used to join tables, should be indexed. But what should be done for columns that have only few distinct values? If they are randomly placed in the table they should not be indexed because reading many rows in random order can be slower than reading the entire table sequentially. However, if the values are sorted or clustered, indexing can be acceptable because [CONNECT](#) indexes store the values in the order they appear into the table and this will make retrieving them almost as fast as reading them sequentially.

[CONNECT](#) provides four indexing types:

1. Standard Indexing
2. Block Indexing
3. Remote Indexing
4. Dynamic Indexing

## Standard Indexing

[CONNECT](#) standard indexes are created and used as the ones of other storage engines although they have a specific internal format. The [CONNECT](#) handler supports the use of standard indexes for most of the file based table types.

You can define them in the [CREATE TABLE](#) statement, or either using the [CREATE INDEX](#) statement or the [ALTER TABLE](#) statement. In all cases, the index files are automatically made. They can be dropped either using the [DROP INDEX](#) statement or the [ALTER TABLE](#) statement, and this erases the index files.

Indexes are automatically reconstructed when the table is created, modified by [INSERT](#), [UPDATE](#) or [DELETE](#) commands, or when the [SEPINDEX](#) option is changed. If you have a lot of changes to do on a table at one moment, you can use table locking to prevent indexes to be reconstructed after each statement. The indexes will be reconstructed when unlocking the table. For instance:

```
lock table t1 write;
insert into t1 values(...);
insert into t1 values(...);
...
unlock tables;
```

If a table was modified by an external application that does not handle indexing, the indexes must be reconstructed to prevent returning false or incomplete results. To do this, use the [OPTIMIZE TABLE](#) command.

For outward tables, index files are not erased when dropping the table. This is the same as for the data file and preserves the possibility of several users using the same data file via different tables.

Unlike other storage engines, [CONNECT](#) constructs the indexes as files that are named by default from the data file name, not from the table name, and located in the data file directory. Depending on the [SEPINDEX](#) table option, indexes are saved in a unique file or in separate files (if [SEPINDEX](#) is true). For instance, if indexes are in separate files, the primary index of the table *dept.dat* of type DOS is a file named *dept\_PRIMARY.dnx*. This makes possible to define several tables on the same data file, with eventual different options such as mapped or not mapped, and to share the index files as well.

If the index file should have a different name, for instance because several tables are created on the same data file with different indexes, specify the base index file name with the [XFILE\\_NAME](#) option.

**Note1:** Indexed columns must be declared NOT NULL; [CONNECT](#) doesn't support indexes containing null values.

**Note 2:** MRR is used by standard indexing if it is enabled.

**Note 3:** Prefix indexing is not supported. If specified, the [CONNECT](#) engine ignores the prefix and builds a whole index.

## Handling index errors

The way [CONNECT](#) handles indexing is very specific. All table modifications are done regardless of indexing. Only after a table has been modified, or when an

[OPTIMIZE TABLE](#)

command is sent are the indexes made. If an error occurs, the corresponding index is not made. However, [CONNECT](#) being a non-transactional engine, it is unable to roll back the changes made to the table. The main causes of indexing errors are:

- Trying to index a nullable column. In this case, you can alter the table to declare the column as not nullable or, if the column is nullable indeed, make it not indexed.

- Entering duplicate values in a column indexed by a unique index. In this case, if the index was wrongly declared as unique, alter is declaration to reflect this. If the column should really contain unique values, you must manually remove or update the duplicate values.

In both cases, after correcting the error, remake the indexes with the [OPTIMIZE TABLE](#) command.

## Index file mapping

To accelerate the indexing process, CONNECT makes an index structure in memory from the index file. This can be done by reading the index file or using it as if it was in memory by “file mapping”. On enabled versions, file mapping is used according to the boolean [connect\\_idx\\_map](#) system variable. Set it to 0 (file read) or 1 (file mapping).

## Block Indexing

To accelerate input/output, CONNECT uses when possible a read/write mode by blocks of n rows, n being the value given in the BLOCK \_ SIZE option of the Create Table, or a default value depending on the table type. This is automatic for fixed files ([FIX](#) , [BIN](#) , [DBF](#) or [VEC](#) ), but must be specified for variable files ([DOS](#) , [CSV](#) or [FMT](#) ).

For blocked tables, further optimization can be achieved if the data values for some columns are “clustered” meaning that they are not evenly scattered in the table but grouped in some consecutive rows. Block indexing permits to skip blocks in which no rows fulfill a conditional predicate without having even to read the block. This is true in particular for sorted columns.

You indicate this when creating the table by using the [DISTRIB =d](#) column option. The enum value d can be *scattered* , *clustered* , or *sorted* . In general only one column can be sorted. Block indexing is used only for clustered and sorted columns.

## Difference between standard indexing and block indexing

- Block indexing is internally handled by CONNECT while reading sequentially a table data. This means in particular that when standard indexing is used on a table, block indexing is not used.
- In a query, only one standard index can be used. However, block indexing can combine the restrictions coming from a where clause implying several clustered/sorted columns.
- The block index files are faster to make and much smaller than standard index files.

## Notes for this Release:

- On all operations that create or modify a table, CONNECT automatically calculates or recalculates and saves the mini/maxi or bitmap values for each block, enabling it to skip block containing no acceptable values. In the case where the optimize file does not correspond anymore to the table, because it has been accidentally destroyed, or because some column definitions have been altered, you can use the [OPTIMIZE TABLE](#) command to reconstruct the optimization file.
- Sorted column special processing is currently restricted to ascending sort. Column sorted in descending order must be flagged as clustered. Improper sorting is not checked in Update or Insert operations but is flagged when optimizing the table.
- Block indexing can be done in two ways. Keeping the min/max values existing for each block, or keeping a bitmap allowing knowing what column distinct values are met in each blocks. This second ways often gives a better optimization, except for sorted columns for which both are equivalent. The bitmap approach can be done only on columns having not too many distinct values. This is estimated by the MAX \_ DIST option value associated to the column when creating the table. Bitmap block indexing will be used if this number is not greater than the MAXBMP setting for the database.
- CONNECT cannot perform block indexing on case insensitive character columns. To force block indexing on a character column, specify its charset as not case insensitive, for instance as binary. However this will also apply to all other clauses, this column being now case sensitive.

## Remote Indexing

Remote indexing is specific to the [MYSQL](#) table type. It is equivalent to what the [FEDERATED](#) storage does. A MYSQL table does not support indexes per se. Because access to the table is handled remotely, it is the remote table that supports the indexes. What the MYSQL table does is just to add a WHERE clause to the [SELECT](#) command sent to the remote server allowing the remote server to use indexing when applicable. Note however that because CONNECT adds when possible all or part of the where clause of the original query, this happens often even if the remote indexed column is not declared locally indexed. The only, but very important, case a column should be locally declared indexed is when it is used to join tables. Otherwise, the required where clause would not be added to the sent SELECT query.

See [Indexing of MYSQL tables](#) for more.

## Dynamic Indexing

An indexed created as “dynamic” is a standard index which, in some cases, can be reconstructed for a specific query. This happens in particular for some queries where two tables are joined by an indexed key column. If the “from” table is big and the “to” big table reduced in size because of a where clause, it can be worthwhile to reconstruct the index on this reduced table.

Because of the time added by reconstructing the index, this will be valuable only if the time gained by reducing the index size if more than this reconstruction time. This is why this should not be done if the “from” table is small because there will not be enough row joining to compensate for the additional time. Otherwise, the gain of using a dynamic index is:

- Indexing time is a little faster if the index is smaller.
- The join process will return only the rows fulfilling the where clause.
- Because the table is read sequentially when reconstructing the index there no need for MRR.

- Constructing the index can be faster if the table is reduced by block indexing.
- While constructing the index, CONNECT also stores in memory the values of other used columns.

This last point is particularly important. It means that after the index is reconstructed, the join is done on a temporary memory table.

Unfortunately, storage engines being called independently by MariaDB for each table, CONNECT has no global information to decide when it is good to use dynamic indexing. This is why you should use it only on cases where you see that some important join queries take a very long time and only on columns used for joining the table. How to declare an index to be dynamic is by using the Boolean DYNAM index option. For instance, the query:

```
select d.diag, count(*) cnt from diag d, patients p where d.pnb =
p.pnb and ageyears < 17 and county = 30 and drg >> 11 and d.diag
between 4296 and 9434 group by d.diag order by cnt desc;
```

Such a query joining the diag table to the patients table may last a very long time if the tables are big. To declare the primary key on the pnb column of the patients table to be dynamic:

```
alter table patients drop primary key;
alter table patients add primary key (pnb) comment 'DYNAMIC' dynam=1;
```

Note 1: The comment is not mandatory here but useful to see that the index is dynamic if you use the [SHOW INDEX](#) command.

Note 2: There is currently no way to just change the DYNAM option without dropping and adding the index. This is unfortunate because it takes time.

## Virtual Indexing

It applies only to the virtual tables of type [VIR](#) and must be made on a column specifying

SPECIAL=ROWID

or

SPECIAL=ROWNUM

## 4.3.7.9.6 Using CONNECT - Condition Pushdown

The [ODBC](#) , [JDBC](#) , [MYSQL](#) , [TBL](#) and WMI table types use engine condition pushdown in order to restrict the number of rows returned by the RDBS source or the WMI component.

The CONDITION\_PUSHDOWN argument used in old versions of CONNECT is no longer needed because CONNECT uses condition pushdown unconditionally.

## 4.3.7.9.7 USING CONNECT - Offline Documentation

Note: You can download a [PDF version of the CONNECT documentation \(1.7.0003\)](#).

## 4.3.7.9.8 Using CONNECT - Partitioning and Sharding

### Contents

1. [Partition engine issues](#)
2. [File Partitioning](#)
  1. [Outward Tables](#)
    1. [Partitioning on a Special Column](#)
    2. [Partitioning of zipped tables](#)
  3. [Table Partitioning](#)
    1. [Indexing with Table Partitioning](#)
    2. [Sharding with Table Partitioning](#)
      1. [Sharding on a Special Column](#)
  4. [Current Partition Limitations](#)
    1. [Update statement](#)
    2. [Alter Table statement](#)
    3. [Rowid special column](#)

CONNECT supports the MySQL/MariaDB partition specification. It is done similar to the way [MyISAM](#) or [InnoDB](#) do by using the PARTITION engine that must be enabled for this to work. This type of partitioning is sometimes referred as "horizontal partitioning".

Partitioning enables you to distribute portions of individual tables across a file system according to rules which you can set largely as needed. In effect, different portions of a table are stored as separate tables in different locations. The user-selected rule by which the division of data is accomplished is known as a partitioning function, which in MariaDB can be the modulus, simple matching against a set of ranges or value lists, an internal hashing function, or a linear hashing function.

CONNECT takes this notion a step further, by providing two types of partitioning:

1. File partitioning. Each partition is stored in a separate file like in multiple tables.

2. Table partitioning. Each partition is stored in a separate table like in TBL tables.

## Partition engine issues

Using partitions sometimes requires creating the tables in an unnatural way to avoid some error due to several partition engine bugs:

1. Engine specific column and index options are not recognized and cause a syntax error when the table is created. The workaround is to create the table in two steps, a CREATE TABLE statement followed by an ALTER TABLE statement.
2. The connection string, when specified for the table, is lost by the partition engine. The workaround is to specify the connection string in the `option_list`.
3. [MySQL upstream bug #71095](#). In case of list columns partitioning it sometimes causes a false “impossible where” clause to be raised. This makes a wrong void result returned when it should not be void. There is no workaround but this bug should be hopefully fixed.

The following examples are using the above workaround syntax to address these issues.

## File Partitioning

File partitioning applies to file-based CONNECT table types. As with multiple tables, physical data is stored in several files instead of just one. The differences to multiple tables are:

1. Data is distributed amongst the different files following the partition rule.
2. Unlike multiple tables, partitioned tables are not read only.
3. Unlike multiple tables, partitioned tables can be indexable.
4. The file names are generated from the partition names.
5. Query pruning is automatically made by the partition engine.

The table file names are generated differently depending on whether the table is an inward or outward table. For inward tables, for which the file name is not specified, the partition file names are:

```
Data file name: table_name#P#partition_name.table_file_type  
Index file name: table_name#P#partition_name.index_file_type
```

For instance for the table:

```
CREATE TABLE t1 (  
    id INT KEY NOT NULL,  
    msg VARCHAR(32))  
ENGINE=CONNECT TABLE_TYPE=FIX  
partition by range(id) (  
    partition first values less than(10),  
    partition middle values less than(50),  
    partition last values less than(MAXVALUE));
```

CONNECT will generate in the current data directory the files:

```
| t1#P#first.fix  
| t1#P#first.fnx  
| t1#P#middle.fix  
| t1#P#middle.fnx  
| t1#P#last.fix  
| t1#P#last.fnx
```

This is similar to what the partition engine does for other engines - CONNECT partitioned inward tables behave like other engines partition tables do. Just the data format is different.

Note: If sub-partitioning is used, inward table files and index files are named:

```
| table_name#P#partition_name#SP#subpartition_name.type  
| table_name#P#partition_name#SP#subpartition_name.index_type
```

## Outward Tables

The real problems occur with outward tables, in particular when they are created from already existing files. The first issue is to make the partition table use the correct existing file names. The second one, only for already existing not void tables, is to be sure the partitioning function match the distribution of the data already existing in the files.

The first issue is addressed by the way data file names are constructed. For instance let us suppose we want to make a table from the fixed formatted files:

```
E:\Data\part1.txt  
E:\Data\part2.txt  
E:\Data\part3.txt
```

This can be done by creating a table such as:

```
create table t2 (
id int not null,
msg varchar(32),
index XID(id))
engine=connect table_type=FIX file_name='E:/Data/part%s.txt'
partition by range(id) (
partition `1` values less than(10),
partition `2` values less than(50),
partition `3` values less than(MAXVALUE));
```

The rule is that for each partition the matching file name is internally generated by replacing in the given FILE\_NAME option value the "%s" part by the partition name.

If the table was initially void, further inserts will populate it according to the partition function. However, if the files did exist and contained data, this is your responsibility to determine what partition function actually matches the data distribution in them. This means in particular that partitioning by key or by hash cannot be used (except in exceptional cases) because you have almost no control over what the used algorithm does.

In the example above, there is no problem if the table is initially void, but if it is not, serious problems can be met if the initial distribution does not match the table distribution. Supposing a row in which "id" as the value 12 was initially contained in the part1.txt file, it will be seen when selecting the whole table but if you ask:

```
select * from t2 where id = 12;
```

The result will have 0 rows. This is because according to the partition function query pruning will only look inside the second partition and will miss the row that is in the wrong partition.

One way to check for wrong distribution if for instance to compare the results from queries such as:

```
SELECT partition_name, table_rows FROM
information_schema.partitions WHERE table_name = 't2';
```

And

```
SELECT CASE WHEN id < 10 THEN 1 WHEN id < 50 THEN 2 ELSE 3 END
AS pn, COUNT(*) FROM part3 GROUP BY pn;
```

If they match, the distribution can be correct although this does not prove it. However, if they do not match, the distribution is surely wrong.

## Partitioning on a Special Column

There are some cases where the files of a multiple table do not contain columns that can be used for range or list partitioning. For instance, let's suppose we have a multiple table based on the following files:

```
tmp/boston.txt
tmp/chicago.txt
tmp/atlanta.txt
```

Each of them containing the same kind of data:

```
ID: int
First_name: varchar(16)
Last_name: varchar(30)
Birth: date
Hired: date
Job: char(10)
Salary: double(8,2)
```

A multiple table can be created on them, for instance by:

```
create table mulemp (
id int NOT NULL,
first_name varchar(16) NOT NULL,
last_name varchar(30) NOT NULL,
birth date NOT NULL date_format='DD/MM/YYYY',
hired date NOT NULL date_format='DD/MM/YYYY',
job char(10) NOT NULL,
salary double(8,2) NOT NULL
) engine=CONNECT table_type=FIX file_name='tmp/*.txt' multiple=1;
```

The issue is that if we want to create a partitioned table on these files, there are no columns to use for defining a partition function. Each city file can have

the same kind of column values and there is no way to distinguish them.

However, there is a solution. It is to add to the table a special column that will be used by the partition function. For instance, the new table creation can be done by:

```
create table partemp (
id int NOT NULL,
first_name varchar(16) NOT NULL,
last_name varchar(30) NOT NULL,
birth date NOT NULL date_format='DD/MM/YYYY',
hired date NOT NULL date_format='DD/MM/YYYY',
job char(16) NOT NULL,
salary double(10,2) NOT NULL,
city char(12) default 'boston' special=PARTID,
index XID(id)
) engine=CONNECT table_type=FIX file_name='E:/Data/Test/%s.txt';
alter table partemp
partition by list columns(city) (
partition `atlanta` values in('atlanta'),
partition `boston` values in('boston'),
partition `chicago` values in('chicago'));
```

Note 1: we had to do it in two steps because of the column CONNECT options.

Note 2: the special column PARTID returns the name of the partition in which the row is located.

Note 3: here we could have used the FNAME special column instead because the file name is specified as being the partition name.

This may seem rather stupid because it means for instance that a row will be in partition boston if it belongs to the partition boston! However, it works because the partition engine doesn't know about special columns and behaves as if the city column was a real column.

What happens if we populate it by?

```
insert into partemp(id,first_name,last_name,birth,hired,job,salary) values
(1205,'Harry','Cover','1982-10-07','2010-09-21','MANAGEMENT',125000.00);
insert into partemp values
(1524,'Jim','Beams','1985-06-18','2012-07-25','SALES',52000.00,'chicago'),
(1431,'Johnny','Walker','1988-03-12','2012-08-09','RESEARCH',46521.87,'boston'),
(1864,'Jack','Daniels','1991-12-01','2013-02-16','DEVELOPMENT',63540.50,'atlanta');
```

The value given for the city column (explicitly or by default) will be used by the partition engine to decide in which partition to insert the rows. It will be ignored by CONNECT (a special column cannot be given a value) but later will return the matching value. For instance:

```
select city, first_name, job from partemp where id in (1524,1431);
```

This query returns:

city	first_name	job
boston	Johnny	RESEARCH
chicago	Jim	SALES

Everything works as if the city column was a real column contained in the table data files.

### Partitioning of zipped tables

Two cases are currently supported:

If a table is based on several zipped files, portioning is done the standard way as above. This is the *file\_name* option specifying the name of the zip files that shall contain the '%s' part used to generate the file names.

If a table is based on only one zip file containing several entries, this will be indicated by placing the '%s' part in the entry option value.

Note: If a table is based on several zipped files each containing several entries, only the first case is possible. Using sub-partitioning to make partitions on each entries is not supported yet.

## Table Partitioning

With table partitioning, each partition is physically represented by a sub-table. Compared to standard partitioning, this brings the following features:

1. The partitions can be tables driven by different engines. This relieves the current existing limitation of the partition engine.
2. The partitions can be tables driven by engines not currently supporting partitioning.
3. Partition tables can be located on remote servers, enabling table sharding.
4. Like for TBL tables, the columns of the partition table do not necessarily match the columns of the sub-tables.

The way it is done is to create the partition table with a table type referring to other tables, [PROXY](#), [MYSQL ODBC](#) or [JDBC](#). Let us see how this is done on a simple example. Supposing we have created the following tables:

```

create table xt1 (
id int not null,
msg varchar(32))
engine=myisam;

create table xt2 (
id int not null,
msg varchar(32)); /* engine=innnoDB */

create table xt3 (
id int not null,
msg varchar(32))
engine=connect table_type=CSV;

```

We can for instance create a partition table using these tables as physical partitions by:

```

create table t3 (
id int not null,
msg varchar(32))
engine=connect table_type=PROXY tablename='xt%';
partition by range columns(id) (
partition `1` values less than(10),
partition `2` values less than(50),
partition `3` values less than(MAXVALUE));

```

Here the name of each partition sub-table will be made by replacing the '%s' part of the tablename option value by the partition name. Now if we do:

```

insert into t3 values
(4, 'four'),(7, 'seven'),(10, 'ten'),(40, 'forty'),
(60, 'sixty'),(81, 'eighty one'),(72, 'seventy two'),
(11, 'eleven'),(1, 'one'),(35, 'thirty five'),(8, 'eight');

```

The rows will be distributed in the different sub-tables according to the partition function. This can be seen by executing the query:

```

select partition_name, table_rows from
information_schema.partitions where table_name = 't3';

```

This query replies:

partition_name	table_rows
1	4
2	4
3	3

Query pruning is of course automatic, for instance:

```

explain partitions select * from t3 where id = 81;

```

This query replies:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	part5	3	ALL	<null>	<null>	<null>	<null>	22	Using where

When executing this select query, only sub-table xt3 will be used.

## Indexing with Table Partitioning

Using the **PROXY** table type seems natural. However, in this current version, the issue is that PROXY (and **ODBC**) tables are not indexable. This is why, if you want the table to be indexed, you must use the **MYSQL** table type. The CREATE TABLE statement will be almost the same:

```

create table t4 (
id int key not null,
msg varchar(32))
engine=connect table_type=MYSQL tablename='xt%';
partition by range columns(id) (
partition `1` values less than(10),
partition `2` values less than(50),
partition `3` values less than(MAXVALUE));

```

The column `id` is declared as a key, and the table type is now MYSQL. This makes Sub-tables accessed by calling a MariaDB server as MYSQL tables do. Note that this modifies only the way CONNECT sub-tables are accessed.

However, indexing just make the partitioned table use “remote indexing” the way FEDERATED tables do. This means that when sending the query to retrieve the table data, a where clause will be added to the query. For instance, let's suppose you ask:

```
select * from t4 where id = 7;
```

The query sent to the server will be:

```
SELECT `id`, `msg` FROM `xt1` WHERE `id` = 7
```

On a query like this one, it does not change much because the where clause could have been added anyway by the cond\_push function, but it does make a difference in case of joins. The main thing to understand is that real indexing is done by the called table and therefore that it should be indexed.

This also means that the `xt1`, `xt2`, and `xt3` table indexes should be made separately because creating the `t2` table as indexed does not make the indexes on the sub-tables.

## Sharding with Table Partitioning

Using table partitioning can have one more advantage. Because the sub-tables can address a table located on another server, it is possible to shard a table on separate servers and hardware machines. This may be required to access as one table data already located on several remote machines, such as servers of a company branches. Or it can be just used to split a huge table for performance reason. For instance, supposing we have created the following tables:

```
create table rt1 (id int key not null, msg varchar(32))
engine=federated connection='mysql://root@host1/test/sales';

create table rt2 (id int key not null, msg varchar(32))
engine=federated connection='mysql://root@host2/test/sales';

create table rt3 (id int key not null, msg varchar(32))
engine=federated connection='mysql://root@host3/test/sales';
```

Creating the partition table accessing all these will be almost like what we did with the `t4` table:

```
create table t5 (
id int key not null,
msg varchar(32))
engine=connect table_type=MYSQL tablename='rt%'
partition by range columns(id) (
partition `1` values less than(10),
partition `2` values less than(50),
partition `3` values less than(MAXVALUE));
```

The only difference is the `tablename` option now referring to the `rt1`, `rt2`, and `rt3` tables. However, even if it works, this is not the best way to do it. This is because accessing a table via the MySQL API is done twice per table. Once by CONNECT to access the FEDERATED table on the local server, then a second time by FEDERATED engine to access the remote table.

The CONNECT MYSQL table type being used anyway, you'd rather use it to directly access the remote tables. Indeed, the partition names can also be used to modify the connection URL's. For instance, in the case shown above, the partition table can be created as:

```
create table t6 (
id int key not null,
msg varchar(32))
engine=connect table_type=MYSQL
option_list='connect=mysql://root@host%_test/sales'
partition by range columns(id) (
partition `1` values less than(10),
partition `2` values less than(50),
partition `3` values less than(MAXVALUE));
```

Several things can be noted here:

1. As we have seen before, the partition engine currently loses the connection string. This is why it was specified as “connect” in the option list.
2. For each partition sub-tables, the “%s” part of the connection string has been replaced by the partition name.
3. It is not needed anymore to define the `rt1`, `rt2`, and `rt3` tables (even it does not harm) and the FEDERATED engine is no more used to access the remote tables.

This is a simple case where the connection string is almost the same for all the sub-tables. But what if the sub-tables are accessed by very different connection strings? For instance:

```

For rt1: connection='mysql://root:tinono@127.0.0.1:3307/test/xt1'
For rt2: connection='mysql://foo:foopass@denver/dbemp/xt2'
For rt3: connection='mysql://root@houston :5505/test/tabc'

```

There are two solutions. The first one is to use the parts of the connection string to differentiate as partition names:

```

create table t7 (
id int key not null,
msg varchar(32))
engine=connect table_type=MYSQL
option_list='connect=mysql://%s'
partition by range columns(id) (
partition `root:tinono@127.0.0.1:3307/test/xt1` values less than(10),
partition `foo:foopass@denver/dbemp/xt2` values less than(50),
partition `root@houston :5505/test/tabc` values less than(MAXVALUE));

```

The second one, allowing avoiding too complicated partition names, is to create federated servers to access the remote tables (if they do not already exist, else just use them). For instance the first one could be:

```

create server `server_one` foreign data wrapper 'mysql'
options
(host '127.0.0.1',
database 'test',
user 'root',
password 'tinono',
port 3307);

```

Similarly, "server\_two" and "server\_three" would be created and the final partition table would be created as:

```

create table t8 (
id int key not null,
msg varchar(32))
engine=connect table_type=MYSQL
option_list='connect=server_%s'
partition by range columns(id) (
partition `one/xt1` values less than(10),
partition `two/xt2` values less than(50),
partition `three/tabc` values less than(MAXVALUE));

```

It would be even simpler if all remote tables had the same name on the remote databases, for instance if they all were named xt1, the connection string could be set as "server\_%s/xt1" and the partition names would be just "one", "two", and "three".

## Sharding on a Special Column

The technique we have seen above with file partitioning is also available with table partitioning. Companies willing to use as one table data sharded on the company branch servers can, as we have seen, add to the table create definition a special column. For instance:

```

create table t9 (
id int not null,
msg varchar(32),
branch char(16) default 'main' special=PARTID,
index XID(id))
engine=connect table_type=MYSQL
option_list='connect=server_%s/sales'
partition by range columns(id) (
partition `main` values in('main'),
partition `east` values in('east'),
partition `west` values in('west'));

```

This example assumes that federated servers had been created named "server\_main", "server\_east" and "server\_west" and that all remote tables are named "sales". Note also that in this example, the column id is no more a key.

## Current Partition Limitations

Because the partition engine was written before some other engines were added to MariaDB, the way it works is sometime incompatible with these engines, in particular with CONNECT.

## Update statement

With the sample tables above, you can do update statements such as:

```
update t2 set msg = 'quatre' where id = 4;
```

It works perfectly and is accepted by CONNECT. However, let us consider the statement:

```
update t2 set id = 41 where msg = 'four';
```

This statement is not accepted by CONNECT. The reason is that the column id being part of the partition function, changing its value may require the modified row to be moved to another partition. The way it is done by the partition engine is to delete the old row and to re-insert the new modified one. However, this is done in a way that is not currently compatible with CONNECT (remember that CONNECT supports UPDATE in a specific way, in particular for the table type MYSQL) This limitation could be temporary. Meanwhile the workaround is to manually do what is done above,

Deleting the row to modify and inserting the modified row:

```
delete from t2 where id = 4;
insert into t2 values(41, 'four');
```

## Alter Table statement

For all CONNECT outward tables, the ALTER TABLE statement does not make any change in the table data. This is why ALTER TABLE should not be used; in particular to modify the partition definition, except of course to correct a wrong definition. Note that using ALTER TABLE to create a partition table in two steps because column options would be lost is valid as it applies to a table that is not yet partitioned.

As we have seen, it is also safe to use it to create or drop indexes. Otherwise, a simple rule of thumb is to avoid altering a table definition and better drop and re-create a table whose definition must be modified. Just remember that for outward CONNECT tables, dropping a table does not erase the data and that creating it does not modify existing data.

## Rowid special column

Each partition being handled separately as one table, the ROWID special column returns the rank of the row in its partition, not in the whole table. This means that for partition tables ROWID and ROWNUM are equivalent.

## 4.3.7.10 CONNECT - Making the GetRest Library

To enable the REST feature with binary distributions of MariaDB, the function calling the cpprestsdk package is not included in CONNECT, thus allowing CONNECT normal operation when the cpprestsdk package is not installed. Therefore, it must be compiled separately as a library (so or dll) that will be loaded by CONNECT when needed.

This library will contain only one file shown here:

```
***** Restget C++ Program Source Code File (.CPP) *****/
/* Adapted from the sample program of the Casablanca tutorial. */
/* Copyright Olivier Bertrand 2019. */
/*****
#include <cpprest/filestream.h>
#include <cpprest/http_client.h>

using namespace utility::conversions; // String conversions utilities
using namespace web; // Common features like URIs.
using namespace web::http; // Common HTTP functionality
using namespace web::http::client; // HTTP client features
using namespace concurrency::streams; // Asynchronous streams

typedef const char* PCSZ;

extern "C" int restGetFile(char* m, bool xt, PCSZ http, PCSZ uri, PCSZ fn);

/*****
/* Make a local copy of the requested file. */
/*****
int restGetFile(char *m, bool xt, PCSZ http, PCSZ uri, PCSZ fn)
{
    int rc = 0;
    auto fileStream = std::make_shared<ostream>();

    if (!http || !fn) {
        strcpy(m, "Missing http or filename");
        return 2;
    } // endif

    if (xt)
        fprintf(stderr, "restGetFile: fn=%s\n", fn);

    // Open stream to output file.
    only::task<void> requestTask = fileStream->open_outstream(to_string(t(fn))
```

```

    .then([=](ostream outFile) {
        *fileStream= outFile;

        if (xt)
            fprintf(stderr, "Outfile isopen=%d\n", outFile.is_open());

        // Create http_client to send the request.
        http_client client(to_string_t(http));

        if (uri) {
            // Build request URI and start the request.
            uri_builder builder(to_string_t(uri));
            return client.request(methods::GET, builder.to_string());
        } else
            return client.request(methods::GET);
    })

    // Handle response headers arriving.
    .then([=](http_response response) {
        if (xt)
            fprintf(stderr, "Received response status code:%u\n",
                    response.status_code());

        // Write response body into the file.
        return response.body().read_to_end(fileStream->streambuf());
    })

    // Close the file stream.
    .then([=](size_t n) {
        if (xt)
            fprintf(stderr, "Return size=%zu\n", n);

        return fileStream->close();
    });

    // Wait for all the outstanding I/O to complete and handle any exceptions
    try {
        if (xt)
            fprintf(stderr, "Waiting\n");

        requestTask.wait();
    } catch (const std::exception &e) {
        if (xt)
            fprintf(stderr, "Error exception: %s\n", e.what());

        sprintf(m, "Error exception: %s", e.what());
        rc= 1;
    } // end try/catch

    if (xt)
        fprintf(stderr, "restget done: rc=%d\n", rc);

    return rc;
} // end of restGetFile

```

This file exists in the source of CONNECT as

`restget.cpp`

. If you have no access to the source, use your favorite editor to make it by copy/pasting from the above.

Then, on Linux, compile the GetRest.so library:

```
g++ -o GetRest.so -O3 -Wall -std=c++11 -fPIC -shared restget.cpp -lcpprest
```

Note: You can replace

-O3

by

-g

to make a debug version.

This library should be placed where it can be accessed. A good place is the directory where the

`libcpprest.so`

is, for instance

`/usr/lib64`

. You can move or copy it there.

On windows, using Visual Studio, make an empty win32 dll project named GetRest and add it the above file. Also add it the module definition file

```
restget.def
```

```
:
```

```
LIBRARY REST
EXPORTS
    restGetFile    @1
```

Important: This file must be specified in the property linker input page.

Once compiled, the release or debug versions can be copied in the

```
cpprestsdk
```

```
corresponding directories, bin or debug\bin.
```

That is all. It is a once-off job. Once done, it will work with all new MariaDB versions featuring CONNECT version 1.07.

Note: the xt tracing variable is true when connect\_xtrace setting includes the value "MONGO" (512).

Caution: If your server crashes when using this feature, this is likely because the GetRest lib is linked to the wrong cpprestsdk lib (this may only apply on Windows)

A Release version of GetRest must be linked to the release version of the cpprestsdk lib (cpprest\_2\_10.dll) but if you make a Debug version of GetRest, make sure it is linked to the Debug version of cpprestsdk lib (cpprest\_2\_10d.dll)

This may be automatic if you use Visual Studio to make the GetRest.dll.

## 4.3.7.11 CONNECT - Adding the REST Feature as a Library Called by an OEM Table

If you are using a version of MariaDB that does not support REST, this is how the REST feature can be added as a library called by an OEM table.

Before making the REST OEM module, the Microsoft Casablanca package must be installed as for compiling MariaDB from source.

Even if this module is to be used with a binary distribution, you need some CONNECT source files in order to successfully make it. It is made with four files existing in the version 1.06.0010 of CONNECT: tabrest.cpp, restget.cpp, tabrest.h and mini-global.h. It also needs the CONNECT header files that are included in tabrest.cpp and the ones they can include. This can be obtained by going to a recent download site of a version of MariaDB that includes the REST feature, downloading the MariaDB source file tar.gz and extracting from it the CONNECT sources files in a directory that will be added to the additional source directories if it is not the directory containing the above files.

On Windows, use a recent version of Visual Studio. Make a new empty DLL project and add the source files tabrest.cpp and restget.cpp. Visual studio should automatically generate all necessary connections to the cpprest SDK. Just edit the properties of the project to add the additional include directory (the one where the CONNECT source was downloaded) et the link to the ha\_connect.lib of the binary version of MariaDB (in the same directory than ha\_connect.dll in your binary distribution). Add the preprocessor definition XML\_SUPPORT. Also set in the linker input page of the project property the Module definition File to the rest.def file (with its full path) also existing in the CONNECT source files. If you are making a debug configuration, make sure that in the C/C++ Code generation page the Runtime library line specifies Multi-threaded Debug DLL (/MDd) or your server will crash when using the feature.

This is not really simple but it is nothing compared with Linux! Someone having made an OEM module for its own application have written:

For whatever reason, g++ / ld on Linux are both extremely picky about what they will and won't consider a \*\*library\*\* for linking purposes. In order to get them to recognize and therefore find `ha\_connect.so` as a "valid" linkable library, `ha\_connect.so` must exist in a directory whose path is in `/etc/ld.so.conf` or `/etc/ld.so.conf.d/ha\_connect.conf` \*AND\* its filename must begin with "lib".

On Fedora, you can make a link to ha\_connect.so by:

```
$ sudo ln -s ..path to../ha_connect.so /usr/lib64/libconnect.so
```

This provides a library whose name begins with "lib". It was made in /usr/lib64/ because it was the directory of the libcpprest.so Casablanca library. This solved the need of a file in /etc/ld.so.conf.d as this was already done for the cpprest library. Note that the -s parameter is a must, without it all sort of nasty errors are met when using the feature.

Then compile and install the OEM module with:

```
$ mkdir oem
$ cd oem
$ makedir Release
$ make -f oemrest.mak
$ sudo cp rest.so /usr/local/mysql/lib/plugin
```

The oemrest.mak file:

```

#LINUX
CPP = g++
LD = g++
OD = ./Release/
SD = /home/olivier/MariaDB/server/storage/connect/
CD =/usr/lib64
# flags to compile object files that can be used in a dynamic library
CFLAGS= -Wall -c -O3 -std=c++11 -fPIC -fno-rtti -I$(SD) -DXML_SUPPORT
# Replace -O3 by -g for debug
LDFLAGS = -L$(CD) -lcpprest -lconnect

# Flags to create a dynamic library.
DYNLINKFLAGS = -shared
# on some platforms, use '-G' instead.

# REST library's archive file
OEMREST = rest.so

SRCS_CPP = $(SD)tabrest.cpp $(SD)restget.cpp
OBJS_CPP = $(OD)tabrest.o $(OD)restget.o

# top-level rule
all: $(OEMREST)

$(OEMREST): $(OBJS_CPP)
    $(LD) $(OBJS_CPP) $(LDFLAGS) $(DYNLINKFLAGS) -o $@

#CPP Source files
$(OD)tabrest.o: $(SD)tabrest.cpp $(SD)mini-global.h $(SD)global.h $(SD)plgdbsem.h $(SD)xtable.h $(SD)filamtxt.h $(SD)plgxml.h
    $(CPP) $(CFLAGS) -o $@ $(SD)$(*F).cpp
$(OD)restget.o: $(SD)restget.cpp $(SD)mini-global.h $(SD)global.h
    $(CPP) $(CFLAGS) -o $@ $(SD)$(*F).cpp

# clean everything
clean:
    $(RM) $(OBJS_CPP) $(OEMREST)

```

The SD and CD variables are the directories of the CONNECT source files and the one containing the libcpprest.so lib. They can be edited to match those on your machine OD is the directory that was made to contain the object files.

A very important flag is -fno-rtti. Without it you would be in big trouble.

The resulting module, for instance rest.so or rest.dll, must be placed in the plugin directory of the MariaDB server. Then, you will be able to use NoSQL tables simply replacing in the CREATE TABLE statement the TABLE\_TYPE option =JSON or XML by TABLE\_TYPE=OEM SUBTYPE=REST MODULE='rest.(so|dll)'. Actually, the module name, here supposedly 'rest', can be anything you like.

The file type is JSON by default. If not, it must be specified like this:

```
OPTION_LIST='Ftype=XML'
```

To be added to the create table statement. For instance:

```

CREATE TABLE webw
ENGINE=CONNECT TABLE_TYPE=OEM MODULE='Rest.dll' SUBTYPE=REST
FILE_NAME='weatherdata.xml'
HTTP='https://samples.openweathermap.org/data/2.5/forecast?q=London,us&mode=xml&appid=b6907d289e10d714a6e88b30761fae22'
OPTION_LIST='Ftype=XML,Depth=3,Rownode=weatherdata';

```

Note: this last example returns an XML file whose format was not recognized by old CONNECT versions. It is here the reason of the option 'Rownode=weatherdata'.

If you have trouble making the module, you can post an issue on [JIRA](#).

## 4.3.7.12 CONNECT - Compiling JSON UDFs in a Separate Library

Although the JSON UDFs can be nicely included in the CONNECT library module, there are cases when you may need to have them in a separate library.

This is when CONNECT is compiled embedded, or if you want to test or use these UDFs with other MariaDB versions not including them.

To make it, you need to have access to the most recent MariaDB source code. Then, make a project containing these files:

1. jsonudf.cpp
2. json.cpp
3. value.cpp

4. osutil.c
5. plugutil.cpp
6. maputil.cpp
7. jsonutil.cpp

jsonutil.cpp

is not distributed with the source code, you will have to make it from the following:

```

#include "my_global.h"
#include "mysqld.h"
#include "plugin.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>

#include "global.h"

extern "C" int GetTraceValue(void) { return 0; }
uint GetJsonGrpSize(void) { return 100; }

/***/
/* These replace missing function of the (not used) DTVAL class. */
/***/
typedef struct _datpar *PDTp;
PDTp MakeDateFormat(PGLOBAL, PSZ, bool, bool, int) { return NULL; }
int ExtractDate(char*, PDTp, int, int val[6]) { return 0; }

#ifndef __WIN__
my_bool CloseFileHandle(HANDLE h)
{
    return !CloseHandle(h);
} /* end of CloseFileHandle */

#else /* UNIX */
my_bool CloseFileHandle(HANDLE h)
{
    return (close(h)) ? TRUE : FALSE;
} /* end of CloseFileHandle */

int GetLastError()
{
    return errno;
} /* end of GetLastError */

#endif // UNIX

/***/
/* Program for sub-allocating one item in a storage area. */
/* Note: This function is equivalent to PlugSubAlloc except that in */
/* case of insufficient memory, it returns NULL instead of doing a */
/* long jump. The caller must test the return value for error. */
/***/
void *PlgDBSubAlloc(PGLOBAL g, void *memp, size_t size)
{
    PPOOLHEADER pph;           // Points on area header.

    if (!memp) // Allocation is to be done in the Sarea
        memp = g->Sarea;

    size = ((size + 7) / 8) * 8; /* Round up size to multiple of 8 */
    pph = (PPOOLHEADER)memp;

    if ((uint)size > pph->FreeBlk) { /* Not enough memory left in pool */
        sprintf(g->Message,
            "Not enough memory in Work area for request of %d (used=%d free=%d)",
            (int)size, pph->To_Free, pph->FreeBlk);
        return NULL;
    } // endif size

    // Do the suballocation the simplest way
    memp = MakePtr(memp, pph->To_Free); // Points to sub_allocated block
    pph->To_Free += size;               // New offset of pool free block
    pph->FreeBlk -= size;              // New size of pool free block

    return (memp);
} // end of PlgDBSubAlloc

```

You can create the file by copy/paste from the above.

Set all the additional include directories to the MariaDB include directories used in plugin compiling plus the reference of the storage/connect directories, and compile like any other UDF giving any name to the made library module (I used

jsonudf.dll

on Windows).

Then you can create the functions using this name as the soname parameter.

There are some restrictions when using the UDFs this way:

- The `connect_json_grp_size` variable cannot be accessed. The group size is set and retrieved using the `jsonset_grp_size` and `jsonget_grp_size` functions (previously 100).
- In case of error, warnings are replaced by messages sent to stderr.
- No trace.

## 4.3.7.13 CONNECT System Variables

### Contents

1. [connect\\_class\\_path](#)
2. [connect\\_cond\\_push](#)
3. [connect\\_conv\\_size](#)
4. [connect\\_default\\_depth](#)
5. [connect\\_default\\_prec](#)
6. [connect\\_enable\\_mongo](#)
7. [connect\\_exact\\_info](#)
8. [connect\\_force\\_bson](#)
9. [connect\\_idx\\_map](#)
10. [connect\\_java\\_wrapper](#)
11. [connect\\_json\\_all\\_path](#)
12. [connect\\_json\\_grp\\_size](#)
13. [connect\\_json\\_null](#)
14. [connect\\_jvm\\_path](#)
15. [connect\\_type\\_conv](#)
16. [connect\\_use\\_tempfile](#)
17. [connect\\_work\\_size](#)
18. [connect\\_xtrace](#)

This page documents system variables related to the [CONNECT storage engine](#). See [Server System Variables](#) for a complete list of system variables and instructions on setting them.

See also the [Full list of MariaDB options, system and status variables](#).

#### connect\_class\_path

- **Description:** Java class path
- **Commandline:**  
`--connect-class-path=value`
- **Scope:** Global
- **Dynamic:**
- **Data Type:**  
`string`
- **Default Value:**

---

#### connect\_cond\_push

- **Description:** Enable condition pushdown
- **Commandline:**  
`--connect-cond-push={0|1}`
- **Scope:** Global, Session
- **Dynamic:** Yes
- **Data Type:**  
`boolean`
- **Default Value:**  
`ON`
- **Introduced:** [MariaDB 10.3.6](#) , [MariaDB 10.2.14](#)

## connect\_conv\_size

- **Description:** The size of the `VARCHAR` created when converting from a `TEXT` type. See `connect_type_conv`.

- **Commandline:**

--connect-conv-size=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

◦ >= MariaDB 10.4.8 , MariaDB 10.3.18 , MariaDB 10.2.27 :

1024

◦ <= MariaDB 10.4.7 , MariaDB 10.3.17 , MariaDB 10.2.26 :

8192

- **Range:**

0

to

65500

## connect\_default\_depth

- **Description:** Default depth used by Json, XML and Mongo discovery.

- **Commandline:**

--connect-default-depth=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

5

- **Range:**

-1

to

16

- **Introduced:** MariaDB 10.5.7 , MariaDB 10.4.16 , MariaDB 10.3.26 , MariaDB 10.2.35

## connect\_default\_prec

- **Description:** Default precision used for doubles.

- **Commandline:**

--connect-default-prec=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

6

- **Range:**

0

to

16

- **Introduced:** MariaDB 10.5.9 , MariaDB 10.4.18 , MariaDB 10.3.28 , MariaDB 10.2.37
- 

connect\_enable\_mongo

- **Description:** Enable the [Mongo table type](#) .

- **Commandline:**

    --connect-enable-mongo={0|1}

- **Scope:** Global, Session

- **Dynamic:**

- **Data Type:**

    boolean

- **Default Value:**

    OFF

- **Introduced:** MariaDB 10.3.2 , MariaDB 10.2.9

- **Removed:** MariaDB 10.3.3
- 

connect\_exact\_info

- **Description:** Whether the CONNECT engine should return an exact record number value to information queries. It is OFF by default because this information can take a very long time for large variable record length tables or for remote tables, especially if the remote server is not available. It can be set to ON when exact values are desired, for instance when querying the repartition of rows in a partition table.

- **Commandline:**

    --connect-exact-info={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

---

connect\_force\_bson

- **Description:** Force using BSON for JSON tables. Starting with these releases, the internal way JSON was parsed and handled was changed. The main advantage of the new way is to reduce the memory required to parse JSON (from 6 to 10 times the size of the JSON source to now only 2 to 4 times). However, this is in Beta mode and JSON tables are still handled using the old mode. To use the new mode, tables should be created with TABLE\_TYPE=BSON, or by setting this session variable to 1 or ON. Then, all JSON tables will be handled as BSON. This is temporary until the new way replaces the old way by default.

- **Commandline:**

    --connect-force-bson={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

- **Introduced:** MariaDB 10.5.9 , MariaDB 10.4.18 , MariaDB 10.3.28 , MariaDB 10.2.37
- 

connect\_idx\_map

- **Description:** Enable file mapping for index files. To accelerate the indexing process, CONNECT makes an index structure in memory from the

index file. This can be done by reading the index file or using it as if it was in memory by “file mapping”. Set to 0 (file read, the default) or 1 (file mapping).

- **Commandline:**

```
--connect-idx-map=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

connect\_java\_wrapper

- **Description:** Java wrapper.

- **Commandline:**

```
--connect-java-wrapper=val
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:**

wrappers/JdbcInterface

- **Introduced:** Connect 1.05.0001, [MariaDB 10.2.4](#)

---

connect\_json\_all\_path

- **Description:** Discovery to generate json path for all columns if ON (the default) or do not when the path is the column name.

- **Commandline:**

```
--connect-json-all-path={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Data Type:**

boolean

- **Default Value:**

ON

- **Introduced:** [MariaDB 10.5.7](#) , [MariaDB 10.4.16](#) , [MariaDB 10.3.26](#) , [MariaDB 10.2.35](#)

---

connect\_json\_grp\_size

- **Description:** Max number of rows for JSON aggregate functions.

- **Commandline:**

```
--connect-json-grp-size=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

50

```
(>= Connect 1.7.0003),  
10  
(<= Connect 1.7.0002)  
• Range:  
1  
to  
2147483647
```

---

### connect\_json\_null

- **Description:** Representation of JSON null values.

- **Commandline:**

```
--connect-json-null=value
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
string
```

- **Default Value:**

```
<null>
```

- **Introduced:** MariaDB 10.2.8
- 

### connect\_jvm\_path

- **Description:** Path to JVM library.

- **Commandline:**

```
--connect-jvm_path=value
```

- **Scope:** Global

- **Dynamic:**

- **Data Type:**

```
string
```

- **Default Value:**

- **Introduced:** Connect 1.04.0006
- 

### connect\_type\_conv

- **Description:** Determines the handling of TEXT columns.

```
◦
```

```
NO
```

: The default until Connect 1.06.005, no conversion takes place, and a TYPE\_ERROR is returned, resulting in a "not supported" message.

```
◦
```

```
YES
```

: The default from Connect 1.06.006. The column is internally converted to a column declared as VARCHAR(n), n being the value of connect\_conv\_size .

```
◦
```

```
FORCE
```

(>= Connect 1.06.006): Also convert ODBC blob columns to TYPE\_STRING.

```
◦
```

```
SKIP
```

: No conversion. When the column declaration is provided via Discovery (meaning the CONNECT table is created without a column description), this column is not generated. Also applies to ODBC tables.

- **Commandline:**

```
--connect-type-conv=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

enum

- **Valid Values:**

NO

,

YES

,

FORCE

or

SKIP

- **Default Value:**

YES

---

## connect\_use\_tempfile

- **Description:**

◦

NO

: The first algorithm is always used. Because it can cause errors when updating variable record length tables, this value should be set only for testing.

◦

AUTO

: This is the default value. It leaves CONNECT to choose the algorithm to use. Currently it is equivalent to

NO

, except when updating variable record length tables ( [DOS](#) , [CSV](#) or [FMT](#) ) with file mapping forced to OFF.

◦

YES

: Using a temporary file is chosen with some exceptions. These are when file mapping is ON, for [VEC](#) tables and when deleting from [DBF](#) tables (soft delete). For variable record length tables, file mapping is forced to OFF.

◦

FORCE

: Like YES but forces file mapping to be OFF for all table types.

◦

TEST

: Reserved for CONNECT development.

- **Commandline:**

--connect-use-tempfile=#

- **Scope:** Session

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

AUTO

---

## connect\_work\_size

- **Description:** Size of the CONNECT work area used for memory allocation. Permits allocating a larger memory sub-allocation space when dealing with very large if sub-allocation fails. If the specified value is too big and memory allocation fails, the size of the work area remains but the variable value is not modified and should be reset.

- **Commandline:**

--connect-work-size=#

- **Scope:** Global, Session (Session-only from CONNECT 1.03.005)

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

67108864

- **Range:**

4194304

upwards, depending on the physical memory size

---

connect\_xtrace

- **Description:** Console trace value. Set to

0

(no trace), or to other values if a console tracing is desired. Note that to test this handler, MariaDB should be executed with the `--console` parameter because CONNECT prints some error and trace messages on the console. In some Linux versions, this is re-routed into the error log file. Console tracing can be set on the command line or later by names or values. Valid values (from Connect 1.06.006) include:

◦

0

: No trace

◦

YES

or

1

: Basic trace

◦

MORE

or

2

: More tracing

◦

INDEX

or

4

: Index construction

◦

MEMORY

or

8

: Allocating and freeing memory

◦

SUBALLOC

or

16

: Sub-allocating in work area

◦

QUERY

or

32

: Constructed query sent to external server

◦

STMT

or

64

: Currently executing statement

◦

HANDLER

or

128

: Creating and dropping CONNECT handlers

◦

BLOCK

or

256

: Creating and dropping CONNECT objects

◦

MONGO

or

512

: Mongo and REST (from Connect 1.06.0010 ) tracing

- For example:

◦

`set global connect_xtrace=0;`

```

No trace

o
    set global connect_xtrace='YES';

By name

o
    set global connect_xtrace=1;

By value

o
    set global connect_xtrace='QUERY,STMT';

By name

o
    set global connect_xtrace=96;

By value

o
    set global connect_xtrace=1023;

Trace all

```

- **Commandline:**

--connect-xtrace=#

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**

set

- **Default Value:**

0

- **Valid Values:** See description

## 4.3.7.14 JSON Sample Files

### Contents

1. [Expense.json](#)
2. [OEM example](#)
  1. [tabfic.h](#)
  2. [tabfic.cpp](#)
  3. [tabfic.def](#)
3. [JSON UDFs in a separate library](#)

### Expense.json

```
[
  {
    "WHO": "Joe",
    "WEEK": [
      {
        "NUMBER": 3,
        "EXPENSE": [
          {
            "WHAT": "Beer",
            "AMOUNT": 18.00
          },
          {
            "WHAT": "Food"
          }
        ]
      }
    ]
  }
]
```

```

        "WHAT": "Food",
        "AMOUNT": 12.00
    },
    {
        "WHAT": "Food",
        "AMOUNT": 19.00
    },
    {
        "WHAT": "Car",
        "AMOUNT": 20.00
    }
]
},
{
    "NUMBER": 4,
    "EXPENSE": [
        {
            "WHAT": "Beer",
            "AMOUNT": 19.00
        },
        {
            "WHAT": "Beer",
            "AMOUNT": 16.00
        },
        {
            "WHAT": "Food",
            "AMOUNT": 17.00
        },
        {
            "WHAT": "Food",
            "AMOUNT": 17.00
        },
        {
            "WHAT": "Beer",
            "AMOUNT": 14.00
        }
    ]
},
{
    "NUMBER": 5,
    "EXPENSE": [
        {
            "WHAT": "Beer",
            "AMOUNT": 14.00
        },
        {
            "WHAT": "Food",
            "AMOUNT": 12.00
        }
    ]
},
{
    "WHO": "Beth",
    "WEEK": [
        {
            "NUMBER": 3,
            "EXPENSE": [
                {
                    "WHAT": "Beer",
                    "AMOUNT": 16.00
                }
            ]
        },
        {
            "NUMBER": 4,
            "EXPENSE": [
                {
                    "WHAT": "Food",
                    "AMOUNT": 17.00
                },
                {
                    "WHAT": "Beer",
                    "AMOUNT": 15.00
                }
            ]
        },
        {
            "WHO": "Beth"
        }
    ]
}

```

```
{
    "NUMBER": 5,
    "EXPENSE": [
        {
            "WHAT": "Food",
            "AMOUNT": 12.00
        },
        {
            "WHAT": "Beer",
            "AMOUNT": 20.00
        }
    ]
},
{
    "WHO": "Janet",
    "WEEK": [
        {
            "NUMBER": 3,
            "EXPENSE": [
                {
                    "WHAT": "Car",
                    "AMOUNT": 19.00
                },
                {
                    "WHAT": "Food",
                    "AMOUNT": 18.00
                },
                {
                    "WHAT": "Beer",
                    "AMOUNT": 18.00
                }
            ]
        },
        {
            "NUMBER": 4,
            "EXPENSE": [
                {
                    "WHAT": "Car",
                    "AMOUNT": 17.00
                }
            ]
        },
        {
            "NUMBER": 5,
            "EXPENSE": [
                {
                    "WHAT": "Beer",
                    "AMOUNT": 14.00
                },
                {
                    "WHAT": "Car",
                    "AMOUNT": 12.00
                },
                {
                    "WHAT": "Beer",
                    "AMOUNT": 19.00
                },
                {
                    "WHAT": "Food",
                    "AMOUNT": 12.00
                }
            ]
        }
    ]
}
]
```

## OEM example

This is an example showing how an OEM table can be implemented. It is out of the scope of this document to explain how it works and to be a full guide on writing OEM tables for CONNECT.

The header File tabfic.h:

```
// TABFIC.H      Olivier Bertrand    2008-2010
// External table type to read FIC files

#define TYPE_AM_FIC  (AMT)129

typedef class FICDEF *PFICDEF;
typedef class TDBFIC *PTDBFIC;
typedef class FICCOL *PFICCOL;

/* ----- FIC classes ----- */

/***
/* FIC: OEM table to read FIC files.          */
/**/

/***
/* This function is exported from the Tabfic.dll   */
/**/

extern "C" PTABDEF __stdcall GetFIC(PGLOBAL g, void *memp);

/***
/* FIC table definition class.                  */
/**/

class FICDEF : public DOSDEF {           /* Logical table description */
public:
    friend class TDBFIC;
    // Constructor
    FICDEF(void) {Pseudo = 3;}

    // Implementation
    virtual const char *GetType(void) {return "FIC";}

    // Methods
    virtual BOOL DefineAM(PGLOBAL g, LPCSTR am, int poff);
    virtual PTDB GetTable(PGLOBAL g, MODE m);

protected:
    // No Members
}; // end of class FICDEF

/***
/* This is the class declaration for the FIC table. */
/**/

class TDBFIC : public TDBFIX {
public:
    friend class FICCOL;
    // Constructor
    TDBFIC(PFICDEF tdp);

    // Implementation
    virtual AMT GetAmType(void) {return TYPE_AM_FIC;}

    // Methods
    virtual void ResetDB(void);
    virtual int RowNumber(PGLOBAL g, BOOL b = FALSE);

    // Database routines
    virtual PCOL MakeCol(PGLOBAL g, PCOLDEF cdp, PCOL cprec, int n);
    virtual BOOL OpenDB(PGLOBAL g, PSQL sqlp);
    virtual int ReadDB(PGLOBAL g);
    virtual int WriteDB(PGLOBAL g);
    virtual int DeleteDB(PGLOBAL g, int irc);

protected:
    // Members
    int ReadMode;      // To read soft deleted lines
    int Rows;           // Used for RowID
}; // end of class TDBFIC

/***
/* Class FICCOL: for Monetary columns.          */
/**/

class FICCOL : public DOSCOL {
public:
    // Constructors
    FICCOL(PGLOBAL g, PCOLDEF cd, PTDB tdp, int n)
```

```

FICCOL(PGLOBAL g, PCOLDEF cdp, PTDB tdp,
       PCOL cprec, int i, PSZ am = "FIC");

// Implementation
virtual int GetAmType(void) {return TYPE_AM_FIC;}

// Methods
virtual void ReadColumn(PGLOBAL g);

protected:
    // Members
    char Fmt;           // The column format
}; // end of class FICCOL

```

## tabfic.cpp

The source File tabfic.cpp:

```

/**
 * FIC: OEM table to read FIC files.                                     */
/**

#if defined(WIN32)
#define WIN32_LEAN_AND_MEAN      // Exclude rarely-used stuff
#include <windows.h>
#endif   // WIN32
#include "global.h"
#include "plgdbsem.h"
#include "reldef.h"
#include "filamfix.h"
#include "tabfix.h"
#include "tabfic.h"

int TDB::Tnum;
int DTVAL::Shift;

/**
 * Initialize the CSORT static members.                                     */
/**

int    CSORT::Limit = 0;
double CSORT::Lg2 = log(2.0);
size_t CSORT::Cpn[1000] = {0};      /* Precalculated cmpnum values */

/* ----- Implementation of the FIC subtype ----- */

/**
 * This function is exported from the DLL.                                 */
/**

PTABDEF __stdcall GetFIC(PGLOBAL g, void *memp)
{
    return new(g, memp) FICDEF;
} // end of GetFIC

/* ----- Implementation of the FIC classes ----- */

/**
 * DefineAM: define specific AM block values from FIC file.          */
/**

BOOL FICDEF::DefineAM(PGLOBAL g, LPCSTR am, int poff)
{
    ReadMode = GetIntCatInfo("Readmode", 0);

    // Indicate that we are a BIN format
    return DOSDEF::DefineAM(g, "BIN", poff);
} // end of DefineAM

/**
 * GetTable: makes a new TDB of the proper type.                         */
/**

PTDB FICDEF::GetTable(PGLOBAL g, MODE m)
{
    return new(g) TDBFIC(this);
} // end of GetTable

/* ----- */

/**
 * Implementation of the TDBFIC class.                                    */
*/

```

```

/***/
TDBFIC::TDBFIC(PFICDEF tdp) : TDBFIX(tdp, NULL)
{
    ReadMode = tdp->ReadMode;
    Rows = 0;
} // end of TDBFIC constructor

/***/
/*  Allocate FIC column description block.          */
/***/
PCOL TDBFIC::MakeCol(PGLOBAL g, PCOLDEF cdp, PCOL cprec, int n)
{
    PCOL colp;

    // BINCOL is alright except for the Monetary format
    if (cdp->GetFmt() && toupper(*cdp->GetFmt()) == 'M')
        colp = new(g) FICCOL(g, cdp, this, cprec, n);
    else
        colp = new(g) BINCOL(g, cdp, this, cprec, n);

    return colp;
} // end of MakeCol

/***/
/*  RowNumber: return the ordinal number of the current row.      */
/***/
int TDBFIC::RowNumber(PGLOBAL g, BOOL b)
{
    return (b) ? Txfp->GetRowID() : Rows;
} // end of RowNumber

/***/
/*  FIC Access Method reset table for re-opening.          */
/***/
void TDBFIC::ResetDB(void)
{
    Rows = 0;
    TDBFIX::ResetDB();
} // end of ResetDB

/***/
/*  FIC Access Method opening routine.          */
/***/
BOOL TDBFIC::OpenDB(PGLOBAL g, PSQL sqlp)
{
    if (Use == USE_OPEN) {
        // Table already open, just replace it at its beginning.
        return TDBFIX::OpenDB(g);
    } // endif use

    if (Mode != MODE_READ) {
        // Currently FIC tables cannot be modified.
        strcpy(g->Message, "FIC tables are read only");
        return TRUE;
    } // endif Mode

    /* Physically open the FIC file.          */
    if (TDBFIX::OpenDB(g))
        return TRUE;

    Use = USE_OPEN;
    return FALSE;
} // end of OpenDB

/***/
/*  ReadDB: Data Base read routine for FIC access method.          */
/***/
int TDBFIC::ReadDB(PGLOBAL g)
{
    int rc;

    /* Now start the reading process.          */
    do {
        rc = TDBFIX::ReadDB(g);

```

```

    /* - read_linecode(s),
} while (rc == RC_OK && ((ReadMode == 0 && *To_Line == '*') ||
    (ReadMode == 2 && *To_Line != '*')));

Rows++;
return rc;
} // end of ReadDB

/***/
/* WriteDB: Data Base write routine for FIC access methods.      */
/***/
int TDBFIC::WriteDB(PGLOBAL g)
{
    strcpy(g->Message, "FIC tables are read only");
    return RC_FX;
} // end of WriteDB

/***/
/* Data Base delete line routine for FIC access methods.      */
/***/
int TDBFIC::DeleteDB(PGLOBAL g, int irc)
{
    strcpy(g->Message, "Delete not enabled for FIC tables");
    return RC_FX;
} // end of DeleteDB

// ----- FICCOL functions -----
/***/
/* FICCOL public constructor.          */
/***/
FICCOL::FICCOL(PGLOBAL g, PCOLDEF cdp, PTDB tdbp, PCOL cprec, int i,
                PSZ am) : DOSCOL(g, cdp, tdbp, cprec, i, am)
{
    // Set additional FIC access method information for column.
    Fmt = toupper(*cdp->GetFmt()); // Column format
} // end of FICCOL constructor

/***/
/* Handle the monetary value of this column. It is a big integer */
/* that represents the value multiplied by 1000.           */
/* In this function we translate it to a double float value. */
/***/
void FICCOL::ReadColumn(PGLOBAL g)
{
    char    *p;
    int     rc;
    uint    n;
    double  fmon;
    PTDBFIC tdbp = (PTDBFIC)To_Tdb;

/* If physical reading of the line was deferred, do it now. */
if (!tdbp->IsRead())
    if ((rc = tdbp->ReadBuffer(g)) != RC_OK) {
        if (rc == RC_EF)
            sprintf(g->Message, MSG(INV_DEF_READ), rc);

        longjmp(g->jumper[g->jump_level], 11);
    } // endif

p = tdbp->To_Line + Deplac;

/* Set Value from the line field. */
if ((*((SHORT*)(p + 8)) < 0) {
    n = ~*(SHORT*)(p + 8);
    fmon = (double)n;
    fmon *= 4294967296.0;
    n = ~*(int*)(p + 4);
    fmon += (double)n;
    fmon *= 4294967296.0;
    n = ~*(int*)p;
    fmon += (double)n;
    fmon++;
    fmon /= 1000000.0;
}

```

```

fmon = -fmon;
} else {
    fmon = ((double)*(USHORT*)(p + 8));
    fmon *= 4294967296.0;
    fmon += ((double)*(ULONG*)(p + 4));
    fmon *= 4294967296.0;
    fmon += ((double)*(ULONG*)p);
    fmon /= 1000000.0;
} // enif neg

Value->SetValue(fmon);
} // end of ReadColumn

```

## tabfic.def

The file tabfic.def: (required only on Windows)

```

LIBRARY      TABFIC
DESCRIPTION 'FIC files'
EXPORTS
    GetFIC      @1

```

## JSON UDFs in a separate library

Although the JSON UDF's can be nicely included in the CONNECT library module, there are cases when you may need to have them in a separate library.

This is when CONNECT is compiled embedded, or if you want to test or use these UDF's with other MariaDB versions not including them.

To make it, you need to have access to the last MariaDB source code. Then, make a project containing these files:

1. jsonudf.cpp
2. json.cpp
3. value.cpp
4. osutil.c
5. plugutil.c
6. maputil.cpp
7. jsonutil.cpp

jsonutil.cpp is not distributed with the source code, you will have to make it from the following:

```

#include "my_global.h"
#include "mysqld.h"
#include "plugin.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>

#include "global.h"

extern "C" int GetTraceValue(void) { return 0; }
uint GetJsonGrpSize(void) { return 100; }

/***/
/* These replace missing function of the (not used) DTVAL class. */
/***/
typedef struct _datpar *PDTp;
PDTp MakeDateFormat(PGLOBAL, PSZ, bool, bool, int) { return NULL; }
int ExtractDate(char*, PDTp, int, int val[6]) { return 0; }

#ifndef __WIN__
my_bool CloseFileHandle(HANDLE h)
{
    return !CloseHandle(h);
} /* end of CloseFileHandle */

#else /* UNIX */
my_bool CloseFileHandle(HANDLE h)
{
    return (close(h)) ? TRUE : FALSE;
} /* end of CloseFileHandle */

int GetLastError()
{
    return errno;
} /* end of GetLastError */

#endif // UNIX

/***/
/* Program for sub-allocating one item in a storage area. */
/* Note: This function is equivalent to PlugSubAlloc except that in */
/* case of insufficient memory, it returns NULL instead of doing a */
/* long jump. The caller must test the return value for error. */
/***/
void *PlgDBSubAlloc(PGLOBAL g, void *memp, size_t size)
{
    PPOOLHEADER pph;           // Points on area header.

    if (!memp) // Allocation is to be done in the Sarea
        memp = g->Sarea;

    size = ((size + 7) / 8) * 8; /* Round up size to multiple of 8 */
    pph = (PPOOLHEADER)memp;

    if ((uint)size > pph->FreeBlk) { /* Not enough memory left in pool */
        sprintf(g->Message,
            "Not enough memory in Work area for request of %d (used=%d free=%d)",
            (int)size, pph->To_Free, pph->FreeBlk);
        return NULL;
    } // endif size

    // Do the suballocation the simplest way
    memp = MakePtr(memp, pph->To_Free); // Points to sub_allocated block
    pph->To_Free += size;               // New offset of pool free block
    pph->FreeBlk -= size;              // New size of pool free block

    return (memp);
} // end of PlgDBSubAlloc

```

You can create the file by copy/paste from the above.

Set all the additional include directories to the MariaDB include directories used in plugin compiling plus the reference of the storage/connect directories, and compile like any other UDF giving any name to the made library module (I used jsonudf.dll on Windows)

Then you can create the functions using this name as the soname parameter.

There are some restrictions when using the UDF's this way:

- The connect\_json\_grp\_size variable cannot be accessed. The group size is set to 100.
- In case of error, warnings are replaced by messages sent to stderr.
- No trace.

## 4.3.8 CSV

### 4.3.8.1 CSV Overview

#### Contents

1. [The CSV storage engine and logging to tables](#)
2. [CSV Storage Engine files](#)
3. [Limitations](#)
4. [Examples](#)
5. [See Also](#)

The CSV Storage Engine can read and append to files stored in CSV (comma-separated-values) format.

However, since [MariaDB 10.0](#), a better storage engine is able to read and write such files: [CONNECT](#).

## The CSV storage engine and logging to tables

The CSV storage engine is the default storage engine when using [logging of SQL queries](#) to tables.

```
mysqld --log-output=table
```

## CSV Storage Engine files

When you create a table using the CSV storage engine, three files are created:

- `<table_name>.frm`
- `<table_name>.CSV`
- `<table_name>.CSM`

The

`.frm`  
file is the table format file.

The

`.CSV`  
file is a plain text file. Data you enter into the table is stored as plain text in comma-separated-values format.

The

`.CSM`  
file stores metadata about the table such as the state and the number of rows in the table.

## Limitations

- CSV tables do not support indexing.
- CSV tables cannot be partitioned.
- Columns in CSV tables must be declared as NOT NULL.
- No [transactions](#).
- The original CSV-format does not enable IETF-compatible parsing of embedded quote and comma characters. From [MariaDB 10.1.8](#), it is possible to do so by setting the [IETF\\_QUOTES](#) option when creating a table.

## Examples

Forgetting to add NOT NULL:

```
CREATE TABLE csv_test (x INT, y DATE, z CHAR(10)) ENGINE=CSV;
ERROR 1178 (42000): The storage engine for the table doesn't support nullable columns
```

Creating, inserting and selecting:

```
CREATE TABLE csv_test (
  x INT NOT NULL, y DATE NOT NULL, z CHAR(10) NOT NULL
) ENGINE=CSV;
```

```
INSERT INTO csv_test VALUES
(1,CURDATE(),'one'),
(2,CURDATE(),'two'),
(3,CURDATE(),'three');
```

```
SELECT * FROM csv_test;
+---+-----+-----+
| x | y      | z      |
+---+-----+-----+
| 1 | 2011-11-16 | one   |
| 2 | 2011-11-16 | two   |
| 3 | 2011-11-16 | three |
+---+-----+-----+
```

Viewing in a text editor:

```
$ cat csv_test.CSV
1,"2011-11-16","one"
2,"2011-11-16","two"
3,"2011-11-16","three"
```

## See Also

- [Checking and Repairing CSV Tables](#)

### 4.3.8.2 Checking and Repairing CSV Tables

CSV tables support the [CHECK TABLE](#) and [REPAIR TABLE](#) statements.

CHECK TABLE will mark the table as corrupt if it finds a problem, while REPAIR TABLE will restore rows until the first corrupted row, discarding the rest.

## Examples

```
CREATE TABLE csv_test (
  x INT NOT NULL, y DATE NOT NULL, z CHAR(10) NOT NULL
) ENGINE=CSV;

INSERT INTO csv_test VALUES
(1,CURDATE(),'one'),
(2,CURDATE(),'two'),
(3,CURDATE(),'three');
```

```
SELECT * FROM csv_test;
+---+-----+-----+
| x | y      | z      |
+---+-----+-----+
| 1 | 2013-07-08 | one   |
| 2 | 2013-07-08 | two   |
| 3 | 2013-07-08 | three |
+---+-----+-----+
```

Using an editor, the actual file will look as follows

```
$ cat csv_test.CSV
1,"2013-07-08","one"
2,"2013-07-08","two"
3,"2013-07-08","three"
```

Let's introduce some corruption with an unwanted quote in the 2nd row:

```
1,"2013-07-08","one"
2,"2013-07-08","two"
3,"2013-07-08","three"
```

```
CHECK TABLE csv_test;
+-----+-----+-----+
| Table | Op   | Msg_type | Msg_text |
+-----+-----+-----+
| test.csv_test | check | error    | Corrupt  |
+-----+-----+-----+
```

We can repair this, but all rows from the corrupt row onwards will be lost:

```
REPAIR TABLE csv_test;
+-----+-----+-----+
| Table | Op   | Msg_type | Msg_text |
+-----+-----+-----+
| test.csv_test | repair | Warning  | Data truncated for column 'x' at row 2 |
| test.csv_test | repair | status   | OK        |
+-----+-----+-----+

SELECT * FROM csv_test;
+---+---+
| x | y      | z   |
+---+---+
| 1 | 2013-07-08 | one |
+---+---+
```

## 4.3.9 FederatedX

### 4.3.9.1 About FederatedX

MariaDB starting with [5.1](#)

The FederatedX storage engine was first released in [MariaDB 5.1](#).

The FederatedX storage engine is a fork of MySQL's [Federated storage engine](#), which is no longer being developed by Oracle. The original purpose of FederatedX was to keep this storage engine's development progressing-- to both add new features as well as fix old bugs.

Since [MariaDB 10.0](#), the `CONNECT` storage engine also allows access to a remote database via MySQL or ODBC connection (table types: [MYSQL](#), [ODBC](#)). However, in the current implementation there are several limitations.

#### Contents

1. [What is the FederatedX storage engine?](#)
2. [History](#)
3. [Installing the Plugin](#)
4. [Uninstalling the Plugin](#)
5. [How FederatedX works](#)
  1. [Internal workings of FederatedX](#)
    1. [FederatedX table creation](#)
  2. [Method calls](#)
    1. [SELECT](#)
    2. [INSERT](#)
    3. [UPDATE](#)
6. [FederatedX capabilities and limitations](#)
7. [How do you use FederatedX?](#)
  1. [How to see the storage engine in action](#)
  8. [How do I create a federated server?](#)
  9. [How does FederatedX differ from the old Federated Engine?](#)
10. [Where can I get FederatedX](#)
  1. [What are the plans for FederatedX?](#)

## What is the FederatedX storage engine?

The FederatedX Storage Engine is a storage engine that works with both MariaDB and MySQL. Where other storage engines are built as interfaces to lower-level file-based data stores, FederatedX uses libmysql to talk to the data source, the data source being a remote RDBMS. Currently, since FederatedX only uses libmysql, it can only talk to another MySQL RDBMS. The plan is of course to be able to use other RDBMS systems as a data

source. There is an existing project Federated ODBC which was able to use PostgreSQL as a remote data source, and it is this type of functionality which will be brought to FederatedX in subsequent versions.

## History

The history of FederatedX is derived from the History of Federated. Cisco needed a MySQL storage engine that would allow them to consolidate remote tables on some sort of routing device, being able to interact with these remote tables as if they were local to the device, but not actually on the device, since the routing device had only so much storage space. The first prototype of the Federated Storage Engine was developed by JD (need to check on this- Brian Aker can verify) using the HANDLER interface. Brian handed the code to Patrick Galbraith and explained how it needed to work, and with Brian and Monty's tutelage and Patrick had a working Federated Storage Engine with MySQL 5.0. Eventually, Federated was released to the public in a MySQL 5.0 release.

When MySQL 5.1 became the production release of MySQL, Federated had more features and enhancements added to it, namely:

- New Federated SERVER added to the parser. This was something Cisco needed that made it possible to change the connection parameters for numerous Federated tables at once without having to alter or re-create the Federated tables.
- Basic Transactional support-- for supporting remote transactional tables
- Various bugs that needed to be fixed from MySQL 5.0
- Plugin capability

In [MariaDB 10.0.2](#) FederatedX got support for assisted [table discovery](#).

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'ha_federatedx';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = ha_federatedx
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'ha_federatedx';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## How FederatedX works

Every storage engine has to implement derived standard handler class API methods for a storage engine to work. FederatedX is no different in that regard. The big difference is that FederatedX needs to implement these handler methods in such as to construct SQL statements to run on the remote server and if there is a result set, process that result set into the internal handler format so that the result is returned to the user.

### Internal workings of FederatedX

Normal database files are local and as such: You create a table called 'users', a file such as 'users.MYD' is created. A handler reads, inserts, deletes, updates data in this file. The data is stored in particular format, so to read, that data has to be parsed into fields, to write, fields have to be stored in this format to write to this data file.

With the FederatedX storage engine, there will be no local files for each table's data (such as .MYD). A foreign database will store the data that would normally be in this file. This will necessitate the use of MySQL client API to read, delete, update, insert this data. The data will have to be retrieved via an SQL call "

```
SELECT
```

```
*
```

```
FROM
```

```
users
```

". Then, to read this data, it will have to be retrieved via

```
mysql_fetch_row
```

one row at a time, then converted from the column in this select into the format that the handler expects.

The basic functionality of how FederatedX works is:

- The user issues an SQL statement against the local federatedX table. This statement is parsed into an item tree
- FederatedX uses the mysql handler API to implement the various methods required for a storage engine. It has access to the item tree for the SQL statement issued, as well as the Table object and each of its Field members. At
- With this information, FederatedX constructs an SQL statement
- The constructed SQL statement is sent to the Foreign data source through libmysql using the mysql client API
- The foreign database reads the SQL statement and sends the result back through the mysql client API to the origin
- If the original SQL statement has a result set from the foreign data source, the FederatedX storage engine iterates through the result set and converts each row and column to the internal handler format
- If the original SQL statement only returns the number of rows returned (affected\_rows), that number is added to the table stats which results in the user seeing how many rows were affected.

### FederatedX table creation

The create table will simply create the .frm file, and within the

```
CREATE
```

```
TABLE
```

SQL statement, there SHALL be any of the following :

```
connection=scheme://username:password@hostname:port/database tablename  
connection=scheme://username@hostname/database tablename  
connection=scheme://username:password@hostname/database tablename  
connection=scheme://username:password@hostname/database tablename
```

Or using the syntax introduced in MySQL versions 5.1 for a Federated server (SQL/MED Spec xxxx)

```
connection="connection_one"  
connection="connection_one/table_foo"
```

An example of a connect string specifying all the connection parameters would be:

```
connection=mysql://username:password@hostname:port/database tablename
```

Or, using a Federated server, first a server is created:

```
create server 'server_one' foreign data wrapper 'mysql' options  
(HOST '127.0.0.1',  
DATABASE 'db1',  
USER 'root',  
PASSWORD '',  
PORT 3306,  
SOCKET '',  
OWNER 'root');
```

Then the FederatedX table is created specifying the newly created Federated server:

```
CREATE TABLE federatedx.t1 (  
`id` int(20) NOT NULL,  
`name` varchar(64) NOT NULL default ''  
)  
ENGINE="FEDERATED" DEFAULT CHARSET=latin1  
CONNECTION='server_one';
```

(Note that in MariaDB, the original Federated storage engine is replaced with the new FederatedX storage engine. And for backward compatibility, the old name "FEDERATED" is used in create table. So in MariaDB, the engine type should be given as "FEDERATED" without an extra "X", not "FEDERATEDX").

The equivalent of above, if done specifying all the connection parameters

```
CONNECTION="mysql://root@127.0.0.1:3306/db1/t1"
```

You can also change the server to point to a new schema:

```
ALTER SERVER 'server_one' options(DATABASE 'db2');
```

All subsequent calls to any FederatedX table using the 'server\_one' will now be against db2.t1! Guess what? You no longer have to perform an alter table in order to point one or more FederatedX tables to a new server!

This

```
connection="connection string"  
is necessary for the handler to be able to connect to the foreign server, either by URL, or by server name.
```

## Method calls

One way to see how the FederatedX storage engine works is to compile a debug build of MariaDB and turn on a trace log. Using a two column table, with one record, the following SQL statements shown below, can be analyzed for what internal methods they result in being called.

**SELECT**

If the query is for instance "

```
SELECT
```

```
*
```

```
FROM
```

```
foo
```

", then the primary methods you would see with debug turned on would be first:

```
ha_federatedx::info
ha_federatedx::scan_time:
ha_federatedx::rnd_init: share->select_query SELECT * FROM foo
ha_federatedx::extra
```

Then for every row of data retrieved from the foreign database in the result set:

```
ha_federatedx::rnd_next
ha_federatedx::convert_row_to_internal_format
ha_federatedx::rnd_next
```

After all the rows of data that were retrieved, you would see:

```
ha_federatedx::rnd_end
ha_federatedx::extra
ha_federatedx::reset
```

## INSERT

If the query was "

```
INSERT
```

```
INTO
```

```
foo
```

```
(
```

```
id
```

```
,
```

```
ts
```

```
)
```

```
VALUES
```

```
(
```

```
2
```

```
,
```

```
now
```

```
());
```

", the trace would be:

```
ha_federatedx::write_row  
ha_federatedx::reset
```

## UPDATE

If the query was "

```
UPDATE
```

```
    foo
```

```
    SET
```

```
        ts
```

```
        =
```

```
        now
```

```
    ()
```

```
WHERE
```

```
    id
```

```
    =
```

```
    1
```

```
;
```

", the resultant trace would be:

```
ha_federatedx::index_init  
ha_federatedx::index_read  
ha_federatedx::index_read_idx  
ha_federatedx::rnd_next  
ha_federatedx::convert_row_to_internal_format  
ha_federatedx::update_row  
  
ha_federatedx::extra  
ha_federatedx::extra  
ha_federatedx::extra  
ha_federatedx::external_lock  
ha_federatedx::reset
```

## FederatedX capabilities and limitations

- Tables MUST be created on the foreign server prior to any action on those tables via the handler, first version. IMPORTANT: IF you MUST use the FederatedX storage engine type on the REMOTE end, make sure that the table you connect to IS NOT a table pointing BACK to your ORIGINAL table! You know and have heard the screeching of audio feedback? You know putting two mirrors in front of each other how the reflection continues for eternity? Well, need I say more?!
- There is no way for the handler to know if the foreign database or table has changed. The reason for this is that this database has to work like a data file that would never be written to by anything other than the database. The integrity of the data in the local table could be breached if there was any change to the foreign database.
- Support for SELECT, INSERT, UPDATE, DELETE indexes.
- No ALTER TABLE, DROP TABLE or any other Data Definition Language calls.
- Prepared statements will not be used in the first implementation, it remains to be seen whether the limited subset of the client API for the server

supports this.

- This uses SELECT, INSERT, UPDATE, DELETE and not HANDLER for its implementation.
- This will not work with the query cache.

## How do you use FederatedX?

To use this handler, it's very simple. You must have two databases running, either both on the same host, or on different hosts.

First, on the foreign database you create a table, for example:

```
CREATE TABLE test_table (
    id      int(20) NOT NULL auto_increment,
    name    varchar(32) NOT NULL default '',
    other   int(20) NOT NULL default '0',
    PRIMARY KEY (id),
    KEY name (name),
    KEY other_key (other)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Then, on the server that will be connecting to the foreign host (client), you create a federated table without specifying the table structure:

```
CREATE TABLE test_table ENGINE=FEDERATED CONNECTION='mysql://root@127.0.0.1:9306/federatedx/test_federatedx';
```

Notice the "ENGINE" and "CONNECTION" fields? This is where you respectively set the engine type, "FEDERATED" and foreign host information, this being the database your 'client' database will connect to and use as the "data file". Obviously, the foreign database is running on port 9306, so you want to start up your other database so that it is indeed on port 9306, and your FederatedX database on a port other than that. In my setup, I use port 5554 for FederatedX, and port 5555 for the foreign database.

Alternatively (or if you're using MariaDB before version 10.0.2) you specify the federated table structure explicitly:

```
CREATE TABLE test_table (
    id      int(20) NOT NULL auto_increment,
    name    varchar(32) NOT NULL default '',
    other   int(20) NOT NULL default '0',
    PRIMARY KEY (id),
    KEY name (name),
    KEY other_key (other)
) ENGINE=FEDERATED
DEFAULT CHARSET=latin1
CONNECTION='mysql://root@127.0.0.1:9306/federatedx/test_federatedx';
```

In this case the table structure must match exactly the table on the foreign server.

## How to see the storage engine in action

When developing this handler, I compiled the FederatedX database with debugging:

```
./configure --with-federatedx-storage-engine \
--prefix=/home/mysql/mysql-build/federatedx/ --with-debug
```

Once compiled, I did a 'make install' (not for the purpose of installing the binary, but to install all the files the binary expects to see in the directory I specified in the build with

```
--prefix=/home/code-dev/maria
```

Then, I started the foreign server:

```
/usr/local/mysql/bin/mysqld_safe \
--user=mysql --log=/tmp/mysqld.5555.log -P 5555
```

Then, I went back to the directory containing the newly compiled mysqld

```
<builddir>/sql/
, started up gdb:
```

```
gdb ./mysqld
```

Then, within the (gdb) prompt:

```
(gdb) run --gdb --port=5554 --socket=/tmp/mysqld.5554 --skip-innodb --debug
```

Next, I open several windows for each:

1. Tail the debug trace: tail -f /tmp/mysql.trace|grep ha\_fed
2. Tail the SQL calls to the foreign database: tail -f /tmp/mysql.5555.log
3. A window with a client open to the federatedx server on port 5554
4. A window with a client open to the federatedx server on port 5555

I would create a table on the client to the foreign server on port 5555, and then to the FederatedX server on port 5554. At this point, I would run whatever queries I wanted to on the FederatedX server, just always remembering that whatever changes I wanted to make on the table, or if I created new tables, that I would have to do that on the foreign server.

Another thing to look for is 'show variables' to show you that you have support for FederatedX handler support:

```
show variables like '%federat%'
```

and:

```
show storage engines;
```

Both should display the federatedx storage handler.

## How do I create a federated server?

A federated server is a way to have a foreign data source defined-- with all connection parameters-- so that you don't have to specify explicitly the connection parameters in a string.

For instance, say if you wanted to create a table, t1, that you would specify with

```
connection="mysql://patg@192.168.1.123/first_db/t1"
```

You could instead create this with a server:

```
create server 'server_one' foreign data wrapper 'mysql' options
(HOST '192.168.1.123',
DATABASE 'first_db',
USER 'patg',
PASSWORD '',
PORT 3306,
SOCKET '',
OWNER 'root');
```

You could now instead specify the server instead of the full URL connection string

```
connect="server_one"
```

## How does FederatedX differ from the old Federated Engine?

FederatedX from a user point of view is the same for the most part. What is different with FederatedX and Federated is the following:

- Rewrite of the main Federated source code from one single ha\_federated.cc file into three main abstracted components:
  - ha\_federatedx.cc - Core implementation of FederatedX
  - federated\_io.cc - Parent connection class to be over-written by derived classes for each RDBMS/client lib
  - federated\_io\_<driver>.cc - derived federated\_io class for a given RDBMS
  - federated\_txn.cc - New support for using transactional engines on the foreign server using a connection poll
- Various bugs fixed (need to look at opened bugs for Federated)

## Where can I get FederatedX

FederatedX is part of [MariaDB 5.1](#) and later. MariaDB merged with the latest FederatedX when there is a need to get a bug fixed. You can get the latest code/follow/participate in the project from the [FederatedX home page](#).

## What are the plans for FederatedX?

- Support for other RDBMS vendors using ODBC
- Support for pushdown conditions
- Ability to limit result set sizes

## 4.3.9.2 Differences Between FederatedX and Federated

The main differences are:

# New features in FederatedX

- Transactions (beta feature)
- Supports partitions (alpha feature)
- New class structure which allows developers to write connection classes for other RDBMSs without having to modify base classes for FederatedX
- Actively developed!

## Different behavior

- FederatedX is statically compiled into MariaDB by default.
- When you create a table with FederatedX, the connection will be tested. The

CREATE  
will fail if MariaDB can't connect to the remote host or if the remote table doesn't exist.

## 4.3.10 MEMORY Storage Engine

### Contents

1. [Index Type](#)
2. [See Also](#)
3. [Example](#)

Contents of the MEMORY storage engine (previously known as HEAP) are stored in memory rather than on disk.

It is best-used for read-only caches of data from other tables, or for temporary work areas.

Since the data is stored in memory, it is highly vulnerable to power outages or hardware failure, and is unsuitable for permanent data storage. In fact, after a server restart,

MEMORY

tables will be recreated (because the definition file is stored on disk), but they will be empty. It is possible to re-populate them with a query using the

--init-file

server startup option.

Variable-length types like

VARCHAR

can be used in MEMORY tables.

BLOB

or

TEXT

columns are not supported for MEMORY tables.

The maximum total size of MEMORY tables cannot exceed the

max\_heap\_table\_size

system server variable. When a table is created this value applies to that table, and when the server is restarted this value applies to existing tables. Changing this value has no effect on existing tables. However, executing a

ALTER TABLE ... ENGINE=MEMORY

statement applies the current value of

max\_heap\_table\_size

to the table. Also, it is possible to change the session value of

max\_heap\_table\_size

before creating a table, to make sure that tables created by other sessions are not affected.

The

MAX\_ROWS

table option provides a hint about the number of rows you plan to store in them. This is not a hard limit that cannot be exceeded, and does not allow to exceed

max\_heap\_table\_size

. The storage engine uses max\_heap\_table\_size and MAX\_ROWS to calculate the maximum memory that could be allocated for the table.

When rows are deleted, space is not automatically freed. The only way to free space without dropping the table is using

ALTER TABLE tbl\_name ENGINE = MEMORY

TRUNCATE TABLE

frees the memory too.

## Index Type

The MEMORY storage engine permits indexes to be either B-tree or Hash. Hash is the default type for MEMORY. See [Storage Engine index types](#) for more on their characteristics.

A MEMORY table can have up to 64 indexes, 16 columns for each index and a maximum key length of 3072 bytes.

## See Also

- [Performance of MEMORY tables](#)

## Example

The following example shows how to create a

MEMORY  
table with a given maximum size, as described above.

```
SET max_heap_table_size = 1024*516;  
  
CREATE TABLE t (a VARCHAR(10), b INT) ENGINE = MEMORY;  
  
SET max_heap_table_size = @@max_heap_table_size;
```

## 4.3.11 MERGE

### Contents

1. [Description](#)
2. [Examples](#)

## Description

The MERGE storage engine, also known as the MRG\_MyISAM engine, is a collection of identical MyISAM tables that can be used as one. "Identical" means that all tables have identical column and index information. You cannot merge MyISAM tables in which the columns are listed in a different order, do not have exactly the same columns, or have the indexes in different order. However, any or all of the MyISAM tables can be compressed with [myisampack](#). Column names and indexes names can be different, as long as data types and NULL/NOT NULL clauses are the same. Differences in table options such as AVG\_ROW\_LENGTH, MAX\_ROWS, or PACK\_KEYS do not matter.

Each index in a MERGE table must match an index in underlying MyISAM tables, but the opposite is not true. Also, a MERGE table cannot have a PRIMARY KEY or UNIQUE indexes, because it cannot enforce uniqueness over all underlying tables.

The following options are meaningful for MERGE tables:

- UNION  
. This option specifies the list of the underlying MyISAM tables. The list is enclosed between parentheses and separated with commas.
- INSERT\_METHOD  
. This option specifies whether, and how, INSERTs are allowed for the table. Allowed values are:  
NO  
(INSERTs are not allowed),  
FIRST  
(new rows will be written into the first table specified in the  
UNION  
list),  
LAST  
(new rows will be written into the last table specified in the  
UNION  
list). The default value is  
NO

If you define a MERGE table with a definition which is different from the underlying MyISAM tables, or one of the underlying tables is not MyISAM, the CREATE TABLE statement will not return any error. But any statement which involves the table will produce an error like the following:

```
ERROR 1168 (HY000): Unable to open underlying table which is differently defined  
or of non-MyISAM type or doesn't exist
```

CHECK TABLE

will show more information about the problem.

The error is also produced if the table is properly defined, but an underlying table's definition changes at some point in time.

If you try to insert a new row into a MERGE table with INSERT\_METHOD=NO, you will get an error like the following:

```
ERROR 1036 (HY000): Table 'tbl_name' is read only
```

It is possible to build a MERGE table on MyISAM tables which have one or more [virtual columns](#). MERGE itself does not support virtual columns, thus such columns will be seen as regular columns. The data types and sizes will still need to be identical, and they cannot be NOT NULL.

## Examples

```
CREATE TABLE t1 (
    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    message CHAR(20) ENGINE=MyISAM;

CREATE TABLE t2 (
    a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    message CHAR(20) ENGINE=MyISAM;

INSERT INTO t1 (message) VALUES ('Testing'),('table'),('t1');

INSERT INTO t2 (message) VALUES ('Testing'),('table'),('t2');

CREATE TABLE total (
    a INT NOT NULL AUTO_INCREMENT,
    message CHAR(20), INDEX(a)
    ENGINE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;

SELECT * FROM total;
+-----+
| a | message |
+-----+
| 1 | Testing |
| 2 | table   |
| 3 | t1      |
| 1 | Testing |
| 2 | table   |
| 3 | t2      |
+-----+
```

In the following example, we'll create three MyISAM tables, and then a MERGE table on them. However, one of them uses a different data type for the column b, so a SELECT will produce an error:

```
CREATE TABLE t1 (
    a INT,
    b INT
) ENGINE = MyISAM;

CREATE TABLE t2 (
    a INT,
    b INT
) ENGINE = MyISAM;

CREATE TABLE t3 (
    a INT,
    b TINYINT
) ENGINE = MyISAM;

CREATE TABLE t_mrg (
    a INT,
    b INT
) ENGINE = MERGE,UNION=(t1,t2,t3);

SELECT * FROM t_mrg;
ERROR 1168 (HY000): Unable to open underlying table which is differently defined
or of non-MyISAM type or doesn't exist
```

To find out what's wrong, we'll use a CHECK TABLE:

CHECK TABLE t_mrg;			
Table	Op	Msg_type	Msg_text
test.t_mrg	check	Error	Table 'test.t3' is differently defined or of non-MyISAM type or doesn't exist
test.t_mrg	check	Error	Unable to open underlying table which is differently defined or of non-MyISAM type or doesn't
test.t_mrg	check	error	Corrupt

Now, we know that the problem is in

t3  
's definition.

## 4.3.12 Mroonga

### 4.3.12.1 About Mroonga

#### Contents

1. [How to Install](#)
2. [Limitations](#)
3. [Available Character Sets](#)
4. [More Information](#)

Mroonga Version	Introduced	Maturity
7.07	<a href="#">MariaDB 10.2.11</a> , <a href="#">MariaDB 10.1.29</a>	Stable
5.04	<a href="#">MariaDB 10.1.6</a>	Stable
5.02	<a href="#">MariaDB 10.0.18</a> , <a href="#">MariaDB 10.1.5</a>	Stable
5.0	<a href="#">MariaDB 10.0.17</a>	Stable
4.06	<a href="#">MariaDB 10.0.15</a>	Stable

Mroonga is a full text search storage engine based on Groonga, which is an open-source CJK-ready (Chinese, Japanese, and Korean) fulltext search engine using column base. See <http://groonga.org> for more.

With Mroonga, you can have a CJK-ready full text search feature, and it is faster than the [MyISAM](#) and [InnoDB full text search](#) for both updating and searching.

Mroonga also supports Groonga's fast geolocation search by using MariaDB's geolocation SQL syntax.

Mroonga currently only supports Linux x86\_64 (Intel64/AMD64).

## How to Install

Enable Mroonga with the following statement:

```
INSTALL SONAME 'ha_mroonga';
```

On Debian and Ubuntu mroonga engine will be installed with

```
sudo apt-get install mariadb-plugin-mroonga
```

See [Plugin overview](#) for details on installing and uninstalling plugins.

`SHOW ENGINES` can be used to check whether Mroonga is installed correctly:

```
SHOW ENGINES;
...
***** 8. row *****
  Engine: Mroonga
  Support: YES
  Comment: CJK-ready fulltext search, column store
Transactions: NO
      XA: NO
  Savepoints: NO
...
```

Once the plugin is installed, add a UDF (User-Defined Function) named "last\_insert\_grn\_id", that returns the record ID assigned by groonga in INSERT, by the following SQL.

```
mysql> CREATE FUNCTION last_insert_grn_id RETURNS INTEGER SONAME 'ha_mroonga.so';
```

## Limitations

- The maximum size of a single key is 4096 bytes.
- The maximum size of all keys is 4GB.
- The maximum number of records in a fulltext index is 268,435,455
- The maximum number of distinct terms in a fulltext index is 268,435,455
- The maximum size of a fulltext index is 256GB

Note that the maximum sizes are not hard limits, and may vary according to circumstance.

For more details, see <http://mroonga.org/docs/reference/limitations.html>.

## Available Character Sets

Mroonga supports a limited number of [character sets](#). These include:

- ASCII
- BINARY
- CP932
- EUCJPMS
- KOI8R
- LATIN1
- SJIS
- UJIS
- UTF8
- UTF8MB4

## More Information

Further documentation for Mroonga can be found at <http://mroonga.org/docs/>

### 4.3.12.2 Mroonga Overview

#### Contents

1. [Score](#)
2. [Parser](#)
  1. [Examples](#)
    1. [TokenBigram vs TokenBigramSplitSymbol](#)
    2. [TokenBigram vs TokenBigramSplitSymbolAlpha](#)

Once Mroonga has been installed (see [About Mroonga](#)), its basic usage is similar to that of a [regular fulltext index](#).

For example:

```
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy)) ENGINE=Mroonga;

INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
('There was a wicked witch'), ('Who ate everybody up');

SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('wicked');
+-----+
| copy          |
+-----+
| There was a wicked witch |
+-----+
```

# Score

Mroonga can also order by weighting. For example, first add another record:

```
INSERT INTO ft_mroonga(copy) VALUES ('She met a wicked, wicked witch');
```

Records can be returned by weighting, for example, the newly added record has two occurrences of the word 'wicked' and a higher weighting:

```
SELECT *, MATCH(copy) AGAINST('wicked') AS score FROM ft_mroonga
  WHERE MATCH(copy) AGAINST('wicked') ORDER BY score DESC;
+-----+-----+
| copy           | score |
+-----+-----+
| She met a wicked, wicked witch | 299594 |
| There was a wicked witch      | 149797 |
+-----+-----+
```

# Parser

Mroonga permits you to set a different parser for searching by specifying the parser in the

```
CREATE TABLE
  statement as a comment or, in older versions, changing the value of the mroonga\_default\_parser system variable.
```

For example:

```
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenDelimitNull"'
  ENGINE=Mroonga,;
```

or

```
SET GLOBAL mroonga_default_parser = 'TokenBigramSplitSymbol';
```

The following parser settings are available:

Setting	Description
off	No tokenizing is performed.
TokenBigram	Default value. Continuous alphabetical characters, numbers or symbols are treated as a token.
TokenBigramIgnoreBlank	Same as TokenBigram except that white spaces are ignored.
TokenBigramIgnoreBlankSplitSymbol	Same as TokenBigramSplitSymbol . except that white spaces are ignore.
TokenBigramIgnoreBlankSplitSymbolAlpha	Same as TokenBigramSplitSymbolAlpha except that white spaces are ignored.
TokenBigramIgnoreBlankSplitSymbolAlphaDigit	Same as TokenBigramSplitSymbolAlphaDigit except that white spaces are ignored.
TokenBigramSplitSymbol	Same as TokenBigram except that continuous symbols are not treated as a token, but tokenised in bigram.
TokenBigramSplitSymbolAlpha	Same as TokenBigram except that continuous alphabetical characters are not treated as a token, but tokenised in bigram.
TokenDelimit	Tokenises by splitting on white spaces.
TokenDelimitNull	Tokenises by splitting on null characters (\0).
TokenMecab	Tokenise using MeCab. Required Groonga to be built with MeCab support.
TokenTrigram	Tokenises in trigrams but continuous alphabetical characters, numbers or symbols are treated as a token.

TokenUnigram	Tokenises in unigrams but continuous alphabetical characters, numbers or symbols are treated as a token.
--------------	----------------------------------------------------------------------------------------------------------

## Examples

### TokenBigram vs TokenBigramSplitSymbol

```
TokenBigram
failing to match partial symbols which
TokenBigramSplitSymbol
matches, since
TokenBigramSplitSymbol
does not treat continuous symbols as a token.
```

```
DROP TABLE ft_mroonga;
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenBigram"'
ENGINE=Mroonga;
INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
('There was a wicked witch'),
('Who ate everybody up'),
('She met a wicked, wicked witch'),
('A really wicked, wicked witch!!?!');
SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('!?');
Empty set (0.00 sec)

DROP TABLE ft_mroonga;
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenBigramSplitSymbol"'
ENGINE=Mroonga;
INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
('There was a wicked witch'),
('Who ate everybody up'),
('She met a wicked, wicked witch'),
('A really wicked, wicked witch!!?!');
SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('!?');
+-----+
| copy           |
+-----+
| A really wicked, wicked witch!!?! |
+-----+
```

### TokenBigram vs TokenBigramSplitSymbolAlpha

```
DROP TABLE ft_mroonga;
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenBigram"'
ENGINE=Mroonga;
INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
('There was a wicked witch'),
('Who ate everybody up'),
('She met a wicked, wicked witch'),
('A really wicked, wicked witch!!?!');
SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('ick');
Empty set (0.00 sec)

DROP TABLE ft_mroonga;
CREATE TABLE ft_mroonga(copy TEXT,FULLTEXT(copy) COMMENT 'parser "TokenBigramSplitSymbolAlpha"'
ENGINE=Mroonga;
INSERT INTO ft_mroonga(copy) VALUES ('Once upon a time'),
('There was a wicked witch'),
('Who ate everybody up'),
('She met a wicked, wicked witch'),
('A really wicked, wicked witch!!?!');
SELECT * FROM ft_mroonga WHERE MATCH(copy) AGAINST('ick');
+-----+
| copy           |
+-----+
| There was a wicked witch      |
| She met a wicked, wicked witch |
| A really wicked, wicked witch!!?! |
+-----+
```

## 4.3.12.3 Mroonga Status Variables

### Contents

1. [Mroonga\\_count\\_skip](#)
2. [Mroonga\\_fast\\_order\\_limit](#)

This page documents status variables related to the [Mroonga storage engine](#). See [Server Status Variables](#) for a complete list of status variables that can be viewed with `SHOW STATUS`.

#### Mroonga\_count\_skip

- **Description:** Incremented each time the 'fast line count feature' is used. Can be used to check if the feature is working after enabling it.
- **Data Type:**

numeric

---

#### Mroonga\_fast\_order\_limit

- **Description:** Incremented each time the 'fast ORDER BY LIMIT feature' is used. Can be used to check if the feature is working after enabling it.
- **Data Type:**

numeric

---

## 4.3.12.4 Mroonga System Variables

### Contents

1. [mroonga\\_action\\_on\\_fulltext\\_query\\_error](#)
2. [mroonga\\_boolean\\_mode\\_syntax\\_flags](#)
3. [mroonga\\_database\\_path\\_prefix](#)
4. [mroonga\\_default\\_parser](#)
5. [mroonga\\_default\\_tokenizer](#)
6. [mroonga\\_default\\_wrapper\\_engine](#)
7. [mroonga\\_dry\\_write](#)
8. [mroonga\\_enable\\_operations\\_recording](#)
9. [mroonga\\_enable\\_optimization](#)
10. [mroonga\\_libgroonga\\_embedded](#)
11. [mroonga\\_libgroonga\\_support\\_lz4](#)
12. [mroonga\\_libgroonga\\_support\\_zlib](#)
13. [mroonga\\_libgroonga\\_support\\_zstd](#)
14. [mroonga\\_libgroonga\\_version](#)
15. [mroonga\\_lock\\_timeout](#)
16. [mroonga\\_log\\_file](#)
17. [mroonga\\_log\\_level](#)
18. [mroonga\\_match\\_escalation\\_threshold](#)
19. [mroonga\\_max\\_n\\_records\\_for\\_estimate](#)
20. [mroonga\\_query\\_log\\_file](#)
21. [mroonga\\_vector\\_column\\_delimiter](#)
22. [mroonga\\_version](#)

This page documents system variables related to the [Mroonga storage engine](#). See [Server System Variables](#) for a complete list of system variables and instructions on setting them.

See also the [Full list of MariaDB options, system and status variables](#).

#### mroonga\_action\_on\_fulltext\_query\_error

- **Description:** Action to take when encountering a Mroonga fulltext error.

◦

ERROR

: Report an error without logging.

◦

ERROR\_AND\_LOG

- : Report an error with logging (the default)
- - IGNORE**  
: No logging or reporting - the error is ignored.
  - IGNORE\_AND\_LOG**  
: Log the error without reporting it.

- **Commandline:**

```
--mroonga-action-on-fulltext-query-error=value
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

ERROR\_AND\_LOG

---

## mroonga\_boolean\_mode\_syntax\_flags

- **Description:** Flags to customize syntax in BOOLEAN MODE searches. Available flags:

- - DEFAULT**  
:(=SYNTAX\_QUERY,ALLOW.LEADING.NOT)
  - ALLOW\_COLUMN**  
: Allows  
COLUMN:...  
syntax in query syntax, an incompatible change to the regular BOOLEAN MODE syntax. Permits multiple indexes in one MATCH () AGAINST ()  
. Can be used in other operations besides full-text search, such as equal, and prefix search. See [Groonga query syntax](#) for more details.
  - ALLOW.LEADING.NOT**  
Permits using the  
NOTINCLUDED\_KEYWORD  
syntax in the query syntax.
  - ALLOW\_UPDATE**  
: Permits updating values with the  
COLUMN:=NEW\_VALUE  
syntax in the query syntax.
  - SYNTAX\_QUERY**  
: Mroonga will use Groonga's query syntax, compatible with MariaDB's BOOLEAN MODE syntax. Unless  
SYNTAX\_SCRIPT  
is specified, this mode is always in use.
  - SYNTAX\_SCRIPT**  
: Mroonga will use Groonga's script syntax, a JavaScript-like syntax. If both  
SYNTAX\_QUERY  
and  
SYNTAX\_SCRIPT  
are specified,  
SYNTAX\_SCRIPT  
will take precedence..

- **Commandline:**

```
--mroonga-boolean-mode-syntax-flags=value
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

DEFAULT

## `mroonga_database_path_prefix`

- **Description:** The database path prefix.
  - **Commandline:**  
`--mroonga-database-path-prefix=value`
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
`string`
  - **Default Value:** (Empty)
- 

## `mroonga_default_parser`

- **Description:** The fulltext default parser, for example  
`TokenBigramSplitSymbolAlphaDigit`  
or  
`TokenBigram`  
(the default). See the list of options at [Mroonga Overview:Parser](#). Deprecated since Mroonga 5.0.4, use `mroonga_default_tokenizer` instead.
  - **Commandline:**  
`--mroonga-default-parser=value`
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
`string`
  - **Default Value:**  
`TokenBigram`
  - **Deprecated:** [MariaDB 10.1.6](#), Mroonga 5.0.4
- 

## `mroonga_default_tokenizer`

- **Description:** The fulltext default parser, for example  
`TokenBigramSplitSymbolAlphaDigit`  
or  
`TokenBigram`  
(the default). See the list of options at [Mroonga Overview:Parser](#).
  - **Commandline:**  
`--mroonga-default-tokenizer=value`
  - **Scope:** Global, Session
  - **Dynamic:** Yes
  - **Data Type:**  
`string`
  - **Default Value:**  
`TokenBigram`
  - **Introduced:** [MariaDB 10.1.6](#), Mroonga 5.0.4
- 

## `mroonga_default_wrapper_engine`

- **Description:** The default engine for wrapper mode.
- **Commandline:**  
`--mroonga-default-wrapper-engine=value`
- **Scope:** Global

- **Dynamic:** No
  - **Data Type:**  
string
  - **Default Value:** (Empty)
- 

mroonga\_dry\_write

- **Description:** If set to
    - on
    - , (
    - offis default), data is not actually written to the Groonga database. Only really useful to change for benchmarking.
  - **Commandline:**  
--mroonga-dry-write[={0|1}]
  - **Scope:** Global, Session
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value:**  
off
- 

mroonga\_enable\_operations\_recording

- **Description:** Whether recording operations for recovery to the Groonga database is enabled (default) or not. Requires reopening the database with [FLUSH TABLES](#) after changing the variable.
  - **Commandline:**  
--mroonga-enable-operations-recording={0|1}
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value:**  
ON
  - **Introduced:** [MariaDB 10.2.11](#) , [MariaDB 10.1.29](#)
- 

mroonga\_enable\_optimization

- **Description:** If set to
    - on
    - (the default), optimization is enabled. Only really useful to change for benchmarking.
  - **Commandline:**  
--mroonga-enable-optimization={0|1}
  - **Scope:** Global, Session
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value:**  
on
- 

mroonga\_libgroonga\_embedded

- **Description:** Whether libgroonga is embedded or not.

- **Commandline:** None

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

- **Introduced:** [MariaDB 10.1.6](#)
- 

#### mroonga\_libgroonga\_support\_lz4

- **Description:** Whether libgroonga supports lz4 or not.

- **Commandline:** None

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** [MariaDB 10.0.17](#)
- 

#### mroonga\_libgroonga\_support\_zlib

- **Description:** Whether libgroonga supports zlib or not.

- **Commandline:** None

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

---

#### mroonga\_libgroonga\_support\_zstd

- **Description:** Whether libgroonga supports Zstandard or not.

- **Commandline:** None

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** [MariaDB 10.2.11](#) , [MariaDB 10.1.29](#)
- 

#### mroonga\_libgroonga\_version

- **Description:** Groonga library version.

- **Commandline:** None

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

---

### mroonga\_lock\_timeout

- **Description:** Lock timeout used in Groonga.

- **Commandline:**

<<

code

>>

--mroonga-lock-timeout=#</code>>

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

900000

- **Range:**

-1

to

2147483647

---

### mroonga\_log\_file

- **Description:** Name and path of the Mroonga log file.

- **Commandline:**

--mroonga-log-file=value

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:**

groonga.log

---

### mroonga\_log\_level

- **Description:** Mroonga log file output level, which determines what is logged. Valid levels include:

○

NONE

No output.

○

EMERG

: Only emergency error messages, such as database corruption.

○

ALERT

: Alert messages, such as internal errors.

○

CRIT

: Critical error messages, such as deadlocks.

- - ERROR  
: Errors, such as API errors.
  - - WARNING  
: Warnings, such as invalid arguments.
  - - NOTICE  
: Notices, such as a change in configuration or a status change.
  - - INFO  
: Information messages, such as file system operations.
  - - DEBUG  
: Debug messages, suggested for developers or testers.
  - - DUMP  
: Dump messages.

- **Commandline:**

```
--mroonga-log-level=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

NOTICE

---

## mroonga\_match\_escalation\_threshold

- **Description:** The threshold to determine whether the match method is escalated.

-1  
means never escalate.

- **Commandline:**

```
--mroonga-match-escalation-threshold=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

-1  
to  
9223372036854775807

---

## mroonga\_max\_n\_records\_for\_estimate

- **Description:** The max number of records to estimate the number of matched records

- **Commandline:**

```
--mroonga-max-n-records-for-estimate=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1000

- **Range:**

```
-1  
to  
2147483647
```

- **Introduced:** [MariaDB 10.0.18](#) , [Mroonga 5.0.2](#)
- 

### mroonga\_query\_log\_file

- **Description:** Query log file for Mroonga.

- **Commandline:**

```
--mroonga-query-log-file=filename
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
string
```

- **Default Value:** (Empty string)

- **Introduced:** [MariaDB 10.2.11](#) , [MariaDB 10.1.29](#)
- 

### mroonga\_vector\_column\_delimiter

- **Description:** Delimiter to use when outputting a vector column. The default is a white space.

- **Commandline:**

```
--mroonga-vector-column-delimiter=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
string
```

- **Default Value:**

```
(white space)
```

---

### mroonga\_version

- **Description:** Mroonga version

- **Commandline:** None

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
string
```

---

## 4.3.12.5

### 4.3.12.5.1 Creating Mroonga User-Defined Functions

The [Mroonga storage engine](#) includes a number of [user-defined functions](#) that need to be created before they can be used. If these are not created already during Mroonga setup, you will need to do so yourself. The full list of available functions and the statements to create them are found in

```
share/mroonga/install.sql  
, for example, as of Mroonga 7.07 ( MariaDB 10.2.11 and MariaDB 10.1.29 ) running on Linux:
```

```

DROP FUNCTION IF EXISTS last_insert_grn_id;
CREATE FUNCTION last_insert_grn_id RETURNS INTEGER
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_snippet;
CREATE FUNCTION mroonga_snippet RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_command;
CREATE FUNCTION mroonga_command RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_escape;
CREATE FUNCTION mroonga_escape RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_snippet_html;
CREATE FUNCTION mroonga_snippet_html RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_normalize;
CREATE FUNCTION mroonga_normalize RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_highlight_html;
CREATE FUNCTION mroonga_highlight_html RETURNS STRING
  SONAME 'ha_mroonga.so';

DROP FUNCTION IF EXISTS mroonga_query_expand;
CREATE FUNCTION mroonga_query_expand RETURNS STRING
  SONAME 'ha_mroonga.so';

```

## 4.3.12.5.2 last\_insert\_grn\_id

### Syntax

```
last_insert_grn_id()
```

### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

### Description

`last_insert_grn_id` is a [user-defined function](#) (UDF) included with the [Mroonga storage engine](#). It returns the unique Groonga id of the last-inserted record. See [Creating Mroonga User-Defined Functions](#) for details on creating this UDF if required.

### Examples

```

SELECT last_insert_grn_id();
+-----+
| last_insert_grn_id() |
+-----+
|            3 |
+-----+

```

### See Also

- [Creating Mroonga User-Defined Functions](#)

## 4.3.12.5.3 mroonga\_command

## Contents

- 1. [Syntax](#)
- 2. [Description](#)
- 3. [Example](#)
- 4. [See Also](#)

## Syntax

```
mroonga_command (command)
```

## Description

`mroonga_command` is a [user-defined function](#) (UDF) included with the [Mroonga storage engine](#). It passes a command to Groonga for execution. See [Creating Mroonga User-Defined Functions](#) for details on creating this UDF if required.

- - command
    - string, required parameter specifying the command to pass that will be executed by Groonga. See [the Groonga reference](#) for a list of commands.

Returns the result of the Groonga command.

## Example

```
SELECT mroonga_command('status');
+-----+
| mroonga_command('status') |
+-----+
| {"alloc_count":593,"starttime":1512022368,"start_time":1512022368,"uptime":13510,"version":"7.0.7","n_queries":0,"cache_hit_ra
|
```

## See Also

- [Creating Mroonga User-Defined Functions](#)

## 4.3.12.5.4 mroonga\_escape

### Contents

- 1. [Syntax](#)
- 2. [Description](#)
- 3. [Example](#)
- 4. [See Also](#)

## Syntax

```
mroonga_escape (string [,special_characters])
```

- - string
    - required parameter specifying the text you want to escape
- - special\_characters
    - optional parameter specifying the characters to escape

## Description

`mroonga_escape` is a [user-defined function](#) (UDF) included with the [Mroonga storage engine](#), used for escaping a string. See [Creating Mroonga User-Defined Functions](#) for details on creating this UDF if required.

If no

    special\_characters  
parameter is provided, by default

+-<>\*():  
are escaped.

Returns the escaped string.

## Example

```
SELECT mroonga_escape("+-<>~*()\"\\:");
'\\+\\-\\<\\>\\~\\*\\(\\)\\\"\\:'
```

## See Also

- [Creating Mroonga User-Defined Functions](#)

### 4.3.12.5.5 mroonga\_highlight\_html

## Syntax

```
mroonga_highlight_html(text[, query AS query])
mroonga_highlight_html(text[, keyword1, ..., keywordN])
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

## Description

`mroonga_highlight_html`

is a [user-defined function](#) (UDF) included with the [Mroonga storage engine](#). It highlights the specified keywords in the target text. See [Creating Mroonga User-Defined Functions](#) for details on creating this UDF if required.

The optional parameter can either be one or more *keywords*, or a Groonga *query*.

The function highlights the specified keywords in the target text by surrounding each keyword with

```
<span class="keyword">...</span>
, and escaping special HTML characters such as
<
and
>
```

Returns highlighted HTML.

## Examples

```
SELECT mroonga_highlight_html('<p>MariaDB includes the Mroonga storage engine</p>.')
 AS highlighted_html;
+-----+
| highlighted_html
+-----+
| &lt;p&gt;MariaDB includes the Mroonga storage engine&lt;/p&gt;.
+-----+
```

Highlighting the words

```
MariaDB
and
Mroonga
in a given text:
```

```
SELECT mroonga_highlight_html('MariaDB includes the Mroonga storage engine.', 'MariaDB', 'Mroonga')
AS highlighted_html;
+-----+
| highlighted_html
+-----+
| <span class="keyword">MariaDB</span> includes the <span class="keyword">Mroonga</span> storage engine. |
+-----+
```

The same outcome, formulated as a Groonga query:

```
SELECT mroonga_highlight_html('MariaDB includes the Mroonga storage engine.', 'MariaDB OR Mroonga'
AS query) AS highlighted_text;
+-----+
| highlighted_text
+-----+
| <span class="keyword">MariaDB</span> includes the <span class="keyword">Mroonga</span> storage engine. |
+-----+
```

## See Also

- [Creating Mroonga User-Defined Functions](#)

### 4.3.12.5.6 mroonga\_normalize

#### Syntax

```
mroonga_normalize(string[, normalizer_name])
```

#### Contents

1. [Syntax](#)
2. [Description](#)
3. [Examples](#)
4. [See Also](#)

#### Description

`mroonga_normalize`

is a [user-defined function \(UDF\)](#) included with the [Mroonga storage engine](#). It uses Groonga's normalizer to normalize text. See [Creating Mroonga User-Defined Functions](#) for details on creating this UDF if required.

Given a string, returns the normalized text.

See the [Groonga Normalizer Reference](#) for details on the Groonga normalizers. The default if no normalizer is provided is

`NormalizerAuto`

#### Examples

```
SELECT mroonga_normalize("ABいり");
+-----+
| mroonga_normalize("ABいり")   |
+-----+
| abiirittorl                  |
+-----+
```

#### See Also

- [Creating Mroonga User-Defined Functions](#)

### 4.3.12.5.7 mroonga\_snippet

#### Syntax

```
mroonga_snippet document,
    max_length,
    max_count,
    encoding,
    skip_leading_spaces,
    html_escape,
    snippet_prefix,
    snippet_suffix,
    word1, word1_prefix, word1_suffix
    ...
    [wordN wordN_prefix wordN_suffix]
```

## Contents

1. [Syntax](#)
2. [Description](#)
3. [Example](#)
4. [See Also](#)

## Description

`mroonga_snippet`

is a [user-defined function](#) (UDF) included with the [Mroonga storage engine](#). It provides a keyword with surrounding text, or the keyword in context. See [Creating Mroonga User-Defined Functions](#) for details on creating this UDF if required.

The required parameters include:

- - document
    - Column name or string value.
  - max\_length
    - Maximum length of the snippet, in bytes.
  - max\_count
    - Maximum snippet elements (N word).
  - encoding
    - Encoding of the document, for example
    - cp932\_japanese\_ci
  - skip\_leading\_spaces
    - 
    - 1
      - to skip leading spaces,
    - 0
      - to not skip.
  - html\_escape
    - =
    - 1
      - to enable HTML escape,
    - 0
      - to disable.
  - prefix
    - Snippet start text.
  - suffix
    - Snippet end text.

The optional parameters include:

- - wordN
    - A word.
  - wordN\_prefix
    - wordN start text.
  - wordN\_suffix
    - wordN end text

It can be used in both storage and wrapper mode.

Returns the snippet string.

## Example

### See Also

- [Creating Mroonga User-Defined Functions](#)

## 4.3.12.5.8 mroonga\_snippet\_html

### Description

`mroonga_snippet_html`

is a [user-defined function](#) (UDF) included with the [Mroonga storage engine](#). It provides a keyword with surrounding text, or the keyword in context. It is still considered experimental. See [Creating Mroonga User-Defined Functions](#) for details on creating this UDF if required.

### See Also

- [Creating Mroonga User-Defined Functions](#)
- [mroonga\\_snippet](#)

## 4.3.12.6 Information Schema MROONGA\_STATS Table

The [Information Schema](#)

MROONGA\_STATS

table only exists if the [Mroonga](#) storage engine is installed, and contains information about its activities.

Column	Description
VERSION	Mroonga version.
rows_written	Number of rows written into Mroonga tables.
rows_read	Number of rows read from all Mroonga tables.

This table always contains 1 row.

## 4.3.13 MyISAM

MyISAM was the default [storage engine](#) from MySQL 3.23 until it was replaced by [InnoDB](#) in MariaDB and MySQL 5.5. It's a light, non-transactional engine with great performance, is easy to copy between systems and has a small data footprint.

You're encouraged to rather use the [Aria](#) storage engine for new applications, which has even better performance and the goal of being crash-safe.

Until [MariaDB 10.4](#), [system tables](#) used the MyISAM storage engine.



### MyISAM Overview

[Light, non-transactional storage engine](#)



### MyISAM System Variables

[MyISAM system variables](#).



### MyISAM Storage Formats

[The MyISAM storage engine supports three different table storage formats](#)



### MyISAM Clients and Utilities

[Clients and utilities for working with MyISAM tables](#)



## MyISAM Index Storage Space

Regular MyISAM tables make use of B-tree indexes



## MyISAM Log

Records all changes to MyISAM tables



## Concurrent Inserts

Under some circumstances, MyISAM allows INSERTs and SELECTs to be executed concurrently.



## Segmented Key Cache

Collection of structures for regular MyISAM key caches

There are 1 related questions .

### 4.3.13.1 MyISAM Overview

The MyISAM storage engine was the default storage engine from MySQL 3.23 until it was replaced by InnoDB in MariaDB and MySQL 5.5. Historically, MyISAM is a replacement for the older ISAM engine, removed in MySQL 4.1.

It's a light, non-transactional engine with great performance, is easy to copy between systems and has a small data footprint.

You're encouraged to rather use the Aria storage engine for new applications, which has even better performance in most cases and the goal of being crash-safe.

A MyISAM table is stored in three files on disk. There's a table definition file with an extension of

```
.frm  
, a data file with the extension  
.MYD  
, and an index file with the extension  
.MYI  
.
```

### MyISAM features

- Does not support transactions .
- Does not support foreign keys.
- Supports FULLTEXT indexes .
- Supports GIS data types.
- Storage limit of 256TB.
- Maximum of 64 indexes per table.
- Maximum of 32 columns per index.
- Maximum index length of 1000 bytes.
- Limit of  $(2^{32})^2$  (1.844E+19) rows per table.
- Supports large files up to 63-bits in length where the underlying system supports this.
- All data is stored with the low byte first, so all files will still work if copied to other systems or other machines.
- The data file and the index file can be placed on different devices to improve speed.
- Supports table locking, not row locking.
- Supports a key buffer that is segmented in MariaDB.
- Supports concurrent inserts .
- Supports fixed length, dynamic and compressed formats - see MyISAM Storage Formats .
- Numeric index values are stored with the high byte first, which enables more efficient index compression.
- Data values are stored with the low byte first, making it mostly machine and operating system independent. The only exceptions are if a machine doesn't use two's-complement signed integers and the IEEE floating-point format.
- Can be copied between databases or systems with normal system tools, as long as the files are not open on either system. Use FLUSH\_TABLES to ensure files are not in use.
- There are a number of tools for working with MyISAM tables. These include:
  - mysqlcheck for checking or repairing
  - myisamchk for checking or repairing
  - myisampack for compressing
- It is possible to build a MERGE table on the top of one or more MyISAM tables.

### 4.3.13.2 MyISAM System Variables

## Contents

1. [key\\_buffer\\_size](#)
2. [key\\_cache\\_age\\_threshold](#)
3. [key\\_cache\\_block\\_size](#)
4. [key\\_cache\\_division\\_limit](#)
5. [key\\_cache\\_file\\_hash\\_size](#)
6. [key\\_cache\\_segments](#)
7. [myisam\\_block\\_size](#)
8. [myisam\\_data\\_pointer\\_size](#)
9. [myisam\\_max\\_extra\\_sort\\_file\\_size](#)
10. [myisam\\_max\\_sort\\_file\\_size](#)
11. [myisam\\_mmap\\_size](#)
12. [myisam\\_recover\\_options](#)
13. [myisam\\_repair\\_threads](#)
14. [myisam\\_sort\\_buffer\\_size](#)
15. [myisam\\_stats\\_method](#)
16. [myisam\\_use\\_mmap](#)

This page documents system variables related to the [MyISAM](#) storage engine. For options, see [MyISAM Options](#).

See [Server System Variables](#) for a complete list of system variables and instructions on setting them.

See also the [Full list of MariaDB options, system and status variables](#).

### key\_buffer\_size

- **Description:** Size of the buffer for the index blocks used by MyISAM tables and shared for all threads. See [Optimizing key\\_buffer\\_size](#) for more on selecting the best value.

- **Commandline:**

```
--key-buffer-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

134217728

- **Range:**

8

upwards (upper limit determined by operating system per process limit)

### key\_cache\_age\_threshold

- **Description:** The lower the setting, the more quickly buffers move from the hot key cache sublist to the warm sublist.

- **Commandline:**

```
--key-cache-age-threshold=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

300

- **Range:**

100

to

4294967295

### key\_cache\_block\_size

- **Description:** MyISAM key cache block size in bytes .

- **Commandline:**

```
--key-cache-block-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1024

- **Range:**

512

to

16384

---

## key\_cache\_division\_limit

- **Description:** Percentage to use for the warm key cache buffer list (the remainder is allocated between the hot and cold caches).

- **Commandline:**

```
--key-cache-division-limit=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

100

- **Range:**

1

to

100

---

## key\_cache\_file\_hash\_size

- **Description:** Number of hash buckets for open and changed files. If you have many MyISAM files open you should increase this for faster flushing of changes. A good value is probably 1/10th of the number of possible open MyISAM files.

- **Commandline:**

```
--key-cache-file-hash-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

512

- **Range:**

128

to

16384

- **Introduced:** MariaDB 10.0.13
- 

## key\_cache\_segments

- **Description:** The number of segments in a key cache. See [Segmented Key Cache](#).

- **Commandline:**

```
--key-cache-segments=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Type:** numeric

- **Default Value:**

0

*(non-segmented)*

- **Range:**

0

to

64

### `myisam_block_size`

- **Description:** Block size to be used for MyISAM index pages.

- **Commandline:**

```
--myisam-block-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

1024

### `myisam_data_pointer_size`

- **Description:** Size in bytes of the default pointer, used in a [MyISAM CREATE TABLE](#) with no MAX\_ROWS option.

- **Commandline:**

```
--myisam-data-pointer-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

6

- **Range:**

2

to

7

### `myisam_max_extra_sort_file_size`

- **Description:** Removed in MySQL 5.0.6, was used as a way to force long character keys in large tables to use the key cache method.

- **Removed:** MySQL 5.0.6

### `myisam_max_sort_file_size`

- **Description:** Maximum size in bytes of the temporary file used while recreating a MyISAM index. If the this size is exceeded, the slower process of

using the key cache is done instead.

- **Commandline:**

```
--myisam-max-sort-file-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value - 32 bit:**

```
    2147483648
```

- **Default Value - 64 bit:**

```
    9223372036854775807
```

---

## myisam\_mmap\_size

- **Description:** Maximum memory in bytes that can be used for memory mapping compressed MyISAM files. Too high a value may result in swapping if there are many compressed MyISAM tables.

- **Commandline:**

```
--myisam-mmap-size=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value - 32 bit:**

```
    4294967295
```

- **Default Value - 64 bit:**

```
    18446744073709547520
```

- **Range - 32-bit:**

```
    7  
    to  
    4294967295
```

- **Range - 64-bit:**

```
    7  
    to  
    18446744073709547520
```

---

## myisam\_recover\_options

- **Description:** MyISAM recovery mode. Multiple options can be selected, comma-delimited. Using no argument is equivalent to specifying

```
    DEFAULT
```

```
, while specifying "" is equivalent to
```

```
    OFF
```

If enabled each time the server opens a MyISAM table, it checks whether it has been marked as crashed, or wasn't closed properly. If so, mysqld will run a check and then attempt to repair the table, writing to the error log beforehand.

- **OFF** : No recovery.

- **BACKUP** : If the data file is changed while recovering, saves a backup of the .MYD data file. t.MYD will be saved as t.MYD-datetime.BAK.

- **BACKUP\_ALL** : Same as

```
    BACKUP
```

```
        but also backs up the .MYI index file. t.MYI will be saved as t.MYI-datetime.BAK.
```

- **DEFAULT** : Recovers without backing up, forcing, or quick checking.

- **FORCE** : Runs the recovery even if it determines that more than one row from the data file will be lost.

- **QUICK** : Does not check rows in the table if there are no delete blocks.

- **Commandline:**

```
--myisam-recover-options[=name]
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**  
enumeration

- **Default Value:**

- BACKUP, QUICK  
( $\geq$  MariaDB 10.2.4 )
- DEFAULT  
( $\leq$  MariaDB 10.2.3 )
- OFF

- **Valid Values:**

- OFF
- ,
- DEFAULT
- ,
- BACKUP
- ,
- BACKUP\_ALL
- ,
- FORCE
- or
- QUICK

---

## myisam\_repair\_threads

- **Description:** If set to more than

- 1 , the default, MyISAM table indexes each have their own thread during repair and sorting. Increasing from the default will usually result in faster repair, but will use more CPU and memory.

- **Commandline:**

- myisam-repair-threads=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

- numeric

- **Default Value:**

- 1

- **Range - 32-bit:**

- 1
- to
- 4294967295

- **Range - 64-bit:**

- 1
- to
- 18446744073709547520

---

## myisam\_sort\_buffer\_size

- **Description:** Size in bytes of the buffer allocated when creating or sorting indexes on a MyISAM table.

- **Commandline:**

- myisam-sort-buffer-size=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

- numeric

- **Default Value:**

```
134217720  
(128MB),  
8388608  
(8MB - before MariaDB 10.0.3 )
```

- **Range:**

```
4096  
to  
18446744073709547520
```

---

### myisam\_stats\_method

- **Description:** Determines how NULLs are treated for MyISAM index statistics purposes. If set to

```
nulls_equal
```

, the default, all NULL index values are treated as a single group. This is usually fine, but if you have large numbers of NULLs the average group size is slanted higher, and the optimizer may miss using the index for ref accesses when it would be useful. If set to  
`nulls_unequal`

, the opposite approach is taken, with each NULL forming its own group of one. Conversely, the average group size is slanted lower, and the optimizer may use the index for ref accesses when not suitable. Setting to

```
nulls_ignored
```

ignores NULLs altogether from index group calculations. See also [Index Statistics](#) , [aria\\_stats\\_method](#) , [innodb\\_stats\\_method](#) .

- **Commandline:**

```
--myisam-stats-method=name
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
enumeration
```

- **Default Value:**

```
nulls_equal
```

- **Valid Values:**

```
nulls_equal
```

```
,
```

```
nulls_unequal
```

```
,
```

```
nulls_ignored
```

### myisam\_use\_mmap

- **Description:** If set to

```
1
```

(0 is default), memory mapping will be used to reading and writing MyISAM tables.

- **Commandline:**

```
--myisam-use-mmap
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
boolean
```

- **Default Value:**

```
OFF
```

## 4.3.13.3 MyISAM Storage Formats

## Contents

1. [Fixed-length](#)
2. [Dynamic](#)
3. [Compressed](#)
4. [See Also](#)

The [MyISAM](#) storage engine supports three different table storage formats.

These are FIXED, DYNAMIC and COMPRESSED. FIXED and DYNAMIC can be set with the ROW FORMAT option in the [CREATE TABLE](#) statement, or will be chosen automatically depending on the columns the table contains. COMPRESSED can only be set via the [myisampack](#) tool.

The [SHOW TABLE STATUS](#) statement can be used to see the storage format used by a table. Note that

COMPRESSED  
tables are reported as  
DYNAMIC  
in that context.

## Fixed-length

Fixed-length (or static) tables contain records of a fixed-length. Each column is the same length for all records, regardless of the actual contents. It is the default format if a table has no [BLOB](#), [TEXT](#), [VARCHAR](#) or [VARBINARY](#) fields, and no ROW FORMAT is provided. You can also specify a fixed table with ROW\_FORMAT=FIXED in the table definition.

Tables containing BLOB or TEXT fields cannot be FIXED, as by design these are both dynamic fields. However, no error or warning will be raised if you specify FIXED.

Fixed-length tables have a number of characteristics

- fast, since MariaDB will always know where a record begins
- easy to repair: [myisamchk](#) is always able to recover all rows, except for the last one if it is not entirely written
- easy to cache
- take up more space than dynamic or compressed tables, as the maximum amount of storage space will be allocated to each record.
- reconstructing after a crash is uncomplicated due to the fixed positions
- no fragmentation or need to re-organize, unless records have been deleted and you want to free the space up.

## Dynamic

Dynamic tables contain records of a variable length. It is the default format if a table has any BLOB, TEXT, VARCHAR or VARBINARY fields, and no ROW FORMAT is provided. You can also specify a DYNAMIC table with ROW\_FORMAT=DYNAMIC in the table definition. If the table contains BLOB or TEXT columns, its format is always DYNAMIC, and the ROW FORMAT option is ignored.

Dynamic tables have a number of characteristics

- Each row contains a header indicating the length of the row.
- Rows tend to become fragmented easily. UPDATING a record to be longer will likely ensure it is stored in different places on the disk. Use [OPTIMIZE TABLE](#) when the fragmentation is too high.
- All string columns with a length of four or more are dynamic.
- They require much less space than fixed-length tables.
- Restoring after a crash is more complicated than with FIXED tables. Some fragments may be lost.

If a DYNAMIC table has some frequently-accessed fixed-length columns, it could be a good idea to move them into a separate FIXED table to avoid fragmentation.

## Compressed

Compressed tables are a read-only format, created with the [myisampack](#) tool. This can be done while the server is running, but external lock must not be disabled. [myisamchk](#) is used to uncompress them.

Compressed tables have a number of characteristics:

- while the data is read-only, DDL statements such as [DROP TABLE](#) and [TRUNCATE TABLE](#) will still function.
- take much less space than fixed or dynamic tables. Each data has usually a 40-70% compression ratio
- rows are compressed separately, reducing access overhead.
- row headers will be from one to three bytes.
- rows can be compressed with different compression types, including
  - prefix space compression
  - suffix space compression
  - columns with small sets of values are converted to ENUM
  - numeric zeros are stored with only one bit
  - integer columns will be reduced to the smallest int type that can hold the contents

## See Also

- [Why we still need MyISAM \(for read-only tables\)](#) describes an important use case for MyISAM compressed tables.

### 4.3.13.4.1 myisamchk

#### Contents

1. [General Options](#)
2. [Checking Tables](#)
3. [Repairing Tables](#)
4. [Other Actions](#)
5. [Examples](#)
6. [See Also](#)

myisamchk is a commandline tool for checking, repairing and optimizing non-partitioned [MyISAM](#) tables.

myisamchk is run from the commandline as follows:

```
myisamchk [OPTIONS] tables[.MYI]
```

The full list of options are listed below. One or more MyISAM tables can be specified. MyISAM tables have an associated .MYI index file, and the table name can either be specified with or without the .MYI extension. Referencing it with the extension allows you to use wildcards, so it's possible to run myisamchk on *all* the MyISAM tables in the database with

\*.MYI

The path to the files must also be specified if they're not in the current directory.

myisamchk should not be run while anyone is accessing any of the affected tables. It is also best to make a backup before running.

With no options, myisamchk simply checks your table as the default operation.

The following options can be set while passed as commandline options to myisamchk, or set with a [myisamchk] section in your [my.cnf](#) file.

#### General Options

Option	Description
-H, --HELP	Display help and exit. Options are presented in a single list.
-?, --help	Display help and exit. Options are grouped by type of operation.
-debug=options, -# options	Write a debugging log. A typical debug_options string is 'd:t:o,file_name'. The default is 'd:t:o,/tmp/myisamchk.trace'. (Available in debug builds only)
-t path, --tmpdir=path	Path for temporary files. Multiple paths can be specified, separated by colon (:) on Unix and semicolon (;) on Windows. They will be used in a round-robin fashion. If not set, the TMPDIR environment variable is used.
-s, --silent	Only print errors. One can use two -s (-ss) to make myisamchk very silent.
-v, --verbose	Print more information. This can be used with --description and --check. Use many -v for more verbosity.
-V, --version	Print version and exit.
-w, --wait	If table is locked, wait instead of returning an error.
--print-defaults	Print the program argument list and exit.
--no-defaults	Don't read default options from any option file.
--defaults-file=filename	Only read default options from the given file <i>filename</i> , which can be the full path, or the path relative to the current directory.
--defaults-extra-file=filename	Read the file <i>filename</i> , which can be the full path, or the path relative to the current directory, after the global files are read.
--defaults-group-suffix=str	Also read groups with a suffix of <i>str</i> . For example, --defaults-group-suffix=x would read the groups [myisamchk] and [myisamchk_x]

The following variables can also be set by using `--var_name=value`, for example `--ft_min_word_len=5`

Variable	Default Value
decode_bits	9

ft_max_word_len	version-dependent
ft_min_word_len	4
ft_stopword_file	built-in list
key_buffer_size	1044480
key_cache_block_size	1024
myisam_block_size	1024
myisam_sort_buffer_size	134216704
myisam_sort_key_blocks	16
read_buffer_size	262136
sort_buffer_size	134216704
sort_key_blocks	16
stats_method	nulls Unequal
write_buffer_size	262136

## Checking Tables

If no option is provided, myisamchk will perform a check table. It is possible to check MyISAM tables without shutting down or restricting access to the server by using [CHECK TABLE](#) instead.

The following check options are available:

Option	Description
-c, --check	Check table for errors. This is the default operation if you specify no option that selects an operation type explicitly.
-e, --extend-check	Check the table VERY thoroughly. Only use this in extreme cases as it may be slow, and myisamchk should normally be able to find out if the table has errors even without this switch. Increasing the key_buffer_size can help speed the process up.
-F, --fast	Check only tables that haven't been closed properly.
-C, --check-only-changed	Check only tables that have changed since last check.
-f, --force	Restart with '-r' (recover) if there are any errors in the table. States will be updated as with '--update-state'.
-i, --information	Print statistics information about the table that is checked.
-m, --medium-check	Faster than extend-check , but only finds 99.99% of all errors. Should be good enough for most cases.
-U --update-state	Mark tables as crashed if you find any errors. This should be used to get the full benefit of the --check-only-changed option, but you shouldn't use this option if the mysqld server is using the table and you are running it with external locking disabled.
-T, --read-only	Don't mark table as checked. This is useful if you use myisamchk to check a table that is in use by some other application that does not use locking, such as mysqld when run with external locking disabled.

## Repairing Tables

It is also possible to repair MyISAM tables by using [REPAIR TABLE](#).

The following repair options are available, and are applicable when using '-r' or '-o':

Option	Description
-B, --backup	Make a backup of the .MYD file as 'filename-time.BAK'.
--correct-checksum	Correct the checksum information for table.
-D len, --data-file-length=#	Max length of data file (when recreating data file when it's full).
-e, --extend-check	Try to recover every possible row from the data file. Normally this will also find a lot of garbage rows; Don't use this option if you are not totally desperate.

-f, --force	Overwrite old temporary files. Add another --force to avoid 'myisam_sort_buffer_size is too small' errors. In this case we will attempt to do the repair with the given myisam_sort_buffer_size and dynamically allocate as many management buffers as needed.
-k val, --keys-used=#	Specify which keys to update. The value is a bit mask of which keys to use. Each binary bit corresponds to a table index, with the first index being bit 0. 0 disables all index updates, useful for faster inserts. Deactivated indexes can be reactivated by using <code>myisamchk -r</code> .
--create-missing-keys	Create missing keys. This assumes that the data file is correct and that the number of rows stored in the index file is correct. Enables --quick
--max-record-length=#	Skip rows larger than this if myisamchk can't allocate memory to hold them.
-r, --recover	Can fix almost anything except unique keys that aren't unique (a rare occurrence). Usually this is the best option to try first. Increase <code>myisam_sort_buffer_size</code> for better performance.
-n, --sort-recover	Forces recovering with sorting even if the temporary file would be very large.
-p, --parallel-recover	Uses the same technique as '-r' and '-n', but creates all the keys in parallel, in different threads.
-o, --safe-recover	Uses old recovery method; Slower than '-r' but uses less disk space and can handle a couple of cases where '-r' reports that it can't fix the data file. Increase <code>key_buffer_size</code> for better performance.
--character-sets-dir=directory_name	Directory where the <a href="#">character sets</a> are installed.
--set-collation=name	Change the collation (and by implication, the <a href="#">character set</a> ) used by the index.
-q, --quick	Faster repair by not modifying the data file. One can give a second '-q' to force myisamchk to modify the original datafile in case of duplicate keys. NOTE: Tables where the data file is corrupted can't be fixed with this option.
-u, --unpack	Unpack file packed with myisampack.

## Other Actions

Option	Description
-a, --analyze	Analyze distribution of keys. Will make some joins faster as the join optimizer can better choose the order in which to join the tables and which indexes to use. You can check the calculated distribution by using '--description --verbose table_name' or <a href="#">SHOW INDEX FROM table_name</a> .
--stats_method=name	Specifies how index statistics collection code should treat NULLs. Possible values of <code>name</code> are "nulls_unequal" (default), "nulls_equal" (emulate MySQL 4.0 behavior), and "nulls_ignored".
-d, --description	Print some descriptive information about the table. Specifying the --verbose option once or twice produces additional information.
-A [value], --set-auto-increment[=value]	Force auto_increment to start at this or higher value. If no value is given, then sets the next auto_increment value to the highest used value for the auto key + 1.
-S, --sort-index	Sort the index tree blocks in high-low order. This optimizes seeks and makes table scans that use indexes faster.
-R index_num, --sort-records=#	Sort records according to the given index (as specified by the index number). This makes your data much more localized and may speed up range-based SELECTs and ORDER BYs using this index. It may be VERY slow to do a sort the first time! To see the index numbers, <a href="#">SHOW INDEX</a> displays table indexes in the same order that myisamchk sees them. The first index is 1.
-b offset, --block-search=offset	Find the record to which a block at the given offset belongs.

For more, see [Memory and Disk Use With myisamchk](#).

## Examples

Check all the MyISAM tables in the current directory:

```
myisamchk *.MYI
```

If you are not in the database directory, you can check all the tables there by specifying the path to the directory:

```
myisamchk /path/to/database_dir/*.MYI
```

Check all tables in all databases by specifying a wildcard with the path to the MariaDB data directory:

```
myisamchk /path/to/datadir/*/*.MYI
```

The recommended way to quickly check all MyISAM tables:

```
myisamchk --silent --fast /path/to/datadir/*/*.MYI
```

Check all MyISAM tables and repair any that are corrupted:

```
myisamchk --silent --force --fast --update-state \  
--key_buffer_size=64M --sort_buffer_size=64M \  
--read_buffer_size=1M --write_buffer_size=1M \  
/path/to/datadir/*/*.MYI
```

## See Also

- [Memory and Disk Use With myisamchk](#)

### 4.3.13.4.2 Memory and Disk Use With myisamchk

myisamchk's performance can be dramatically enhanced for larger tables by making sure that its memory-related variables are set to an optimum level.

By default, myisamchk will use very little memory (about 3MB is allocated), but can temporarily use a lot of disk space. If disk space is a limitation when repairing, the `--safe-recover` option should be used instead of `--recover`. However, if TMPDIR points to a memory file system, an out of memory error can easily be caused, as myisamchk places temporary files in TMPDIR. The `--tmpdir=path` option should be used in this case to specify a directory on disk.

myisamchk has the following requirements for disk space:

- When repairing, space for twice the size of the data file, available in the same directory as the original file. This is for the original file as well as a copy. This space is not required if the `--quick` option is used, in which case only the index file is re-created.
- Disk space in the temporary directory (TMPDIR or the `tmpdir=path` option) is needed for sorting if the `--recover` or `--sort-recover` options are used when not using `--safe-recover`. The space required will be approximately  $(\text{largest\_key} + \text{row\_pointer\_length}) * \text{number\_of\_rows} * 2$ . To get information about the length of the keys as well as the row pointer length, use `myisamchk -dv table_name`.
- Space for a new index file to replace the existing one. The old index is first truncated, so unless the old index file is not present or is smaller for some reason, no significant extra space will be needed.

There are a number of [system variables](#) that are useful to adjust when running myisamchk. They will increase memory usage, and since some are per-session variables, you don't want to increase the general value, but you can either pass an increased value to myisamchk as a commandline option, or with a [myisamchk] section in your [my.cnf](#) file.

- [sort\\_buffer\\_size](#) . By default this is 4M, but it's very useful to increase to make myisamchk sorting much faster. Since the server won't be running when you run myisamchk, you can increase substantially. 16M is usually a minimum, but values such as 256M are not uncommon if memory is available.
- [key\\_buffer\\_size](#) (which particularly helps with the `--extend-check` and `--safe-recover` options).
- [read\\_buffer\\_size](#)
- [write\\_buffer\\_size](#)

For example, if you have more than 512MB available to allocate to the process, the following settings could be used:

```
myisamchk  
--myisam_sort_buffer_size=256M  
--key_buffer_size=512M  
--read_buffer_size=64M  
--write_buffer_size=64M  
...
```

### 4.3.13.4.3 myisamchk Table Information

#### Contents

1. [-dvv output](#)
2. [-eiv output](#)
3. [Examples](#)

[myisamchk](#) can be used to obtain information about MyISAM tables, particularly with the `-d` , `-e` , `-i` and `-v` options.

Common options for gathering information include:

- myisamchk -d
- myisamchk -dv
- myisamchk -dvv
- myisamchk -ei
- myisamchk -eiv

The `-d` option returns a short description of the table and its keys. Running the option while the table is being updated, and with external locking disabled,

may result in an error, but no damage will be done to the table. Each extra *v* adds more output. *-e* checks the table thoroughly (but slowly), and the *-i* options adds statistical information,

## -dvv output

The following table describes the output from the running myisamchk with the *-dvv* option:

Heading	Description
MyISAM file	Name and path of the MyISAM index file (without the extension)
Record format	<a href="#">Storage format</a> . One of <i>packed</i> (dynamic), <i>fixed</i> or <i>compressed</i> .
Character set	Default <a href="#">character set</a> for the table.
File-version	Always 1.
Creation time	Time the data file was created
Recover time	Most recent time the file was reconstructed.
Status	Table status. One or more of <i>analyzed</i> , <i>changed</i> , <i>crashed</i> , <i>open</i> , <i>optimized keys</i> and <i>sorted index pages</i> .
Auto increment key	Index number of the table's <a href="#">auto-increment</a> column. Not shown if no auto-increment column exists.
Last value	Most recently generated auto-increment value. Not shown if no auto-increment column exists.
Data records	Number of records in the table.
Deleted blocks	Number of deleted blocks that are still reserving space. Use <a href="#">OPTIMIZE TABLE</a> to defragment.
Datafile parts	For <a href="#">dynamic tables</a> , the number of data blocks. If the table is optimized, this will match the number of data records.
Deleted data	Number of bytes of unreclaimed deleted data, Use <a href="#">OPTIMIZE TABLE</a> to reclaim the space.
Datafile pointer	Size in bytes of the data file pointer. The size of the data file pointer, in bytes.
Keyfile pointer	Size in bytes of the index file pointer.
Max datafile length	Maximum length, in bytes, that the data file could become.
Max keyfile length	Maximum length, in bytes, that the index file could become.
Recordlength	Space, in bytes, that each row takes.
table description	Description of all indexes in the table, followed by all columns
Key	Index number, starting with one. If not shown, the index is part of a multiple-column index.
Start	Where the index part starts in the row.
Len	Length of the index or index part. The length of a multiple-column index is the sum of the component lengths. Indexes of string columns will be shorter than the full column length if only a string prefix is indexed.
Index	Whether an index value is unique or not. Either <i>multi</i> . or <i>unique</i> .
Type	Data type of the index of index part.
Rec/key	Record of the number of rows per value for the index or index part. Used by the optimizer to calculate query plans. Can be updated with <a href="#">myisamchk-a</a> . If not present, defaults to 30 .
Root	Root index block address.
Blocksize	Index block size, in bytes.
Field	Column number, starting with one. The first line will contain the position and number of bytes used to store NULL flags, if any (see <i>Nullpos</i> and <i>Nullbit</i> , below).
Start	Column's byte position within the table row.
Length	Column length, in bytes.
Nullpos	Byte containing the flag for NULL values. Empty if column cannot be NULL.

Nullbit	Bit containing the flag for NULL values. Empty if column cannot be NULL.
Type	Data type - see the table below for a list of possible values.
Huff tree	Only present for packed tables, contains the Huffman tree number associated with the column.
Bits	Only present for packed tables, contains the number of bits used in the Huffman tree.

Data type	Description
constant	All rows contain the same value.
noendspace	No endspace is stored.
no endspace, not_always	No endspace is stored, and endspace compression is not always performed for all values.
no endspace, no empty	No endspace is stored, no empty values are stored.
table-lookup	Column was converted to an <a href="#">ENUM</a> .
zerofill(N)	Most significant $N$ bytes of the value are not stored, as they are always zero.
no zeros	Zeros are not stored.
always zero	Zero values are stored with one bit.

## -eiv output

The following table describes the output from the running myisamchk with the `-eiv` option:

Heading	Description
Data records	Number of records in the table.
Deleted blocks	Number of deleted blocks that are still reserving space. Use <a href="#">OPTIMIZE TABLE</a> to defragment.
Key	Index number, starting with one.
Keyblocks used	Percentage of the keyblocks that are used. Percentages will be higher for optimized tables.
Packed	Percentage space saved from packing key values with a common suffix.
Max levels	Depth of the <a href="#">b-tree index</a> for the key. Larger tables and longer key values result in higher values.
Records	Number of records in the table.
M.recordlength	Average row length. For fixed rows, will be the actual length of each row.
Packed	Percentage saving from stripping spaces from the end of strings.
Recordspace used	Percentage of the data file used.
Empty space	Percentage of the data file unused.
Blocks/Record	Average number of blocks per record. Values higher than one indicate fragmentation. Use <a href="#">OPTIMIZE TABLE</a> to defragment.
Recordblocks	Number of used blocks. Will match the number of rows for fixed or optimized tables.
Deleteblocks	Number of deleted blocks
Recorddata	Used bytes in the data file.
Deleted data	Unused bytes in the data file.
Lost space	Total bytes lost, such as when a row is updated to a shorter length.
Linkdata	Sum of the bytes used for pointers linking disconnected blocks. Each is four to seven bytes in size.

## Examples

```
myisamchk -d /var/lib/mysql/test/posts

MyISAM file:      /var/lib/mysql/test/posts
Record format:    Compressed
Character set:   utf8mb4_unicode_ci (224)
Data records:     1680 Deleted blocks:          0
Recordlength:    2758
Using only keys '0' of 5 possibly keys
```

table description:				
Key	Start	Len	Index	Type
1	1	8	unique	ulonglong
2	2265	80	multip.	varchar prefix
	63	80		varchar
	17	5		binary
	1	8		ulonglong
3	1231	8	multip.	ulonglong
4	9	8	multip.	ulonglong
5	387	764	multip.	? prefix

```
myisamchk -dvv /var/lib/mysql/test/posts
```

```
MyISAM file:      /var/lib/mysql/test/posts
Record format:    Compressed
Character set:   utf8mb4_unicode_ci (224)
File-version:    1
Creation time:  2015-08-10 16:26:54
Recover time:   2015-08-10 16:26:54
Status:         checked,analyzed,optimized keys
Auto increment key: 1 Last value:           1811
Checksum:        2299272165
Data records:    1680 Deleted blocks:          0
Datafile parts:  1680 Deleted data:           0
Datafile pointer (bytes): 6 Keyfile pointer (bytes): 6
Datafile length: 4298092 Keyfile length:       156672
Max datafile length: 281474976710654 Max keyfile length: 288230376151710719
Recordlength:    2758
Using only keys '0' of 5 possibly keys
```

table description:						
Key	Start	Len	Index	Type	Rec/key	Root Blocksize
1	1	8	unique	ulonglong	1	1024
2	2265	80	multip.	varchar prefix	336	1024
	63	80		varchar	187	
	17	5		binary	1	
	1	8		ulonglong	1	
3	1231	8	multip.	ulonglong	10	1024
4	9	8	multip.	ulonglong	840	1024
5	387	764	multip.	? prefix	1	4096

Field	Start	Length	Nullpos	Nullbit	Type	Huff	tree	Bits
1	1	8			zerofill(6)	1		9
2	9	8			zerofill(7)	1		9
3	17	5				1		9
4	22	5				1		9
5	27	12			blob	2		9
6	39	10			blob	3		9
7	49	4			always zero	1		9
8	53	10			blob	1		9
9	63	81			varchar	4		9
10	144	81			varchar	5		5
11	225	81			varchar	5		5
12	306	81			varchar	1		9
13	387	802			varchar	6		9
14	1189	10			blob	1		9
15	1199	10			blob	7		9
16	1209	5				1		9
17	1214	5				1		9
18	1219	12			blob	1		9
19	1231	8			no zeros, zerofill(6)	1		9
20	1239	1022			varchar	7		9
21	2261	4			always zero	1		9
22	2265	81			varchar	8		8
23	2346	402			varchar	2		9
24	2748	8			no zeros, zerofill(7)	1		9

```

myisamchk -eiv /var/lib/mysql/test/posts
Checking MyISAM file: /var/lib/mysql/test/posts
Data records: 1680 Deleted blocks: 0
- check file-size
- check record delete-chain
No recordlinks
- check key delete-chain
block_size 1024:
block_size 2048:
block_size 3072:
block_size 4096:
- check index reference
- check data record references index: 1
Key: 1: Keyblocks used: 92% Packed: 0% Max levels: 2
- check data record references index: 2
Key: 2: Keyblocks used: 93% Packed: 90% Max levels: 2
- check data record references index: 3
Key: 3: Keyblocks used: 92% Packed: 0% Max levels: 2
- check data record references index: 4
Key: 4: Keyblocks used: 92% Packed: 0% Max levels: 2
- check data record references index: 5
Key: 5: Keyblocks used: 88% Packed: 97% Max levels: 2
Total: Keyblocks used: 91% Packed: 91%

- check records and index references
Records: 1680 M.recordlength: 4102 Packed: 0%
Recordspace used: 100% Empty space: 0% Blocks/Record: 1.00
Record blocks: 1680 Delete blocks: 0
Record data: 6892064 Deleted data: 0
Lost space: 1284 Linkdata: 6264

User time 0.11, System time 0.00
Maximum resident set size 3036, Integral resident set size 0
Non-physical pagefaults 925, Physical pagefaults 0, Swaps 0
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 0, Involuntary context switches 74

```

## 4.3.13.4.4 myisamlog

`myisamlog`  
processes and returns the contents of a [MyISAM log file](#).

Invoke

`myisamlog`  
like this:

```

shell> myisamlog [options] [log_file [tbl_name] ...]
shell> isamlog [options] [log_file [tbl_name] ...]

```

The default operation is update (

- u
- ). If a recovery is done (
- r
- ), all writes and possibly updates and deletes are done and errors are only counted. The default log file name is `myisam.log`
- for
- `myisamlog`
- and
- `isam.log`
- for
- `isamlog`
- if no
- `log_file`

argument is given. If tables are named on the command line, only those tables are updated.

`myisamlog`  
supports the following options:

Option	Description
--------	-------------

-?	Display a help message and exit.
,	
-I	
-c <i>N</i>	Execute only <i>N</i> commands.
-f <i>N</i>	Specify the maximum number of open files.
-i	Display extra information before exiting.
-o <i>offset</i>	Specify the starting offset.
-p <i>N</i>	Remove <i>N</i> components from path.
-r	Perform a recovery operation.
-R <i>record_pos_file</i> <i>record_pos</i>	Specify record position file and record position.
-u	Displays update operations.
-v	Verbose mode. Print more output about what the program does. This option can be given multiple times ( -vv , -vvv ) to produce more and more output.
-w <i>write_file</i>	Specify the write file.
-V	Display version information.

### 4.3.13.4.5 myisampack

## Contents

1. Options
2. Uncompressing
3. Examples
4. See Also

`myisampack`

is a tool for compressing [MyISAM](#) tables. The resulting tables are read-only, and usually about 40% to 70% smaller. It is run as follows:

```
myisampack [options] file_name [file_name2...]
```

The

`file_name`  
is the  
`.MYI`

index file. The extension can be omitted, although keeping it permits wildcards, such as:

```
myisampack *.MYI
```

...to compress all the files.

`myisampack`

compresses each column separately, and, when the resulting data is read, only the individual rows and columns required need to be decompressed, allowing for quicker reading.

Once a table has been packed, use

```
myisamchk -rq
```

(the quick and recover options) to rebuild its indexes.

`myisampack`

does not support partitioned tables.

Do not run `myisampack` if the tables could be updated during the operation, and `skip_external_locking` has been set.

## Options

The following variables can be set while passed as commandline options to

`myisampack`  
, or set with a  
[`myisampack`]  
section in your [my.cnf](#) file.

Option	Description
<code>-b</code> , <code>--backup</code>	Make a backup of the table as <code>table_name.OLD</code> .
<code>--character-sets-dir=name</code>	Directory where character sets are.
<code>-#</code> , <code>--debug[=name]</code>	Output debug log. Often this is <code>'d:t:o,filename'</code> .
<code>-f</code> , <code>--force</code>	Force packing of table even if it gets bigger or if tempfile exists.

-j , --join=name	Join all given tables into 'new_table_name' . All tables <b>must</b> have identical layouts.
-? , --help	Display help and exit.
-s , --silent	Only write output when an error occurs
-T , --tmpdir=name	Use temporary directory to store temporary table.
-t , --test	Don't pack table, only test packing it.
-v , --verbose	Write info about progress and packing result. Use multiple -v flags for more verbosity.
-V , --version	Output version information and exit.
-w , --wait	Wait and retry if table is in use.

## Uncompressing

To uncompress a table compressed with

`myisampack`

, use the

`myisamchk -u`

option.

## Examples

```
> myisampack /var/lib/mysql/test/posts
Compressing /var/lib/mysql/test/posts.MYD: (1680 records)
- Calculating statistics
- Compressing file
37.71%
> myisamchk -rq /var/lib/mysql/test/posts
- check record delete-chain
- recovering (with sort) MyISAM-table '/var/lib/mysql/test/posts'
Data records: 1680
- Fixing index 1
- Fixing index 2
```

## See Also

- [FLUSH TABLES FOR EXPORT](#)
- [myisamchk](#)

### 4.3.13.4.6 myisam\_ftdump

myisam\_ftdump is a utility for displaying information about MyISAM FULLTEXT indexes. It will scan and dump the entire index, and can be a lengthy process.

If the server is running, make sure you run a [FLUSH TABLES](#) statement first.

## Usage

```
myisam_ftdump <table_name> <index_num>
```

The table\_name can be specified with or without the .MYI index extension.

The index number refers to the number of the index when the table was defined, starting at zero. For example, take the following table definition:

```
CREATE TABLE IF NOT EXISTS `employees_example` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `first_name` varchar(30) NOT NULL,
  `last_name` varchar(40) NOT NULL,
  `position` varchar(25) NOT NULL,
  `home_address` varchar(50) NOT NULL,
  `home_phone` varchar(12) NOT NULL,
  `employee_code` varchar(25) NOT NULL,
  `bio` text NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `employee_code` (`employee_code`),
  FULLTEXT (`bio`)
) ENGINE=MyISAM;
```

The fulltext index will be

```
2
. The primary key is index
0
, and the unique key index
1
.
```

You can use *myisam\_ftdump* to generate a list of index entries in order of frequency of occurrence as follows:

```
myisam_ftdump -c mytexttable 1 | sort -r
```

## Options

Option	Description
-h, --help	Display help and exit.
-?, --help	Synonym for -h.
-c, --count	Calculate per-word stats (counts and global weights).
-d, --dump	Dump index (incl. data offsets and word weights).
-l, --length	Report length distribution.
-s, --stats	Report global stats.
-v, --verbose	Be verbose.

### 4.3.13.5 MyISAM Index Storage Space

Regular MyISAM tables make use of [B-tree indexes](#).

String indexes are space-compressed, which reduces the size of [VARCHARs](#) that don't use the full length, or a string that has trailing spaces. String indexes also make use of prefix-compression, where strings with identical prefixes are compressed.

Numeric indexes can also be prefix-compressed if the [PACK\\_KEYS=1](#) option is used. Regardless, the high byte is always stored first, which allows a reduced index size.

In the worst case, with no strings being space-compressed, the total index storage space will be  $(\text{index\_length}+4)/0.67$  per index.

## 4.3.13.6 MyISAM Log

The MyISAM log records all changes to [MyISAM](#) tables. It is not enabled by default. To enable it, start the server with the `--log-isam` option, for example:

```
--log-isam=myisam.log
```

The *isam* instead of *myisam* above is not a typo - it's a legacy from when the predecessor to the MyISAM format, called ISAM. The option can be used without specifying a filename, in which case the default, *myisam.log* is used.

To process the contents of the log file, use the [myisamlog](#) utility.

## 4.3.13.7 Concurrent Inserts

### Contents

1. [Notes](#)
2. [See Also](#)

The [MyISAM](#) storage engine supports concurrent inserts. This feature allows [SELECT](#) statements to be executed during [INSERT](#) operations, reducing contention.

Whether concurrent inserts can be used or not depends on the value of the [concurrent\\_insert](#) server system variable:

- NEVER  
(0) disables concurrent inserts.
- AUTO  
(1) allows concurrent inserts only when the target table has no free blocks (no data in the middle of the table has been deleted after the last [OPTIMIZE TABLE](#)). This is the default.
- ALWAYS  
(2) always enables concurrent inserts, in which case new rows are added at the end of a table if the table is being used by another thread.

If the [binary log](#) is used, [CREATE TABLE ... SELECT](#) and [INSERT ... SELECT](#) statements cannot use concurrent inserts. These statements acquire a read lock on the table, so concurrent inserts will need to wait. This way the log can be safely used to restore data.

Concurrent inserts are not used by replicas with the row based [replication](#) (see [binary log formats](#) ).

If an [INSERT](#) statement contain the [HIGH\\_PRIORITY](#) clause, concurrent inserts cannot be used. [INSERT ... DELAYED](#) is usually unneeded if concurrent inserts are enabled.

[LOAD DATA INFILE](#) uses concurrent inserts if the

CONCURRENT  
keyword is specified and [concurrent\\_insert](#) is not  
NEVER  
. This makes the statement slower (even if no other sessions access the table) but reduces contention.

[LOCK TABLES](#) allows non-conflicting concurrent inserts if a

READ LOCAL  
lock is used. Concurrent inserts are not allowed if the  
LOCAL  
keyword is omitted.

## Notes

The decision to enable concurrent insert for a table is done when the table is opened. If you change the value of [concurrent\\_insert](#) it will only affect new opened tables. If you want it to work for also for tables in use or cached, you should do [FLUSH TABLES](#) after setting the variable.

## See Also

- [INSERT](#)
- [INSERT DELAYED](#)
- [INSERT SELECT](#)
- [HIGH\\_PRIORITY and LOW\\_PRIORITY](#)
- [INSERT - Default & Duplicate Values](#)
- [INSERT IGNORE](#)
- [INSERT ON DUPLICATE KEY UPDATE](#)

## 4.3.13.8 Segmented Key Cache

## Contents

1. About Segmented Key Cache
2. Segmented Key Cache Syntax
3. Segmented Key Cache Statistics
4. See Also

## About Segmented Key Cache

A segmented key cache is a collection of structures for regular [MyISAM](#) key caches called key cache segments. Segmented key caches mitigate one of the major problems of the simple key cache: thread contention for key cache lock (mutex). With regular key caches, every call of a key cache interface function must acquire this lock. So threads compete for this lock even in the case when they have acquired shared locks for the file and the pages they want to read from are in the key cache buffers.

When working with a segmented key cache any key cache interface function that needs only one page has to acquire the key cache lock only for the segment the page is assigned to. This makes the chances for threads not having to compete for the same key cache lock better.

Any page from a file can be placed into a buffer of only one segment. The number of the segment is calculated from the file number and the position of the page in the file, and it's always the same for the page. Pages are evenly distributed among segments.

The idea and the original code of the segmented key cache was provided by Fredrik Nylander from Stardoll.com. The code was extensively reworked, improved, and eventually merged into MariaDB by Igor Babaev from Monty Program (now MariaDB Corporation).

You can find some benchmark results comparing various settings on the [Segmented Key Cache Performance](#) page.

## Segmented Key Cache Syntax

New global variable: `key_cache_segments`. This variable sets the number of segments in a key cache. Valid values for this variable are whole numbers between

0  
and  
64  
. If the number of segments is set to a number greater than  
64  
the number of segments will be truncated to 64 and a warning will be issued.

A value of

0  
means the key cache is a regular (i.e. non-segmented) key cache. This is the default. If  
`key_cache_segments`  
is  
1  
(or higher) then the new key cache segmentation code is used. In practice there is no practical use of a single-segment segmented key cache  
except for testing purposes, and setting  
`key_cache_segments = 1`  
should be slower than any other option and should not be used in production.

Other global variables used when working with regular key caches also apply to segmented key caches: `key_buffer_size`, `key_cache_age_threshold`, `key_cache_block_size`, and `key_cache_division_limit`.

## Segmented Key Cache Statistics

Statistics about the key cache can be found by looking at the `KEY_CACHES` table in the `INFORMATION_SCHEMA` database. Columns in this table are:

Column Name	Description
<code>KEY_CACHE_NAME</code>	The name of the key cache
<code>SEGMENTS</code>	total number of segments (set to NULL for regular key caches)
<code>SEGMENT_NUMBER</code>	segment number (set to NULL for any regular key caches and for rows containing aggregation statistics for segmented key caches)
<code>FULL_SIZE</code>	memory for cache buffers/auxiliary structures
<code>BLOCK_SIZE</code>	size of the blocks

USED_BLOCKS	number of currently used blocks
UNUSED_BLOCKS	number of currently unused blocks
DIRTY_BLOCKS	number of currently dirty blocks
READ_REQUESTS	number of read requests
READS	number of actual reads from files into buffers
WRITE_REQUESTS	number of write requests
WRITES	number of actual writes from buffers into files

## See Also

- [Segmented Key Cache Performance](#)

### 4.3.14 MyRocks

MyRocks is a storage engine that adds the RocksDB database to MariaDB. RocksDB is an LSM database with a great compression ratio that is optimized for flash storage.

MyRocks Version	Introduced	Maturity
MyRocks 1.0	<a href="#">MariaDB 10.3.7 , MariaDB 10.2.16</a>	Stable
MyRocks 1.0	<a href="#">MariaDB 10.3.5 , MariaDB 10.2.14</a>	Gamma
MyRocks 1.0	<a href="#">MariaDB 10.3.4 , MariaDB 10.2.13</a>	Beta
MyRocks 1.0	<a href="#">MariaDB 10.2.5</a>	Alpha



#### About MyRocks for MariaDB

*Enables greater compression than InnoDB, and less write amplification.*



#### Getting Started with MyRocks

*Installing and getting started with MyRocks.*



#### Building MyRocks in MariaDB

*MariaDB compile process for MyRocks.*



#### Loading Data Into MyRocks

*MyRocks has ways to load data much faster than normal INSERTs*



#### MyRocks Status Variables

*MyRocks-related status variables.*



#### MyRocks System Variables

*MyRocks server system variables.*



#### MyRocks Transactional Isolation

*TODO: MyRocks uses snapshot isolation Support do READ-COMMITTED and REPEAT...*



#### MyRocks and Replication

*Details about how MyRocks works with replication.*



#### MyRocks and Group Commit with Binary log

*MyRocks supports group commit with the binary log*



## Optimizer Statistics in MyRocks

*How MyRocks provides statistics to the query optimizer*



## Differences Between MyRocks Variants

*Differences between Facebook's, MariaDB's and Percona Server's MyRocks.*



## MyRocks and Bloom Filters

*Bloom filters are used to reduce read amplification.*



## MyRocks and CHECK TABLE

*MyRocks supports the CHECK TABLE command.*



## MyRocks and Data Compression

*MyRocks supports several compression algorithms.*



## MyRocks and Index-Only Scans

*MyRocks and index-only scans on secondary indexes.*



## MyRocks and START TRANSACTION WITH CONSISTENT SNAPSHOT

*FB/MySQL has added new syntax which returns the binlog coordinates pointing at the snapshot.*



## MyRocks Column Families

*MyRocks stores data in column families, which are similar to tablespaces.*



## MyRocks in MariaDB 10.2 vs MariaDB 10.3

*MyRocks storage engine in MariaDB 10.2 and MariaDB 10.3.*



## MyRocks Performance Troubleshooting

*MyRocks exposes its performance metrics through several interfaces.*

### 4.3.14.1 About MyRocks for MariaDB

#### Contents

1. [About MyRocks for MariaDB](#)

1. Benefits

1. Greater Space Efficiency
2. Greater Writing Efficiency
3. Faster Data Loading
4. Faster Replication

2. Requirements and Limitations

## About MyRocks for MariaDB

MyRocks is an open source storage engine that was originally developed by Facebook.

MyRocks has been extended by the MariaDB engineering team to be a pluggable storage engine that you use in your MariaDB solutions. It works seamlessly with MariaDB features. This openness in the storage layer allows you to use the right storage engine to optimize your usage requirements, which provides optimum performance. Community contributions are one of MariaDB's greatest advantages over other databases. Under the lead of our developer Sergey Petrunia, MyRocks in MariaDB is occasionally being merged with upstream MyRocks from Facebook.

See more at: <https://mariadb.com/resources/blog/facebook-myrocks-mariadb#sthash.ZIEr7kNq.dpuf>



MyRocks, typically, gives greater performance for web scale type applications. It can be an ideal storage engine solution when you have workloads that require greater compression and IO efficiency. It uses a Log Structured Merge (LSM) architecture, which has advantages over B-Tree algorithms, to provide efficient data ingestion, like read-free replication slaves, or fast bulk data loading. MyRocks distinguishing features include:

- compaction filter
- merge operator
- backup
- column families
- bulk loading
- persistent cache

## Benefits

On production workloads, MyRocks was tested to prove that it provides:

### Greater Space Efficiency

- 2x more compression

MyRocks has 2x better compression compared to compressed InnoDB, 3-4x better compression compared to uncompressed InnoDB, meaning you use less space.

### Greater Writing Efficiency

- 2x lower write rates to storage

MyRocks has a 10x less write amplification compared to InnoDB, giving you better endurance of flash storage and improving overall throughput.

### Faster Data Loading

- faster database loads

MyRocks writes data directly onto the bottommost level, which avoids all compaction overheads when you enable faster data loading for a session.

### Faster Replication

- No random reads for updating secondary keys, except for unique indexes. The Read-Free Replication option does away with random reads when updating primary keys, regardless of uniqueness, with a row-based binary logging format.

<http://myrocks.io> <https://mariadb.com/resources/blog/facebook-myrocks-mariadb>

## Requirements and Limitations

- MyRocks is included from [MariaDB 10.2.5](#).
- MyRocks is available in the MariaDB Server packages for Linux and Windows.
- Maria DB optimistic parallel replication may not be supported.
- MyRocks is not available for 32-bit platforms
- [Galera Cluster](#) is tightly integrated into InnoDB storage engine (it also supports Percona's XtraDB which is a modified version of InnoDB). Galera Cluster does not work with any other storage engines, including MyRocks (or TokuDB for example).

MyRocks builds are available on platforms that support a sufficiently modern compiler, for example:

- Ubuntu Trusty, Xenial, (amd64 and ppc64el)
- Ubuntu Yakkety (amd64)
- Debian Jessie, stable (amd64, ppc64el)
- Debian Stretch, Sid (testing and unstable) (amd64)
- CentOS/RHEL 7 (amd64)
- Centos/RHEL 7.3 (amd64)
- Fedora 24 and 25 (amd64)
- OpenSUSE 42 (amd64)
- Windows 64 (zip and MSI)

### 4.3.14.2 Getting Started with MyRocks

MariaDB starting with [10.2.5](#)

The MyRocks storage engine was first released in [MariaDB 10.2.5](#).

MyRocks is a storage engine that adds the RocksDB database to MariaDB. RocksDB is an LSM database with a great compression ratio that is optimized for flash storage.

The storage engine must be installed before it can be used.

## Contents

1. [Installing the Plugin's Package](#)
  1. [Installing on Linux](#)
    1. [Installing with a Package Manager](#)
      1. [Installing with yum/dnf](#)
      2. [Installing with apt-get](#)
      3. [Installing with zypper](#)
    2. [Installing on Windows](#)
  2. [Installing the Plugin](#)
  3. [Uninstalling the Plugin](#)
  4. [Verifying the Installation](#)
  5. [Compression](#)
  6. [System and Status Variables](#)

## Installing the Plugin's Package

The MyRocks storage engine's shared library is included in MariaDB packages as the

`ha_rocksdb.so`

or

`ha_rocksdb.dll`

shared library on systems where it can be built. The plugin was first included in [MariaDB 10.2.5](#).

## Installing on Linux

The MyRocks storage engine is included in [binary tarballs](#) on Linux.

### Installing with a Package Manager

The MyRocks storage engine can also be installed via a package manager on Linux. In order to do so, your system needs to be configured to install from one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the [MariaDB Package Repository setup script](#).

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the [MariaDB Repository Configuration Tool](#).

#### Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant [RPM package](#) from MariaDB's repository using

`yum`

or

`dnf`

. Starting with RHEL 8 and Fedora 22,

`yum`

has been replaced by

`dnf`

, which is the next major version of

`yum`

. However,

`yum`

commands still work on many systems that use

`dnf`

. For example:

```
sudo yum install MariaDB-rocksdb-engine
```

#### Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant [DEB package](#) from MariaDB's repository using

`apt-get`

. For example:

```
sudo apt-get install mariadb-plugin-rocksdb
```

Installing with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant [RPM package](#) from MariaDB's repository using

```
zypper
```

. For example:

```
sudo zypper install MariaDB-rocksdb-engine
```

## Installing on Windows

The MyRocks storage engine is included in [MSI](#) and [ZIP](#) packages on Windows.

## Installing the Plugin

Once the shared library is in place, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'ha_rocksdb';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#) . For example:

```
[mariadb]
...
plugin_load_add = ha_rocksdb
```

Note: When installed with a package manager, an option file that contains the

```
--plugin-load-add
```

option may also be installed. The RPM package installs it as

/etc/my.cnf.d/rocksdb.cnf

, and the DEB package installs it as

/etc/mysql/mariadb.conf.d/rocksdb.cnf

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'ha_rocksdb';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Verifying the Installation

After installing MyRocks you will see RocksDB in the list of plugins:

```
SHOW PLUGINS;
+-----+-----+-----+-----+
| Name           | Status | Type      | Library          | License |
+-----+-----+-----+-----+
...
| ROCKSDB         | ACTIVE | STORAGE ENGINE | ha_rocksdb.so | GPL    |
| ROCKSDB_CFSTATS | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_DBSTATS | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_PERF_CONTEXT | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_PERF_CONTEXT_GLOBAL | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_CF_OPTIONS | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_COMPACTION_STATS | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_GLOBAL_INFO | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_DDL     | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_INDEX_FILE_MAP | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_LOCKS   | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
| ROCKSDB_TRX     | ACTIVE | INFORMATION SCHEMA | ha_rocksdb.so | GPL    |
...
+-----+-----+-----+-----+
```

## Compression

Supported compression types are listed in the `rocksdb_supported_compression_types` variable. For example:

```
SHOW VARIABLES LIKE 'rocksdb_supported_compression_types';
+-----+-----+
| Variable_name        | Value   |
+-----+-----+
| rocksdb_supported_compression_types | Snappy,Zlib |
+-----+-----+
```

See [MyRocks and Data Compression](#) for more.

## System and Status Variables

All MyRocks [system variables](#) and [status variables](#) are prefaced with "rocksdb", so you can query them with, for example:

```
SHOW VARIABLES LIKE 'rocksdb%';
SHOW STATUS LIKE 'rocksdb%';
```

## 4.3.14.3 Building MyRocks in MariaDB

## Contents

1. Build Process and Requirements
2. Building on Ubuntu 16.04
3. Starting MyRocks

This page describes how to get [MyRocks in MariaDB](#) when compiling MariaDB from source.

(See <https://github.com/facebook/mysql-5.6/wiki/Build-Steps> for instructions how to build the upstream)

## Build Process and Requirements

MariaDB compile process will compile [MyRocks](#) into

ha\_rocksdb.so  
by default if the platform supports it (That is, no WITH\_ROCKSDB switch is necessary).

Platform requirements:

- A 64-bit platform (due to some 32 bit compilers having difficulties with RocksDB)
- git installed (or git submodules fetched somehow)
- A sufficiently recent compiler:
  - gcc >= 4.8, or
  - clang >= 3.3, or
  - MS Visual Studio 2015 or newer

## Building on Ubuntu 16.04

The steps were checked on a fresh install of Ubuntu 16.04.2 LTS Xenial.

```
sudo apt-get update
sudo apt-get -y install g++ cmake libbz2-dev libaio-dev bison zlib1g-dev libsnapy-dev
sudo apt-get -y install libgflags-dev libreadline6-dev libncurses5-dev libssl-dev liblz4-dev gdb git
;
```

```
git clone https://github.com/MariaDB/server.git mariadb-10.2
cd mariadb-10.2
git checkout 10.2
git submodule init
git submodule update
cmake .
make -j10
```

This should produce

```
storage/rocksdb/ha_rocksdb.so
which is MyRocks storage engine in the loadable form.
```

## Starting MyRocks

MyRocks does not require any special way to initialize the data directory. Minimal my.cnf file:

```
cat > ~/my1.cnf <<EOF
[mysqld]

datadir=../mysql-test/var/install.db
plugin-dir=../storage/rocksdb
language=./share/english
socket=/tmp/mysql.sock
port=3307

plugin-load=ha_rocksdb
default-storage-engine=rocksdb
EOF
```

Run the server like this

```
(cd mysql-test; ./mtr alias)
cp -r mysql-test/var/install.db ~./data1
cd ..sql
./mysqld --defaults-file=~/my1.cnf
```

Compression libraries. Supported compression libraries are listed in [rocksdb\\_supported\\_compression\\_types](#). Compiling like the above, I get:

## 4.3.14.4 Loading Data Into MyRocks

Being a write-optimized storage engine, MyRocks has special ways to load data much faster than normal INSERTs would.

See

- <http://myrocks.io/docs/getting-started/> ; the section about "Migrating from InnoDB to MyRocks in production" has some clues.
- <https://github.com/facebook/mysql-5.6/wiki/Data-Loading> covers the topic in greater detail.

Note When one loads data with `rocksdb_bulk_load=1` and the data conflicts with the data already in the database, one may get non-trivial errors, for example:

```
ERROR 1105 (HY000): [./rocksdb/test.t1_PRIMARY_2_0.bulk_load.tmp] bulk load error:  
Invalid argument: External file requires flush
```

## 4.3.14.5 MyRocks Status Variables

### Contents

1. [Rocksdb\\_block\\_cache\\_add](#)
2. [Rocksdb\\_block\\_cache\\_add\\_failures](#)
3. [Rocksdb\\_block\\_cache\\_bytes\\_read](#)
4. [Rocksdb\\_block\\_cache\\_bytes\\_write](#)
5. [Rocksdb\\_block\\_cache\\_data\\_add](#)
6. [Rocksdb\\_block\\_cache\\_data\\_bytes\\_insert](#)
7. [Rocksdb\\_block\\_cache\\_data\\_hit](#)
8. [Rocksdb\\_block\\_cache\\_data\\_miss](#)
9. [Rocksdb\\_block\\_cache\\_filter\\_add](#)
10. [Rocksdb\\_block\\_cache\\_filter\\_bytes\\_evict](#)
11. [Rocksdb\\_block\\_cache\\_filter\\_bytes\\_insert](#)
12. [Rocksdb\\_block\\_cache\\_filter\\_hit](#)
13. [Rocksdb\\_block\\_cache\\_filter\\_miss](#)
14. [Rocksdb\\_block\\_cache\\_hit](#)
15. [Rocksdb\\_block\\_cache\\_index\\_add](#)
16. [Rocksdb\\_block\\_cache\\_index\\_bytes\\_evict](#)
17. [Rocksdb\\_block\\_cache\\_index\\_bytes\\_insert](#)
18. [Rocksdb\\_block\\_cache\\_index\\_hit](#)
19. [Rocksdb\\_block\\_cache\\_index\\_miss](#)
20. [Rocksdb\\_block\\_cache\\_miss](#)
21. [Rocksdb\\_block\\_cachecompressed\\_hit](#)
22. [Rocksdb\\_block\\_cachecompressed\\_miss](#)
23. [Rocksdb\\_bloom\\_filter\\_full\\_positive](#)
24. [Rocksdb\\_bloom\\_filter\\_full\\_true\\_positive](#)
25. [Rocksdb\\_bloom\\_filter\\_prefix\\_checked](#)
26. [Rocksdb\\_bloom\\_filter\\_prefix\\_useful](#)
27. [Rocksdb\\_bloom\\_filter\\_useful](#)
28. [Rocksdb\\_bytes\\_read](#)
29. [Rocksdb\\_bytes\\_written](#)
30. [Rocksdb\\_compact\\_read\\_bytes](#)
31. [Rocksdb\\_compact\\_write\\_bytes](#)
32. [Rocksdb\\_compaction\\_key\\_drop\\_new](#)
33. [Rocksdb\\_compaction\\_key\\_drop\\_obsolete](#)
34. [Rocksdb\\_compaction\\_key\\_drop\\_user](#)
35. [Rocksdb\\_covered\\_secondary\\_key\\_lookups](#)
36. [Rocksdb\\_flush\\_write\\_bytes](#)
37. [Rocksdb\\_get\\_hit\\_I0](#)
38. [Rocksdb\\_get\\_hit\\_I1](#)
39. [Rocksdb\\_get\\_hit\\_I2\\_and\\_up](#)
40. [Rocksdb\\_getupdatessince\\_calls](#)
41. [Rocksdb\\_iter\\_bytes\\_read](#)
42. [Rocksdb\\_I0\\_num\\_files\\_stall\\_micros](#)
43. [Rocksdb\\_I0\\_slowdown\\_micros](#)
44. [Rocksdb\\_manual\\_compactions\\_processed](#)
45. [Rocksdb\\_manual\\_compactions\\_running](#)
46. [Rocksdb\\_memtable\\_compaction\\_micros](#)
47. [Rocksdb\\_memtable\\_hit](#)
48. [Rocksdb\\_memtable\\_miss](#)
49. [Rocksdb\\_memtable\\_total](#)
50. [Rocksdb\\_memtable\\_unflushed](#)
51. [Rocksdb\\_no\\_file\\_closes](#)

52. Rocksdb\_no\_file\_errors  
53. Rocksdb\_no\_fileOpens  
54. Rocksdb\_num\_iterators  
55. Rocksdb\_number\_block\_not\_compressed  
56. Rocksdb\_number\_db\_next  
57. Rocksdb\_number\_db\_next\_found  
58. Rocksdb\_number\_db\_prev  
59. Rocksdb\_number\_db\_prev\_found  
60. Rocksdb\_number\_db\_seek  
61. Rocksdb\_number\_db\_seek\_found  
62. Rocksdb\_number\_deletes\_filtered  
63. Rocksdb\_number\_keys\_read  
64. Rocksdb\_number\_keys\_updated  
65. Rocksdb\_number\_keys\_written  
66. Rocksdb\_number\_merge\_failures  
67. Rocksdb\_number\_multiget\_bytes\_read  
68. Rocksdb\_number\_multiget\_get  
69. Rocksdb\_number\_multiget\_keys\_read  
70. Rocksdb\_number\_reseeks\_iteration  
71. Rocksdb\_number\_sst\_entry\_delete  
72. Rocksdb\_number\_sst\_entry\_merge  
73. Rocksdb\_number\_sst\_entry\_other  
74. Rocksdb\_number\_sst\_entry\_put  
75. Rocksdb\_number\_sst\_entry\_singledelete  
76. Rocksdb\_number\_superversion\_acquires  
77. Rocksdb\_number\_superversion\_cleanups  
78. Rocksdb\_number\_superversion\_releases  
79. Rocksdb\_queries\_point  
80. Rocksdb\_queries\_range  
81. Rocksdb\_row\_lock\_deadlocks  
82. Rocksdb\_row\_lock\_wait\_timeouts  
83. Rocksdb\_rows\_deleted  
84. Rocksdb\_rows\_deleted\_blind  
85. Rocksdb\_rows\_expired  
86. Rocksdb\_rows\_filtered  
87. Rocksdb\_rows\_inserted  
88. Rocksdb\_rows\_read  
89. Rocksdb\_rows\_updated  
90. Rocksdb\_snapshot\_conflict\_errors  
91. Rocksdb\_stall\_I0\_file\_count\_limit\_slowdowns  
92. Rocksdb\_stall\_I0\_file\_count\_limit\_stops  
93. Rocksdb\_stall\_locked\_I0\_file\_count\_limit\_stops  
94. Rocksdb\_stall\_locked\_I0\_file\_count\_limit\_stops  
95. Rocksdb\_stall\_memtable\_limit\_slowdowns  
96. Rocksdb\_stall\_memtable\_limit\_stops  
97. Rocksdb\_stall\_micros  
98. Rocksdb\_stall\_pending\_compaction\_limit\_s  
99. Rocksdb\_stall\_pending\_compaction\_limit\_s  
00. Rocksdb\_stall\_total\_slowdowns  
01. Rocksdb\_stall\_total\_stops  
02. Rocksdb\_system\_rows\_deleted  
03. Rocksdb\_system\_rows\_inserted  
04. Rocksdb\_system\_rows\_read  
05. Rocksdb\_system\_rows\_updated  
06. Rocksdb\_wal\_bytes  
07. Rocksdb\_wal\_group\_syncs  
08. Rocksdb\_wal\_synced  
09. Rocksdb\_write\_other  
10. Rocksdb\_write\_self  
11. Rocksdb\_write\_timedout  
12. Rocksdb\_write\_wal

This page documents status variables related to the [MyRocks](#) storage engine. See [Server Status Variables](#) for a complete list of status variables that can be viewed with `SHOW STATUS`.

See also the [Full list of MariaDB options, system and status variables](#).

[Rocksdb\\_block\\_cache\\_add](#)

- **Description:** Number of blocks added to the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_block\_cache\_add\_failures

- **Description:** Number of failures when adding blocks to Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_block\_cache\_bytes\_read

- **Description:** Bytes read from Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_block\_cache\_bytes\_write

- **Description:** Bytes written to Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_block\_cache\_data\_add

- **Description:** Number of data blocks added to the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_block\_cache\_data\_bytes\_insert

- **Description:** Bytes added to the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_block\_cache\_data\_hit

- **Description:** Number of hits when accessing the data block from the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_block\_cache\_data\_miss

- **Description:** Number of misses when accessing the data block from the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_block\_cache\_filter\_add

- **Description:** Number of bloom filter blocks added to the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_block\_cache\_filter\_bytes\_evict

- **Description:** Bytes of bloom filter blocks evicted from the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_block\_cache\_filter\_bytes\_insert

- **Description:** Bytes of bloom filter blocks added to the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_block\_cache\_filter\_hit

- **Description:** Number of hits when accessing the filter block from the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_block\_cache\_filter\_miss

- **Description:** Number of misses when accessing the filter block from the Block Cache.

- **Scope:** Global, Session

- **Data Type:**

numeric

---

### Rocksdb\_block\_cache\_hit

- **Description:** Total number of hits for the Block Cache.
- **Scope:** Global, Session
- **Data Type:**

numeric

---

### Rocksdb\_block\_cache\_index\_add

- **Description:** Number of index blocks added to Block Cache index.
- **Scope:** Global, Session
- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

### Rocksdb\_block\_cache\_index\_bytes\_evict

- **Description:** Bytes of index blocks evicted from the Block Cache.
- **Scope:** Global, Session
- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

### Rocksdb\_block\_cache\_index\_bytes\_insert

- **Description:** Bytes of index blocks added to the Block Cache.
- **Scope:** Global, Session
- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

### Rocksdb\_block\_cache\_index\_hit

- **Description:** Number of hits for the Block Cache index.
- **Scope:** Global, Session
- **Data Type:**

numeric

---

### Rocksdb\_block\_cache\_index\_miss

- **Description:** Number of misses for the Block Cache index.
- **Scope:** Global, Session
- **Data Type:**

numeric

---

---

### Rocksdb\_block\_cache\_miss

- **Description:** Total number of misses for the Block Cache.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_block\_cachecompressed\_hit

- **Description:** Number of hits for the compressed Block Cache.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_block\_cachecompressed\_miss

- **Description:** Number of misses for the compressed Block Cache.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_bloom\_filter\_full\_positive

- **Description:**
- **Scope:** Global, Session
- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.18](#) , [MariaDB 10.3.10](#)
- 

### Rocksdb\_bloom\_filter\_full\_true\_positive

- **Description:**
- **Scope:** Global, Session
- **Data Type:**

    numeric

- **Introduced:** [MariaDB 10.2.18](#) , [MariaDB 10.3.10](#)
- 

### Rocksdb\_bloom\_filter\_prefix\_checked

- **Description:** Number of times the Bloom Filter checked before creating an iterator on a file.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_bloom\_filter\_prefix\_useful

- **Description:** Number of times the Bloom Filter check used to avoid creating an iterator on a file.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_bloom\_filter\_useful

- **Description:** Number of times the Bloom Filter used instead of reading from file.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_bytes\_read

- **Description:** Total number of uncompressed bytes read from memtables, cache or table files.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_bytes\_written

- **Description:** Total number of uncompressed bytes written.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_compact\_read\_bytes

- **Description:** Number of bytes read during compaction.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_compact\_write\_bytes

- **Description:** Number of bytes written during compaction.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_compaction\_key\_drop\_new

- **Description:** Number of keys dropped during compaction due their being overwritten by new values.
- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Rocksdb\_compaction\_key\_drop\_obsolete

- **Description:** Number of keys dropped during compaction due to their being obsolete.

- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Rocksdb\_compaction\_key\_drop\_user

- **Description:** Number of keys dropped during compaction due to user compaction.

- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Rocksdb\_covered\_secondary\_key\_lookups

- **Description:** Incremented when avoiding reading a record via a keyread. This indicates lookups that were performed via a secondary index containing a field that is only a prefix of the **VARCHAR** column, and that could return all requested fields directly from the secondary index.

- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Rocksdb\_flush\_write\_bytes

- **Description:** Number of bytes written during flush.

- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Rocksdb\_get\_hit\_l0

- **Description:** Number of times reads got data from the L0 compaction layer.

- **Scope:** Global, Session

- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

#### Rocksdb\_get\_hit\_l1

- **Description:** Number of times reads got data from the L1 compaction layer.

- **Scope:** Global, Session

- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

`Rocksdb_get_hit_l2_and_up`

- **Description:** Number of times reads got data from the L2 and up compaction layer.
- **Scope:** Global, Session
- **Data Type:**  
    numeric

- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

`Rocksdb_getupdatessince_calls`

- **Description:** Number of calls to the  
    `GetUpdatesSince`  
    function. You may find this useful when monitoring refreshes of the transaction log.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

`Rocksdb_iter_bytes_read`

- **Description:** Total uncompressed bytes read from an iterator, including the size of both key and value.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

`Rocksdb_10_num_files_stall_micros`

- **Description:** Shows how long in microseconds throttled due to too many files in L0.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- **Removed:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

`Rocksdb_10_slowdown_micros`

- **Description:** Total time spent waiting in microseconds while performing L0-L1 compactions.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- **Removed:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

`Rocksdb_manual_compactions_processed`

- **Description:**
- **Scope:** Global, Session
- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.18](#) , [MariaDB 10.3.10](#)
- 

#### Rocksdb\_manual\_compactions\_running

- **Description:**
- **Scope:** Global, Session
- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.18](#) , [MariaDB 10.3.10](#)
- 

#### Rocksdb\_memtable\_compaction\_micros

- **Description:**
- **Scope:** Global, Session
- **Data Type:**

numeric

- **Removed:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

#### Rocksdb\_memtable\_hit

- **Description:** Number of memtable hits.
- **Scope:** Global, Session
- **Data Type:**

numeric

#### Rocksdb\_memtable\_miss

- **Description:** Number of memtable misses.
- **Scope:** Global, Session
- **Data Type:**

numeric

#### Rocksdb\_memtable\_total

- **Description:** Memory used, in bytes, of all memtables.
- **Scope:** Global, Session
- **Data Type:**

numeric

#### Rocksdb\_memtable\_unflushed

- **Description:** Memory used, in bytes, of all unflushed memtables.
- **Scope:** Global, Session
- **Data Type:**

numeric

---

### Rocksdb\_no\_file\_closes

- **Description:** Number of times files were closed.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_no\_file\_errors

- **Description:** Number of errors encountered while trying to read data from an SST file.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_no\_file\_opens

- **Description:** Number of times files were opened.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_num\_iterators

- **Description:** Number of iterators currently open.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_number\_block\_not\_compressed

- **Description:** Number of uncompressed blocks.
- **Scope:** Global, Session
- **Data Type:**

    numeric

---

### Rocksdb\_number\_db\_next

- **Description:** Number of next calls.
  - **Scope:** Global, Session
  - **Data Type:**
- numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
-

## Rocksdb\_number\_db\_next\_found

- **Description:** Number of next calls that returned data.
  - **Scope:** Global, Session
  - **Data Type:** numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

## Rocksdb\_number\_db\_prev

- **Description:** Number of prev calls.
  - **Scope:** Global, Session
  - **Data Type:** numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

## Rocksdb\_number\_db\_prev\_found

- **Description:** Number of prev calls that returned data.
  - **Scope:** Global, Session
  - **Data Type:** numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

## Rocksdb\_number\_db\_seek

- **Description:** Number of seek calls.
  - **Scope:** Global, Session
  - **Data Type:** numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

## Rocksdb\_number\_db\_seek\_found

- **Description:** Number of seek calls that returned data.
  - **Scope:** Global, Session
  - **Data Type:** numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
-

### Rocksdb\_number\_deletes\_filtered

- **Description:** Number of deleted records were not written to storage due to a nonexistent key.
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

### Rocksdb\_number\_keys\_read

- **Description:** Number of keys have been read.
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

### Rocksdb\_number\_keys\_updated

- **Description:** Number of keys have been updated.
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

### Rocksdb\_number\_keys\_written

- **Description:** Number of keys have been written.
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

### Rocksdb\_number\_merge\_failures

- **Description:** Number of failures encountered while performing merge operator actions.
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

### Rocksdb\_number\_multiget\_bytes\_read

- **Description:** Number of bytes read during RocksDB MultiGet()  
calls.
  - **Scope:** Global, Session
  - **Data Type:**  
numeric
- 

### Rocksdb\_number\_multiget\_get

- **Description:** Number of RocksDB

```
    MultiGet()  
    requests made.
```

- **Scope:** Global, Session

- **Data Type:**

```
    numeric
```

---

#### Rocksdb\_number\_multiget\_keys\_read

- **Description:** Number of keys read through RocksDB

```
    MultiGet()  
    calls.
```

- **Scope:** Global, Session

- **Data Type:**

```
    numeric
```

---

#### Rocksdb\_number\_reseeks\_iteration

- **Description:** Number of reseeks that have occurred inside an iteration that skipped over a large number of keys with the same user key.

- **Scope:** Global, Session

- **Data Type:**

```
    numeric
```

---

#### Rocksdb\_number\_sst\_entry\_delete

- **Description:** Number of delete markers written.

- **Scope:** Global, Session

- **Data Type:**

```
    numeric
```

---

#### Rocksdb\_number\_sst\_entry\_merge

- **Description:** Number of merge keys written.

- **Scope:** Global, Session

- **Data Type:**

```
    numeric
```

---

#### Rocksdb\_number\_sst\_entry\_other

- **Description:** Number of keys written that are not delete, merge or put keys.

- **Scope:** Global, Session

- **Data Type:**

```
    numeric
```

---

#### Rocksdb\_number\_sst\_entry\_put

- **Description:** Number of put keys written.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_number\_sst\_entry\_singledelete

- **Description:** Number of single-delete keys written.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_number\_superversion\_acquires

- **Description:** Number of times the superversion structure acquired. This is useful when tracking files for the database.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_number\_superversion\_cleanups

- **Description:** Number of times the superversion structure performed cleanups.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_number\_superversion\_releases

- **Description:** Number of times the superversion structure released.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_queries\_point

- **Description:** Number of single-row queries.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_queries\_range

- **Description:** Number of multi-row queries.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

## Rocksdb\_row\_lock\_deadlocks

- **Description:** Number of deadlocks.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

## Rocksdb\_row\_lock\_wait\_timeouts

- **Description:** Number of row lock wait timeouts.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

## Rocksdb\_rows\_deleted

- **Description:** Number of rows deleted.
- **Scope:** Global, Session
- **Data Type:**  
    numeric

## Rocksdb\_rows\_deleted\_blind

- **Description:**
- **Scope:** Global, Session
- **Data Type:**  
    numeric

## Rocksdb\_rows\_expired

- **Description:** Number of expired rows.
- **Scope:** Global, Session
- **Data Type:**  
    numeric

## Rocksdb\_rows\_filtered

- **Description:** Number of TTL filtered rows.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- **Introduced:** [MariaDB 10.2.15](#) , [MariaDB 10.3.7](#)
- 

## Rocksdb\_rows\_inserted

- **Description:** Number of rows inserted.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_rows\_read

- **Description:** Number of rows read.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_rows\_updated

- **Description:** Number of rows updated.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_snapshot\_conflict\_errors

- **Description:** Number of snapshot conflict errors that have occurred during transactions that forced a rollback.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_stall\_l0\_file\_count\_limit\_slowdowns

- **Description:** Write slowdowns due to L0 being near to full.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_stall\_l0\_file\_count\_limit\_stops

- **Description:** Write stops due to L0 being full.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_stall\_locked\_l0\_file\_count\_limit\_slowdowns

- **Description:** Write slowdowns due to L0 being near to full and L0 compaction in progress.

- **Scope:** Global, Session

- **Data Type:**

numeric

---

#### Rocksdb\_stall\_locked\_l0\_file\_count\_limit\_stops

- **Description:** Write stops due to L0 being full and L0 compaction in progress.
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### Rocksdb\_stall\_memtable\_limit\_slowdowns

- **Description:** Write slowdowns due to approaching maximum permitted number of memtables.
- **Scope:** Global, Session
- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.10](#) , [MariaDB 10.3.3](#)
- 

#### Rocksdb\_stall\_memtable\_limit\_stops

- **Description:** \* **Description:** Write stops due to reaching maximum permitted number of memtables.
- **Scope:** Global, Session
- **Data Type:**

numeric

- **Introduced:** [MariaDB 10.2.10](#) , [MariaDB 10.3.3](#)
- 

#### Rocksdb\_stall\_micros

- **Description:** Time in microseconds that the writer had to wait for the compaction or flush to complete.
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### Rocksdb\_stall\_pending\_compaction\_limit\_slowdowns

- **Description:** Write slowdowns due to nearing the limit for the maximum number of pending compaction bytes.
- **Scope:** Global, Session
- **Data Type:**

numeric

---

#### Rocksdb\_stall\_pending\_compaction\_limit\_stops

- **Description:** Write stops due to reaching the limit for the maximum number of pending compaction bytes.
- **Scope:** Global, Session
- **Data Type:**

numeric

---

### Rocksdb\_stall\_total\_slowdowns

- **Description:** Total number of write slowdowns.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_stall\_total\_stops

- **Description:** Total number of write stops.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_system\_rows\_deleted

- **Description:** Number of rows deleted from system tables.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_system\_rows\_inserted

- **Description:** Number of rows inserted into system tables.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_system\_rows\_read

- **Description:** Number of rows read from system tables.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_system\_rows\_updated

- **Description:** Number of rows updated for system tables.
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

### Rocksdb\_wal\_bytes

- **Description:** Number of bytes written to WAL.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_wal\_group\_syncs

- **Description:** Number of group commit WAL file syncs have occurred. This is provided by MyRocks and is not a view of a RocksDB counter.

Increased in

    rocksdb\_flush\_wal()  
    when doing the  
    rdb->FlushWAL()  
    call.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_wal\_synced

- **Description:** Number of syncs made on RocksDB WAL file.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_write\_other

- **Description:** Number of writes processed by a thread other than the requesting thread.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_write\_self

- **Description:** Number of writes processed by requesting thread.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_write\_timedout

- **Description:** Number of writes that timed out.

- **Scope:** Global, Session

- **Data Type:**

    numeric

---

#### Rocksdb\_write\_wal

- **Description:** Number of write calls that requested WAL.

- **Scope:** Global, Session

- **Data Type:**

numeric

## 4.3.14.6 MyRocks System Variables

### Contents

1. [rocksdb\\_access\\_hint\\_on\\_compaction\\_start](#)
2. [rocksdb\\_advise\\_random\\_on\\_open](#)
3. [rocksdb\\_allow\\_concurrent\\_memtable\\_write](#)
4. [rocksdb\\_allow\\_mmap\\_reads](#)
5. [rocksdb\\_allow\\_mmap\\_writes](#)
6. [rocksdb\\_allow\\_to\\_start\\_after\\_corruption](#)
7. [rocksdb\\_background\\_sync](#)
8. [rocksdb\\_base\\_background\\_compactions](#)
9. [rocksdb\\_blind\\_delete\\_primary\\_key](#)
10. [rocksdb\\_block\\_cache\\_size](#)
11. [rocksdb\\_block\\_restart\\_interval](#)
12. [rocksdb\\_block\\_size](#)
13. [rocksdb\\_block\\_size\\_deviation](#)
14. [rocksdb\\_bulk\\_load](#)
15. [rocksdb\\_bulk\\_load\\_allow\\_sk](#)
16. [rocksdb\\_bulk\\_load\\_allow\\_unsorted](#)
17. [rocksdb\\_bulk\\_load\\_size](#)
18. [rocksdb\\_bytes\\_per\\_sync](#)
19. [rocksdb\\_cache\\_dump](#)
20. [rocksdb\\_cache\\_high\\_pri\\_pool\\_ratio](#)
21. [rocksdb\\_cache\\_index\\_and\\_filter\\_blocks](#)
22. [rocksdb\\_cache\\_index\\_and\\_filter\\_with\\_high](#)
23. [rocksdb\\_checksums\\_pct](#)
24. [rocksdb\\_collect\\_sst\\_properties](#)
25. [rocksdb\\_commit\\_in\\_the\\_middle](#)
26. [rocksdb\\_commit\\_time\\_batch\\_for\\_recovery](#)
27. [rocksdb\\_compact\\_cf](#)
28. [rocksdb\\_compaction\\_readahead\\_size](#)
29. [rocksdb\\_compaction\\_sequential\\_deletes](#)
30. [rocksdb\\_compaction\\_sequential\\_deletes\\_co](#)
31. [rocksdb\\_compaction\\_sequential\\_deletes\\_fil](#)
32. [rocksdb\\_compaction\\_sequential\\_deletes\\_w](#)
33. [rocksdb\\_concurrent\\_prepare](#)
34. [rocksdb\\_create\\_checkpoint](#)
35. [rocksdb\\_create\\_if\\_missing](#)
36. [rocksdb\\_create\\_missing\\_column\\_families](#)
37. [rocksdb\\_datadir](#)
38. [rocksdb\\_db\\_write\\_buffer\\_size](#)
39. [rocksdb\\_deadlock\\_detect](#)
40. [rocksdb\\_deadlock\\_detect\\_depth](#)
41. [rocksdb\\_debug\\_manual\\_compaction\\_delay](#)
42. [rocksdb\\_debug\\_optimizer\\_no\\_zero\\_cardina](#)
43. [rocksdb\\_debug\\_ttl\\_ignore\\_pk](#)
44. [rocksdb\\_debug\\_ttl\\_read\\_filter\\_ts](#)
45. [rocksdb\\_debug\\_ttl\\_rec\\_ts](#)
46. [rocksdb\\_debug\\_ttl\\_snapshot\\_ts](#)
47. [rocksdb\\_default\\_cf\\_options](#)
48. [rocksdb\\_delayed\\_write\\_rate](#)
49. [rocksdb\\_delete\\_cf](#)
50. [rocksdb\\_delete\\_obsolete\\_files\\_period\\_micro](#)
51. [rocksdb\\_enable\\_2pc](#)
52. [rocksdb\\_enable\\_bulk\\_load\\_api](#)
53. [rocksdb\\_enable\\_insert\\_with\\_update\\_cachin](#)
54. [rocksdb\\_enable\\_thread\\_tracking](#)
55. [rocksdb\\_enable\\_ttl](#)
56. [rocksdb\\_enable\\_ttl\\_read\\_filtering](#)
57. [rocksdb\\_enable\\_write\\_thread\\_adaptive\\_yiel](#)
58. [rocksdb\\_error\\_if\\_exists](#)
59. [rocksdb\\_error\\_on\\_suboptimal\\_collation](#)
60. [rocksdb\\_flush\\_log\\_at\\_trx\\_commit](#)
61. [rocksdb\\_flush\\_memtable\\_on\\_analyze](#)
62. [rocksdb\\_force\\_compute\\_memtable\\_stats](#)
63. [rocksdb\\_force\\_compute\\_memtable\\_stats\\_c](#)

53. rocksdb\_force\_compute\_memtable\_stats\_c  
64. rocksdb\_force\_flush\_memtable\_and\_lzero  
65. rocksdb\_force\_flush\_memtable\_now  
66. rocksdb\_force\_index\_records\_in\_range  
67. rocksdb\_git\_hash  
68. rocksdb\_hash\_index\_allow\_collision  
69. rocksdb\_ignore\_unknown\_options  
70. rocksdb\_index\_type  
71. rocksdb\_info\_log\_level  
72. rocksdb\_io\_write\_timeout  
73. rocksdb\_is\_fd\_close\_on\_exec  
74. rocksdb\_keep\_log\_file\_num  
75. rocksdb\_large\_prefix  
76. rocksdb\_lock\_scanned\_rows  
77. rocksdb\_lock\_wait\_timeout  
78. rocksdb\_log\_file\_time\_to\_roll  
79. rocksdb\_manifest\_preallocation\_size  
80. rocksdb\_manual\_compaction\_threads  
81. rocksdb\_manual\_wal\_flush  
82. rocksdb\_master\_skip\_tx\_api  
83. rocksdb\_max\_background\_compactions  
84. rocksdb\_max\_backgroundFlushes  
85. rocksdb\_max\_backgroundJobs  
86. rocksdb\_max\_latest\_deadlocks  
87. rocksdb\_max\_log\_file\_size  
88. rocksdb\_max\_manifest\_file\_size  
89. rocksdb\_max\_manual\_compactions  
90. rocksdb\_max\_open\_files  
91. rocksdb\_max\_row\_locks  
92. rocksdb\_max\_subcompactions  
93. rocksdb\_max\_total\_wal\_size  
94. rocksdb\_merge\_buf\_size  
95. rocksdb\_merge\_combine\_read\_size  
96. rocksdb\_merge\_tmp\_file\_removal\_delay\_m  
97. rocksdb\_new\_table\_reader\_for\_compaction  
98. rocksdb\_no\_block\_cache  
99. rocksdb\_override\_cf\_options  
00. rocksdb\_paranoid\_checks  
01. rocksdb\_pause\_background\_work  
02. rocksdb\_perf\_context\_level  
03. rocksdb\_persistent\_cache\_path  
04. rocksdb\_persistent\_cache\_size\_mb  
05. rocksdb\_pin\_I0\_filter\_and\_index\_blocks\_in  
06. rocksdb\_print\_snapshot\_conflict\_queries  
07. rocksdb\_rate\_limiter\_bytes\_per\_sec  
08. rocksdb\_read\_free\_rpl\_tables  
09. rocksdb\_records\_in\_range  
10. rocksdb\_remove\_mariabackup\_checkpoint  
11. rocksdb\_reset\_stats  
12. rocksdb\_rollback\_on\_timeout  
13. rocksdb\_seconds\_between\_stat\_computes  
14. rocksdb\_signal\_drop\_index\_thread  
15. rocksdb\_sim\_cache\_size  
16. rocksdb\_skip\_bloom\_filter\_on\_read  
17. rocksdb\_skip\_fill\_cache  
18. rocksdb\_skip\_unique\_check\_tables  
19. rocksdb\_sst\_mgr\_rate\_bytes\_per\_sec  
20. rocksdb\_stats\_dump\_period\_sec  
21. rocksdb\_stats\_level  
22. rocksdb\_stats\_recalc\_rate  
23. rocksdb\_store\_row\_debug\_checksums  
24. rocksdb\_strict\_collation\_check  
25. rocksdb\_strict\_collation\_exceptions  
26. rocksdb\_supported\_compression\_types  
27. rocksdb\_table\_cache\_numshardbits  
28. rocksdb\_table\_stats\_sampling\_pct  
29. rocksdb\_tmpdir  
30. rocksdb\_trace\_sst\_api  
31. rocksdb\_two\_write\_queues  
32. rocksdb\_unsafe\_for\_binlog  
33. rocksdb\_update\_cf\_options  
34. rocksdb\_use\_adaptive\_mutex  
35. rocksdb\_use\_clock\_cache

- 36. `rocksdb_use_direct_io_for_flush_and_compaction`
- 37. `rocksdb_use_direct_reads`
- 38. `rocksdb_use_direct_writes`
- 39. `rocksdb_use_fsync`
- 40. `rocksdb_validate_tables`
- 41. `rocksdb_verify_row_debug_checksums`
- 42. `rocksdb_wal_bytes_per_sync`
- 43. `rocksdb_wal_dir`
- 44. `rocksdb_wal_recovery_mode`
- 45. `rocksdb_wal_size_limit_mb`
- 46. `rocksdb_wal_ttl_seconds`
- 47. `rocksdb_whole_key_filtering`
- 48. `rocksdb_write_batch_max_bytes`
- 49. `rocksdb_write_disable_wal`
- 50. `rocksdb_write_ignore_missing_column_family`
- 51. `rocksdb_write_policy`

This page documents system variables related to the [MyRocks](#) storage engine. See [Server System Variables](#) for a complete list of system variables and instructions on setting them.

See also the [Full list of MariaDB options, system and status variables](#).

### `rocksdb_access_hint_on_compaction_start`

- **Description:** DBOptions::access\_hint\_on\_compaction\_start for RocksDB. Specifies the file access pattern, applied to all input files, once a compaction starts.
- **Commandline:**  
`--rocksdb-access-hint-on-compaction-start=`
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric
- **Default Value:**  
1
- **Range:**  
0  
to  
3

---

### `rocksdb_advise_random_on_open`

- **Description:** DBOptions::advise\_random\_on\_open for RocksDB.
- **Commandline:**  
`--rocksdb-advise-random-on-open={0|1}`
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean
- **Default Value:**  
ON

---

### `rocksdb_allow_concurrent_memtable_write`

- **Description:** DBOptions::allow\_concurrent\_memtable\_write for RocksDB.
- **Commandline:**  
`--rocksdb-allow-concurrent-memtable-write={0|1}`

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

rocksdb\_allow\_mmap\_reads

- **Description:** DBOptions::allow\_mmap\_reads for RocksDB
- **Commandline:**

--rocksdb-allow-mmap-reads={0|1}

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

rocksdb\_allow\_mmap\_writes

- **Description:** DBOptions::allow\_mmap\_writes for RocksDB
- **Commandline:**

--rocksdb-allow-mmap-writes={0|1}

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

rocksdb\_allow\_to\_start\_after\_corruption

- **Description:** Allow server still to start successfully even if RocksDB corruption is detected.
- **Commandline:**

--rocksdb-allow-to-start-after-corruption={0|1}

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
OFF

- **Introduced:** [MariaDB 10.3.7](#) , [MariaDB 10.2.15](#)
- 

rocksdb\_background\_sync

- **Description:** Turns on background syncs for RocksDB
- **Commandline:**

```
--rocksdb-background-sync={0|1}
```

- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
    boolean
  - **Default Value:**  
    OFF
  - **Removed:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

### rocksdb\_base\_background\_compactions

- **Description:** DBOptions::base\_background\_compactions for RocksDB
  - **Commandline:**  
    --rocksdb-base-background-compactions=#
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
    numeric
  - **Default Value:**  
    1
  - **Range:**  
    -1  
    to  
    64
  - **Removed:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

### rocksdb\_blind\_delete\_primary\_key

- **Description:** Deleting rows by primary key lookup, without reading rows (Blind Deletes). Blind delete is disabled if the table has secondary key.
  - **Commandline:**  
    --rocksdb-blind-delete-primary-key={0|1}
  - **Scope:** Global, Session
  - **Dynamic:** Yes
  - **Data Type:**  
    boolean
  - **Default Value:**  
    OFF
- 

### rocksdb\_block\_cache\_size

- **Description:** Block\_cache size for RocksDB (block size 1024)
  - **Commandline:**  
    --rocksdb-block-cache-size=#
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
    numeric
  - **Default Value:**  
    536870912
-

- **Range:**

```
1024
to
9223372036854775807
```

To see the statistics of block cache usage, check

```
SHOW ENGINE ROCKSDB STATUS
output (search for lines starting with
rocksdb.block.cache
).
```

One can check the size of data of the block cache in

```
DB_BLOCK_CACHE_USAGE
column of the
INFORMATION_SCHEMA.ROCKSDB_DBSTATS
table.
```

---

### rocksdb\_block\_restart\_interval

- **Description:** BlockBasedTableOptions::block\_restart\_interval for RocksDB

- **Commandline:**

```
--rocksdb-block-restart-interval=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
numeric
```

- **Default Value:**

```
16
```

- **Range:**

```
1
to
2147483647
```

---

### rocksdb\_block\_size

- **Description:** BlockBasedTableOptions::block\_size for RocksDB

- **Commandline:**

```
--rocksdb-block-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
numeric
```

- **Default Value:**

```
4096
```

- **Range:**

```
1
to
18446744073709551615
```

---

### rocksdb\_block\_size\_deviation

- **Description:** BlockBasedTableOptions::block\_size\_deviation for RocksDB

- **Commandline:**

```
--rocksdb-block-size-deviation=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    10

- **Range:**

    0

    to

    2147483647

---

### rocksdb\_bulk\_load

- **Description:** Use bulk-load mode for inserts. This disables unique\_checks and enables rocksdb\_commit\_in\_the\_middle.

- **Commandline:**

    --rocksdb-bulk-load={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

---

### rocksdb\_bulk\_load\_allow\_sk

- **Description:** Allow bulk loading of sk keys during bulk-load. Can be changed only when bulk load is disabled.

- **Commandline:**

    --rocksdb-bulk-load\_allow\_sk={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

- **Introduced:** [MariaDB 10.3.10](#) , [MariaDB 10.2.18](#)
- 

### rocksdb\_bulk\_load\_allow\_unsorted

- **Description:** Allow unsorted input during bulk-load. Can be changed only when bulk load is disabled.

- **Commandline:**

    --rocksdb-bulk-load\_allow\_unsorted={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

---

## rocksdb\_bulk\_load\_size

- **Description:** Maximum number of records in a batch for bulk-load mode.

- **Commandline:**

--rocksdb-bulk-load-size=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1000

- **Range:**

1

to

1073741824

---

## rocksdb\_bytes\_per\_sync

- **Description:** DBOptions::bytes\_per\_sync for RocksDB.

- **Commandline:**

--rocksdb-bytes-per-sync=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

18446744073709551615

---

## rocksdb\_cache\_dump

- **Description:** Include RocksDB block cache content in core dump.

- **Commandline:**

--rocksdb-cache-dump={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

- **Introduced:** [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#)
- 

## rocksdb\_cache\_high\_pri\_pool\_ratio

- **Description:** Specify the size of block cache high-pri pool.

- **Commandline:**

```
--rocksdb-cache-high-pri-pool-ratio=#
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
double

- **Default Value:**  
0.000000

- **Range:**  
0  
to  
1

- **Introduced:** [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#)
- 

### rocksdb\_cache\_index\_and\_filter\_blocks

- **Description:** BlockBasedTableOptions::cache\_index\_and\_filter\_blocks for RocksDB.
- **Commandline:**

```
--rocksdb-cache-index-and-filter-blocks={0|1}
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
ON

### rocksdb\_cache\_index\_and\_filter\_with\_high\_priority

- **Description:** cache\_index\_and\_filter\_blocks\_with\_high\_priority for RocksDB.
- **Commandline:**

```
--rocksdb-cache-index-and-filter-with-high-priority={0|1}
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
ON

- **Introduced:** [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#)
- 

### rocksdb\_checksums\_pct

- **Description:** Percentage of rows to be checksummed.
- **Commandline:**

```
--rocksdb-checksums-pct=#
```

- **Scope:** Global, Session
- **Dynamic:** Yes
- **Data Type:**  
numeric

- **Default Value:**  
100

- **Range:**

0  
to  
100

---

### rocksdb\_collect\_sst\_properties

- **Description:** Enables collecting SST file properties on each flush.

- **Commandline:**

--rocksdb-collect-sst-properties={0|1}

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

---

### rocksdb\_commit\_in\_the\_middle

- **Description:** Commit rows implicitly every rocksdb\_bulk\_load\_size, on bulk load/insert, update and delete.

- **Commandline:**

--rocksdb-commit-in-the-middle={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### rocksdb\_commit\_time\_batch\_for\_recovery

- **Description:** TransactionOptions::commit\_time\_batch\_for\_recovery for RocksDB.

- **Commandline:**

--rocksdb-commit-time-batch-for-recovery={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** [MariaDB 10.3.10](#) , [MariaDB 10.2.18](#)
- 

### rocksdb\_compact\_cf

- **Description:** Compact column family.

- **Commandline:**

--rocksdb-compact-cf=value

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:** (Empty)
- 

### rocksdb\_compaction\_readahead\_size

- **Description:** DBOptions::compaction\_readahead\_size for RocksDB.

- **Commandline:**

--rocksdb-compaction-readahead-size=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

18446744073709551615

---

### rocksdb\_compaction\_sequential\_deletes

- **Description:** RocksDB will trigger compaction for the file if it has more than this number sequential deletes per window.

- **Commandline:**

--rocksdb-compaction-sequential-deletes=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

2000000

---

### rocksdb\_compaction\_sequential\_deletes\_count\_sd

- **Description:** Counting SingleDelete as rocksdb\_compaction\_sequential\_deletes.

- **Commandline:**

--rocksdb-compaction-sequential-deletes-count-sd={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

## `rocksdb_compaction_sequential_deletes_file_size`

- **Description:** Minimum file size required for compaction\_sequential\_deletes.

- **Commandline:**

```
--rocksdb-compaction-sequential-deletes-file-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    0

- **Range:**

    -1

    to

    9223372036854775807

## `rocksdb_compaction_sequential_deletes_window`

- **Description:** Size of the window for counting rocksdb\_compaction\_sequential\_deletes.

- **Commandline:**

```
--rocksdb-compaction-sequential-deletes-window=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    0

- **Range:**

    0

    to

    2000000

## `rocksdb_concurrent_prepare`

- **Description:** DBOptions::concurrent\_prepare for RocksDB.

- **Commandline:**

```
--rocksdb-coconcurrent-prepare={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    boolean

- **Default Value:**

    1

- **Removed:** [MariaDB 10.3.7](#) , [MariaDB 10.2.15](#)

## `rocksdb_create_checkpoint`

- **Description:** Checkpoint directory.

- **Commandline:**

```
--rocksdb-create-checkpoint=value
```

- **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    string
  - **Default Value:** (Empty)
- 

rocksdb\_create\_if\_missing

- **Description:** DBOptions::create\_if\_missing for RocksDB.

- **Commandline:**

```
--rocksdb-create-if-missing={0|1}
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    boolean

- **Default Value:**

ON

---

rocksdb\_create\_missing\_column\_families

- **Description:** DBOptions::create\_missing\_column\_families for RocksDB.

- **Commandline:**

```
--rocksdb-create-missing-column-families={0|1}
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    boolean

- **Default Value:**

OFF

---

rocksdb\_datadir

- **Description:** RocksDB data directory.

- **Commandline:**

```
--rocksdb-datadir[=value]
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    string

- **Default Value:**

./#rocksdb

---

rocksdb\_db\_write\_buffer\_size

- **Description:** DBOptions::db\_write\_buffer\_size for RocksDB.

- **Commandline:**

```
--rocksdb-db-write-buffer-size=
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**

0

- **Range:**

0

to

18446744073709551615

### rocksdb\_deadlock\_detect

- **Description:** Enables deadlock detection.

- **Commandline:**

--rocksdb-deadlock-detect={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

### rocksdb\_deadlock\_detect\_depth

- **Description:** Number of transactions deadlock detection will traverse through before assuming deadlock.

- **Commandline:**

--rocksdb-deadlock-detect-depth=#

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

50

- **Range:**

2

to

18446744073709551615

### rocksdb\_debug\_manual\_compaction\_delay

- **Description:** For debugging purposes only. Sleeping specified seconds for simulating long running compactions.

- **Commandline:**

--rocksdb-debug\_manual\_compaction\_delay=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0  
to  
4294967295

- **Introduced:** [MariaDB 10.3.10](#) , [MariaDB 10.2.18](#)
- 

### rocksdb\_debug\_optimizer\_no\_zero\_cardinality

- **Description:** If cardinality is zero, override it with some value.

- **Commandline:**

--rocksdb-debug-optimizer-no-zero-cardinality={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

### rocksdb\_debug\_ttl\_ignore\_pk

- **Description:** For debugging purposes only. If true, compaction filtering will not occur on PK TTL data. This variable is a no-op in non-debug builds.
- **Commandline:**

--rocksdb-debug-ttl-ignore-pk={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

### rocksdb\_debug\_ttl\_read\_filter\_ts

- **Description:** For debugging purposes only. Overrides the TTL read filtering time to time + debug\_ttl\_read\_filter\_ts. A value of 0 denotes that the variable is not set. This variable is a no-op in non-debug builds.

- **Commandline:**

--rocksdb-debug-ttl-read-filter-ts=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

-3600  
to  
3600

- **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
-

## `rocksdb_debug_ttl_rec_ts`

- **Description:** For debugging purposes only. Overrides the TTL of records to now() + debug\_ttl\_rec\_ts. The value can be +/- to simulate a record inserted in the past vs a record inserted in the 'future'. A value of 0 denotes that the variable is not set. This variable is a no-op in non-debug builds.
  - **Commandline:**  
  `--rocksdb-debug-ttl-read-filter-ts=#`
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
  numeric
  - **Default Value:**  
  0
  - **Range:**  
  -3600  
  to  
  3600
  - **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

## `rocksdb_debug_ttl_snapshot_ts`

- **Description:** For debugging purposes only. Sets the snapshot during compaction to now() + debug\_set\_ttl\_snapshot\_ts. The value can be positive or negative to simulate a snapshot in the past vs a snapshot created in the 'future'. A value of 0 denotes that the variable is not set. This variable is a no-op in non-debug builds.
  - **Commandline:**  
  `--rocksdb-debug-ttl-snapshot-ts=#`
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
  numeric
  - **Default Value:**  
  0
  - **Range:**  
  -3600  
  to  
  3600
  - **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

## `rocksdb_default_cf_options`

- **Description:** Default cf options for RocksDB.
  - **Commandline:**  
  `--rocksdb-default-cf-options=value`
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
  string
  - **Default Value:** (Empty)
- 

## `rocksdb_delayed_write_rate`

- **Description:** DBOptions::delayed\_write\_rate.

- **Commandline:**  
--rocksdb-delayed-write-rate=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    0  
    (Previously  
    16777216  
    )  
    )

- **Range:**

    0  
    to  
    18446744073709551615

---

### rocksdb\_delete\_cf

- **Description:** Delete column family.

- **Commandline:**

    --rocksdb-delete-cf=val

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    string

- **Default Value:** (Empty string)

- **Introduced:** [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#)
- 

### rocksdb\_delete\_obsolete\_files\_period\_micros

- **Description:** DBOptions::deleteObsoleteFilesPeriodMicros for RocksDB.

- **Commandline:**

    --rocksdb-delete-obsolete-files-period-micros=#

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    21600000000

- **Range:**

    0  
    to  
    9223372036854775807

---

### rocksdb\_enable\_2pc

- **Description:** Enable two phase commit for MyRocks. When set, MyRocks will keep its data consistent with the [binary log](#) (in other words, the server will be a crash-safe master). The consistency is achieved by doing two-phase XA commit with the binary log.

- **Commandline:**

    --rocksdb-enable-2pc={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

---

### rocksdb\_enable\_bulk\_load\_api

- **Description:** Enables using SstFileWriter for bulk loading.

- **Commandline:**

--rocksdb-enable-bulk-load-api={0|1}

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

---

### rocksdb\_enable\_insert\_with\_update\_caching

- **Description:** Whether to enable optimization where we cache the read from a failed insertion attempt in [INSERT ON DUPLICATE KEY UPDATE](#).

- **Commandline:**

--rocksdb-enable-insert-with-update-caching={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

- **Introduced:** [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#)

---

### rocksdb\_enable\_thread\_tracking

- **Description:** DBOptions::enable\_thread\_tracking for RocksDB.

- **Commandline:**

--rocksdb-enable-thread-tracking={0|1}

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### rocksdb\_enable\_ttl

- **Description:** Enable expired TTL records to be dropped during compaction.

- **Commandline:**

--rocksdb-enable-ttl={0|1}

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**  
ON

- **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

#### `rocksdb_enable_ttl_read_filtering`

- **Description:** For tables with TTL, expired records are skipped/filtered out during processing and in query results. Disabling this will allow these records to be seen, but as a result rows may disappear in the middle of transactions as they are dropped during compaction. Use with caution.

- **Commandline:**  
`--rocksdb-enable-ttl-read-filtering={0|1}`

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**  
ON

- **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

#### `rocksdb_enable_write_thread_adaptive_yield`

- **Description:** DBOptions::enable\_write\_thread\_adaptive\_yield for RocksDB.
- **Commandline:**  
`--rocksdb-enable-write-thread-adaptive-yield={0|1}`

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

#### `rocksdb_error_if_exists`

- **Description:** DBOptions::error\_if\_exists for RocksDBB.
- **Commandline:**  
`--rocksdb-error-if-exists={0|1}`

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

#### `rocksdb_error_on_suboptimal_collation`

- **Description:** Raise an error instead of warning if a sub-optimal collation is used.

- **Commandline:**

```
--rocksdb-error-on-suboptimal-collation={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

- **Introduced:** [MariaDB 10.3.10](#) , [MariaDB 10.2.18](#)
- 

### rocksdb\_flush\_log\_at\_trx\_commit

- **Description:** Sync on transaction commit. Similar to [innodb\\_flush\\_log\\_at\\_trx\\_commit](#). One can check the flushing by examining the [rocksdb\\_wal\\_synced](#) and [rocksdb\\_wal\\_bytes](#) status variables.

- 1 : Always sync on commit (the default).
- 0 : Never sync.
- 2 : Sync based on a timer controlled via rocksdb-background-sync.

- **Commandline:**

```
--rocksdb-flush-log-at-trx-commit=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0

to

2

---

### rocksdb\_flush\_memtable\_on\_analyze

- **Description:** Forces memtable flush on ANALYZE table to get accurate cardinality.

- **Commandline:**

```
--rocksdb-flush-memtable-on-analyze={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

- **Removed:** [MariaDB 10.3.7](#) , [MariaDB 10.2.15](#)
- 

### rocksdb\_force\_compute\_memtable\_stats

- **Description:** Force to always compute memtable stats.

- **Commandline:**

```
--rocksdb-force-compute-memtable-stats={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

---

### rocksdb\_force\_compute\_memtable\_stats\_cachetime

- **Description:** Time in usecs to cache memtable estimates.

- **Commandline:**

--rocksdb-force-compute-memtable-stats-cachetime=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

60000000

- **Range:**

0

to

2147483647

---

### rocksdb\_force\_flush\_memtable\_and\_lzero\_now

- **Description:** Acts similar to force\_flush\_memtable\_now, but also compacts all L0 files.

- **Commandline:**

--rocksdb-force-flush-memtable-and-lzero-now={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

- **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

### rocksdb\_force\_flush\_memtable\_now

- **Description:** Forces memstore flush which may block all write requests so be careful.

- **Commandline:**

--rocksdb-force-flush-memtable-now={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### rocksdb\_force\_index\_records\_in\_range

- **Description:** Used to override the result of records\_in\_range() when **FORCE INDEX** is used.

- **Commandline:**

```
--rocksdb-force-index-records-in-range=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

2147483647

## rocksdb\_git\_hash

- **Description:** Git revision of the RocksDB library used by MyRocks.

- **Commandline:**

```
--rocksdb-git-hash=value=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

- **Default Value:** As per git revision.

## rocksdb\_hash\_index\_allow\_collision

- **Description:** BlockBasedTableOptions::hash\_index\_allow\_collision for RocksDB.

- **Commandline:**

```
--rocksdb-hash-index-allow-collision={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

## rocksdb\_ignore\_unknown\_options

- **Description:** Enable ignoring unknown options passed to RocksDB.

- **Commandline:**

```
--rocksdb-ignore-unknown-options={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

- **Introduced:** MariaDB 10.3.7 , MariaDB 10.2.15

## rocksdb\_index\_type

- **Description:** BlockBasedTableOptions::index\_type for RocksDB.

- **Commandline:**

```
--rocksdb-index-type=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

enum

- **Default Value:**

```
kBinarySearch
```

- **Valid Values:**

```
kBinarySearch
```

,

```
kHashSearch
```

## rocksdb\_info\_log\_level

- **Description:** Filter level for info logs to be written mysqld error log. Valid values include 'debug\_level', 'info\_level', 'warn\_level', 'error\_level' and 'fatal\_level'.

- **Commandline:**

```
--rocksdb-info-log-level=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

```
error_level
```

- **Valid Values:**

```
error_level
```

,

```
debug_level
```

,

```
info_level
```

,

```
warn_level
```

,

```
fatal_level
```

## rocksdb\_io\_write\_timeout

- **Description:** Timeout for experimental I/O watchdog.

- **Commandline:**

```
--rocksdb-io-write-timeout=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Valid Values:**

0

to

- **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

### `rocksdb_is_fd_close_on_exec`

- **Description:** DBOptions::is\_fd\_close\_on\_exec for RocksDB.

- **Commandline:**

```
--rocksdb-is-fd-close-on-exec={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

### `rocksdb_keep_log_file_num`

- **Description:** DBOptions::keep\_log\_file\_num for RocksDB.

- **Commandline:**

```
--rocksdb-keep-log-file-num=
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

1000

- **Range:**

0

to

18446744073709551615

### `rocksdb_large_prefix`

- **Description:** Support large index prefix length of 3072 bytes. If off, the maximum index prefix length is 767.

- **Commandline:**

```
--rocksdb-large_prefix={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

### `rocksdb_lock_scanned_rows`

- **Description:** Take and hold locks on rows that are scanned but not updated.

- **Commandline:**

```
--rocksdb-lock-scanned-rows={0|1}
```

- **Scope:** Global, Session
- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

#### rocksdb\_lock\_wait\_timeout

- **Description:** Number of seconds to wait for lock.
- **Commandline:**  
--rocksdb-lock-wait-timeout=#

- **Scope:** Global, Session
- **Dynamic:** Yes
- **Data Type:**  
numeric

- **Default Value:**  
1

- **Range:**  
1  
to  
1073741824
- 

#### rocksdb\_log\_file\_time\_to\_roll

- **Description:** DBOptions::log\_file\_time\_to\_roll for RocksDB.
- **Commandline:**  
--rocksdb-log-file-time-to\_roll=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**  
0

- **Range:**  
0  
to  
18446744073709551615
- 

#### rocksdb\_manifest\_preallocation\_size

- **Description:** DBOptions::manifest\_preallocation\_size for RocksDB.

- **Commandline:**  
--rocksdb-manifest-preallocation-size=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**

- **Range:**

```
0
to
18446744073709551615
```

---

### `rocksdb_manual_compaction_threads`

- **Description:** How many rocksdb threads to run for manual compactions.

- **Commandline:**

```
--rocksdb-manual-compation-threads=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
numeric
```

- **Default Value:**

```
0
```

- **Range:**

```
0
to
128
```

- **Introduced:** [MariaDB 10.3.10](#) , [MariaDB 10.2.18](#)
- 

### `rocksdb_manual_wal_flush`

- **Description:** DBOptions::manual\_wal\_flush for RocksDB.

- **Commandline:**

```
--rocksdb-manual-wal-flush={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
boolean
```

- **Default Value:**

```
ON
```

---

### `rocksdb_master_skip_tx_api`

- **Description:** Skipping holding any lock on row access. Not effective on slave.

- **Commandline:**

```
--rocksdb-master-skip-tx-api={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
boolean
```

- **Default Value:**

```
OFF
```

---

### `rocksdb_max_background_compactions`

- **Description:** DBOptions::max\_background\_compactions for RocksDB.

- **Commandline:**

```
--rocksdb-max-background-compactions=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    1

- **Range:**

    1

    to

    64

- **Removed:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

## rocksdb\_max\_background\_flushes

- **Description:** DBOptions::max\_background\_flushes for RocksDB.

- **Commandline:**

```
--rocksdb-max-background-flushes=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    1

- **Range:**

    1

    to

    64

- **Removed:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

## rocksdb\_max\_background\_jobs

- **Description:** DBOptions::max\_background\_jobs for RocksDB.

- **Commandline:**

```
--rocksdb-max-background-jobs=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    2

- **Range:**

    -1

    to

    64

- **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
-

## `rocksdb_max_latest_deadlocks`

- **Description:** Maximum number of recent deadlocks to store.

- **Commandline:**

```
--rocksdb-max-latest-deadlocks=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    5

- **Range:**

    0

    to

    4294967295

## `rocksdb_max_log_file_size`

- **Description:** DBOptions::max\_log\_file\_size for RocksDB.

- **Commandline:**

```
--rocksdb-max-log-file-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    0

- **Range:**

    0

    to

    18446744073709551615

## `rocksdb_max_manifest_file_size`

- **Description:** DBOptions::max\_manifest\_file\_size for RocksDB.

- **Commandline:**

```
--rocksdb-manifest-log-file-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    1073741824

- **Range:**

    0

    to

    18446744073709551615

## `rocksdb_max_manual_compactions`

- **Description:** Maximum number of pending + ongoing number of manual compactions..

- **Commandline:**

```
--rocksdb-manual_compactions=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    10

- **Range:**

    0

    to

    4294967295

- **Introduced:** [MariaDB 10.3.10](#) , [MariaDB 10.2.18](#)
- 

## rocksdb\_max\_open\_files

- **Description:** DBOptions::max\_open\_files for RocksDB.

- **Commandline:**

```
--rocksdb-max-open-files=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    -2

- **Range:**

    -2

    to

    2147483647

---

## rocksdb\_max\_row\_locks

- **Description:** Maximum number of locks a transaction can have.

- **Commandline:**

```
--rocksdb-max-row-locks=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    1048576

- **Range:**

    ◦

        1

        to

        1073741824

        (>= [MariaDB 10.3.10](#) , [MariaDB 10.2.18](#) )

    ◦

        1

        to

        1048576

        (<= [MariaDB 10.3.9](#) , [MariaDB 10.2.17](#) )

---

## rocksdb\_max\_subcompactions

- **Description:** DBOptions::max\_subcompactions for RocksDB.

- **Commandline:**

```
--rocksdb-max-subcompactions=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    1

- **Range:**

    1

    to

    64

## rocksdb\_max\_total\_wal\_size

- **Description:** DBOptions::max\_total\_wal\_size for RocksDB. The maximum size limit for write-ahead-log files. Once this limit is reached, RocksDB forces the flushing of memtables.

- **Commandline:**

```
--rocksdb-max-total-wal-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    numeric

- **Default Value:**

    0

- **Range:**

    0

    to

    9223372036854775807

## rocksdb\_merge\_buf\_size

- **Description:** Size to allocate for merge sort buffers written out to disk during inplace index creation.

- **Commandline:**

```
--rocksdb-merge-buf-size=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    67108864

- **Range:**

    100

    to

    18446744073709551615

## `rocksdb_merge_combine_read_size`

- **Description:** Size that we have to work with during combine (reading from disk) phase of external sort during fast index creation.

- **Commandline:**

```
--rocksdb-merge-combine-read-size=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    1073741824

- **Range:**

    100

    to

    18446744073709551615

## `rocksdb_merge_tmp_file_removal_delay_ms`

- **Description:** Fast index creation creates a large tmp file on disk during index creation. Removing this large file all at once when index creation is complete can cause trim stalls on Flash. This variable specifies a duration to sleep (in milliseconds) between calling chsize() to truncate the file in chunks. The chunk size is the same as merge\_buf\_size.

- **Commandline:**

```
--rocksdb-merge-tmp-file-removal-delay-ms=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    0

- **Range:**

    0

    to

    18446744073709551615

## `rocksdb_new_table_reader_for_compaction_inputs`

- **Description:** DBOptions::new\_table\_reader\_for\_compaction\_inputs for RocksDB.

- **Commandline:**

```
--rocksdb-new-table-reader-for-compaction-inputs={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    boolean

- **Default Value:**

    OFF

## `rocksdb_no_block_cache`

- **Description:** BlockBasedTableOptions::no\_block\_cache for RocksDB.

- **Commandline:**

```
--rocksdb-no-block-cache={0|1}
```

- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
- 

### rocksdb\_override\_cf\_options

- **Description:** Option overrides per cf for RocksDB. Note that the

```
rocksdb-override-cf-options
```

syntax is quite strict, and any typos will result in a parse error, and the MyRocks plugin will not be loaded. Depending on your configuration, the server may still start. If it does start, you can use this command to check if the plugin is loaded:

```
select * from information_schema.plugins where plugin_name='ROCKSDB'
```

(note that you need the "ROCKSDB" plugin. Other auxiliary plugins like "ROCKSDB\_TRX" might still get loaded). Another way is to detect the error is check the error log.

- **Commandline:**  
`--rocksdb-override-cf-options=value`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
string
```

- **Default Value:** (Empty)
- 

### rocksdb\_paranoid\_checks

- **Description:** DBOptions::paranoid\_checks for RocksDB.

- **Commandline:**

```
--rocksdb-paranoid-checks={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
boolean
```

- **Default Value:**

```
ON
```

---

### rocksdb\_pause\_background\_work

- **Description:** Disable all rocksdb background operations.

- **Commandline:**

```
--rocksdb-pause-background-work={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
boolean
```

- **Default Value:**

```
OFF
```

---

## rocksdb\_perf\_context\_level

- **Description:** Perf Context Level for rocksdb internal timer stat collection.

- **Commandline:**

```
--rocksdb-perf-context-level=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    0
```

- **Range:**

```
    0
```

```
    to
```

```
    5
```

## rocksdb\_persistent\_cache\_path

- **Description:** Path for BlockBasedTableOptions::persistent\_cache for RocksDB.

- **Commandline:**

```
--rocksdb-persistent-cache-path=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
    string
```

- **Default Value:** (Empty)

## rocksdb\_persistent\_cache\_size\_mb

- **Description:** Size of cache in MB for BlockBasedTableOptions::persistent\_cache for RocksDB.

- **Commandline:**

```
--rocksdb-persistent-cache-size-mb=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
    numeric
```

- **Default Value:**

```
    0
```

- **Range:**

```
    0
```

```
    to
```

```
    18446744073709551615
```

## rocksdb\_pin\_10\_filter\_and\_index\_blocks\_in\_cache

- **Description:** pin\_10\_filter\_and\_index\_blocks\_in\_cache for RocksDB.

- **Commandline:**

```
--rocksdb-pin-10-filter-and-index-blocks-in-cache={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

---

### rocksdb\_print\_snapshot\_conflict\_queries

- **Description:** Logging queries that got snapshot conflict errors into \*.err log.

- **Commandline:**

--rocksdb-print-snapshot-conflict-queries={0|1}

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### rocksdb\_rate\_limiter\_bytes\_per\_sec

- **Description:** DBOptions::rate\_limiter bytes\_per\_sec for RocksDB.

- **Commandline:**

--rocksdb-rate-limiter-bytes-per-sec=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

9223372036854775807

---

### rocksdb\_read\_free\_rpl\_tables

- **Description:** List of tables that will use read-free replication on the slave (i.e. not lookup a row during replication).

- **Commandline:**

--rocksdb-read-free-rpl-tables=value

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:** (Empty)

- **Removed:** [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#)

---

### rocksdb\_records\_in\_range

- **Description:** Used to override the result of records\_in\_range(). Set to a positive number to override.

- **Commandline:**

```
--rocksdb-records-in-range=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    0

- **Range:**

    0

    to

    2147483647

---

### rocksdb\_remove\_mariabackup\_checkpoint

- **Description:** Remove [mariabackup](#) checkpoint.

- **Commandline:**

```
--rocksdb-remove-mariabackup-checkpoint={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

- **Introduced:** [MariaDB 10.3.8](#) , [MariaDB 10.2.16](#)

---

### rocksdb\_reset\_stats

- **Description:** Reset the RocksDB internal statistics without restarting the DB.

- **Commandline:**

```
--rocksdb-reset-stats={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

- **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)

---

### rocksdb\_rollback\_on\_timeout

- **Description:** Whether to roll back the complete transaction or a single statement on lock wait timeout (a single statement by default).

- **Commandline:**

```
--rocksdb-rollback-on-timeout={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

- **Introduced:** MariaDB 10.4.7 , MariaDB 10.3.17 , MariaDB 10.2.26
- 

### rocksdb\_seconds\_between\_stat\_computes

- **Description:** Sets a number of seconds to wait between optimizer stats recomputation. Only changed indexes will be refreshed.
- **Commandline:**

```
--rocksdb-seconds-between-stat-computes=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

3600

- **Range:**

0

to

4294967295

---

### rocksdb\_signal\_drop\_index\_thread

- **Description:** Wake up drop index thread.

- **Commandline:**

```
--rocksdb-signal-drop-index-thread={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### rocksdb\_sim\_cache\_size

- **Description:** Simulated cache size for RocksDB.

- **Commandline:**

```
--rocksdb-sim-cache-size=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

9223372036854775807

---

### rocksdb\_skip\_bloom\_filter\_on\_read

- **Description:** Skip using bloom filter for reads.

- **Commandline:**

```
--rocksdb-skip-bloom-filter-on-read={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### rocksdb\_skip\_fill\_cache

- **Description:** Skip filling block cache on read requests.

- **Commandline:**

```
--rocksdb-skip-fill-cache={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### rocksdb\_skip\_unique\_check\_tables

- **Description:** Skip unique constraint checking for the specified tables.

- **Commandline:**

```
--rocksdb-skip-unique-check-tables=value
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:**

.\*

---

### rocksdb\_sst\_mgr\_rate\_bytes\_per\_sec

- **Description:** DBOptions::sst\_file\_manager rate\_bytes\_per\_sec for RocksDB

- **Commandline:**

```
--rocksdb-sst-mgr-rate-bytes-per-sec=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

18446744073709551615

---

## rocksdb\_stats\_dump\_period\_sec

- **Description:** DBOptions::stats\_dump\_period\_sec for RocksDB.

- **Commandline:**

```
--rocksdb-stats-dump-period-sec=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

600

- **Range:**

0

to

2147483647

---

## rocksdb\_stats\_level

- **Description:** Statistics Level for RocksDB. Default is 0 (kExceptHistogramOrTimers).

- **Commandline:**

```
--rocksdb-stats-level=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

4

- **Introduced:** [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#)
- 

## rocksdb\_stats\_recalc\_rate

- **Description:** The number of indexes per second to recalculate statistics for. 0 to disable background recalculation.

- **Commandline:**

```
--rocksdb-stats-recalc_rate=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

4294967295

- **Introduced:** [MariaDB 10.3.10](#) [MariaDB 10.2.18](#)
-

## `rocksdb_store_row_debug_checksums`

- **Description:** Include checksums when writing index/table records.

- **Commandline:**

```
--rocksdb-store-row-debug-checksums={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

## `rocksdb_strict_collation_check`

- **Description:** Enforce case sensitive collation for MyRocks indexes.

- **Commandline:**

```
--rocksdb-strict-collation-check={0|1}
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

## `rocksdb_strict_collation_exceptions`

- **Description:** List of tables (using regex) that are excluded from the case sensitive collation enforcement.

- **Commandline:**

```
--rocksdb-strict-collation-exceptions=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:** (Empty)

## `rocksdb_supported_compression_types`

- **Description:** Compression algorithms supported by RocksDB. Note that RocksDB does not make use of [MariaDB 10.7 compression-plugins](#).

- **Commandline:**

```
--rocksdb-supported-compression-types=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

- **Default Value:**

Snappy,Zlib,ZSTDNotFinal

## rocksdb\_table\_cache\_numshardbits

- **Description:** DBOptions::table\_cache\_numshardbits for RocksDB.

- **Commandline:**

```
--rocksdb-table-cache-numshardbits=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

6

- **Range:**

0

to

19

## rocksdb\_table\_stats\_sampling\_pct

- **Description:** Percentage of entries to sample when collecting statistics about table properties. Specify either 0 to sample everything or percentage [1..100]. By default 10% of entries are sampled.

- **Commandline:**

```
--rocksdb-table-stats-sampling-pct=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

10

- **Range:**

0

to

100

## rocksdb\_tmpdir

- **Description:** Directory for temporary files during DDL operations.

- **Commandline:**

```
--rocksdb-tmpdir[=value]
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:** (Empty)

## rocksdb\_trace\_sst\_api

- **Description:** Generate trace output in the log for each call to the SSTFileWriter.

- **Commandline:**

```
--rocksdb-trace-sst-api={0|1}
```

- **Scope:** Global, Session

- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

### rocksdb\_two\_write\_queues

- **Description:** DBOptions::two\_write\_queues for RocksDB.

- **Commandline:**  
--rocksdb-two-write-queues={0|1}

- **Scope:** Global,

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

- **Introduced:** [MariaDB 10.3.7](#) , [MariaDB 10.2.15](#)
- 

### rocksdb\_unsafe\_for\_binlog

- **Description:** Allowing statement based binary logging which may break consistency.

- **Commandline:**  
--rocksdb-unsafe-for-binlog={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

### rocksdb\_update\_cf\_options

- **Description:** Option updates per column family for RocksDB.

- **Commandline:**  
--rocksdb-update-cf-options=value

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

varchar

- **Default Value:** (Empty)

- **Introduced:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

### rocksdb\_use\_adaptive\_mutex

- **Description:** DBOptions::use\_adaptive\_mutex for RocksDB.

- **Commandline:**

--rocksdb-use-adaptive-mutex={0|1}

- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
- **Default Value:**  
OFF
- 

#### rocksdb\_use\_clock\_cache

- **Description:** Use ClockCache instead of default LRUCache for RocksDB.
- **Commandline:**  
--rocksdb-use-clock-cache={0|1}

- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
- **Default Value:**  
OFF
- 

#### rocksdb\_use\_direct\_io\_for\_flush\_and\_compaction

- **Description:** DBOptions::use\_direct\_io\_for\_flush\_and\_compaction for RocksDB.
  - **Commandline:**  
--rocksdb-use-direct-io-for-flush-and-compaction={0|1}
- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
- **Default Value:**  
OFF
- **Introduced:** MariaDB 10.3.1 , MariaDB 10.2.8
- 

#### rocksdb\_use\_direct\_reads

- **Description:** DBOptions::use\_direct\_reads for RocksDB.
  - **Commandline:**  
--rocksdb-use-direct-reads={0|1}
- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
- **Default Value:**  
OFF
- 

#### rocksdb\_use\_direct\_writes

- **Description:** DBOptions::use\_direct\_writes for RocksDB.
- **Commandline:**

```
--rocksdb-use-direct-reads={0|1}
```

- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
  - **Removed:** [MariaDB 10.3.1](#) , [MariaDB 10.2.8](#)
- 

### rocksdb\_use\_fsync

- **Description:** DBOptions::use\_fsync for RocksDB.

- **Commandline:**  
`--rocksdb-use-fsync={0|1}`

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
boolean

- **Default Value:**  
OFF
- 

### rocksdb\_validate\_tables

- **Description:** Verify all .frm files match all RocksDB tables (0 means no verification, 1 means verify and fail on error, and 2 means verify but continue).

- **Commandline:**  
`--rocksdb-validate-tables=#`

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
numeric

- **Default Value:**  
1

- **Range:**  
0  
to  
2
- 

### rocksdb\_verify\_row\_debug\_checksums

- **Description:** Verify checksums when reading index/table records.

- **Commandline:**  
`--rocksdb-verify-row-debug-checksums={0|1}`

- **Scope:** Global, Session
- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**  
OFF

---

## rocksdb\_wal\_bytes\_per\_sync

- **Description:** DBOptions::wal\_bytes\_per\_sync for RocksDB.

- **Commandline:**

```
--rocksdb-wal-bytes-per-sync=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

18446744073709551615

---

## rocksdb\_wal\_dir

- **Description:** DBOptions::wal\_dir for RocksDB. Directory where the write-ahead-log files are stored.

- **Commandline:**

```
--rocksdb-wal-dir=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

- **Default Value:** (Empty)
- 

## rocksdb\_wal\_recovery\_mode

- **Description:** DBOptions::wal\_recovery\_mode for RocksDB. Default is kAbsoluteConsistency. Records that are not yet committed are stored in the Write-Ahead-Log (WAL). If the server is not cleanly shut down, the recovery mode will determine the WAL recovery behavior.

- 1 : kAbsoluteConsistency. Will not start if any corrupted entries (including incomplete writes) are detected (the default).
- 0 : kTolerateCorruptedTailRecords. Ignores any errors found at the end of the WAL
- 2 : kPointInTimeRecovery. Replay of the WAL is halted after finding an error. The system will be recovered to the latest consistent point-in-time. Data from a replica can be used to replay past the point-in-time.
- 3 : kSkipAnyCorruptedRecords. A risky option where any corrupted entries are skipped while subsequent healthy WAL entries are applied.

- **Commandline:**

```
--rocksdb-wal-recovery-mode=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0

to

3

---

## rocksdb\_wal\_size\_limit\_mb

- **Description:** DBOptions::WAL\_size\_limit\_MB for RocksDB. Write-ahead-log files are moved to an archive directory once their memtables are flushed. This variable specifies the largest size, in MB, that the archive can be.

- **Commandline:**

```
--rocksdb-wal-size-limit-mb=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

9223372036854775807

---

## rocksdb\_wal\_ttl\_seconds

- **Description:** DBOptions::WAL\_ttl\_seconds for RocksDB. Oldest time, in seconds, that a write-ahead-log file should exist.

- **Commandline:**

```
--rocksdb-wal-ttl-seconds=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

numeric

- **Default Value:**

0

- **Range:**

0

to

9223372036854775807

---

## rocksdb\_whole\_key\_filtering

- **Description:** BlockBasedTableOptions::whole\_key\_filtering for RocksDB. If set (the default), the bloomfilter to use the whole key (rather than only the prefix) for filtering is enabled. Lookups should use the whole key for matching to make best use of this setting.

- **Commandline:**

```
--rocksdb-whole-key-filtering={0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

ON

---

## rocksdb\_write\_batch\_max\_bytes

- **Description:** Maximum size of write batch in bytes. 0 means no limit.

- **Commandline:**

```
--rocksdb-write-batch-max-bytes=#
```

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    numeric

- **Default Value:**

    0

- **Range:**

    0

    to

    18446744073709551615

---

### rocksdb\_write\_disable\_wal

- **Description:** WriteOptions::disableWAL for RocksDB.

- **Commandline:**

    --rocksdb-write-disable-wal={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

---

### rocksdb\_write\_ignore\_missing\_column\_families

- **Description:** WriteOptions::ignore\_missing\_column\_families for RocksDB.

- **Commandline:**

    --rocksdb-write-ignore-missing-column-families={0|1}

- **Scope:** Global, Session

- **Dynamic:** Yes

- **Data Type:**

    boolean

- **Default Value:**

    OFF

---

### rocksdb\_write\_policy

- **Description:** DBOptions::write\_policy for RocksDB.

- **Commandline:**

    --rocksdb-write-policy=val

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

    enum

- **Default Value:**

    write\_committed

- **Valid Values:**

    write\_committed

,

```
    write_prepared  
,  
    write_unprepared
```

- **Introduced:** MariaDB 10.3.10 , MariaDB 10.2.18

## 4.3.14.7 MyRocks Transactional Isolation

TODO:

- MyRocks uses snapshot isolation
- Support do READ-COMMITTED and REPEATABLE-READ
- SERIALIZABLE is not supported
  - There is no "Gap Locking" which makes Statement Based Replication unsafe (See MyRocks and Replication).

## 4.3.14.8 MyRocks and Replication

### Contents

1. [MyRocks and Statement-Based Replication](#)
  1. [Can One Still Use SBR with MyRocks?](#)
2. [Read-Free Slave](#)
3. [Differences From Upstream MyRocks](#)

Details about how MyRocks works with [replication](#) .

## MyRocks and Statement-Based Replication

[Statement-based](#) replication (SBR) works as follows: SQL statements are executed on the master (possibly concurrently). They are written into the binlog (this fixes their ordering, "a serialization"). The slave then reads the binlog and executes the statements in their binlog order.

In order to prevent data drift, serial execution of statements on the slave must have the same effect as concurrent execution of these statements on the master. In other words, transaction isolation on the master must be close to

SERIALIZABLE  
transaction isolation level (This is not a strict mathematical proof but shows the idea).

InnoDB achieves this by (almost) supporting

SERIALIZABLE  
transactional isolation level. It does so by supporting "Gap Locks". MyRocks doesn't support  
SERIALIZABLE  
isolation, and it doesn't support gap locks.

Because of that, generally one cannot use MyRocks and statement-based replication.

Updating a MyRocks table while having SBR on, will result in an error as follow:

```
ERROR 4056 (HY000): Can't execute updates on master with binlog_format != ROW.
```

### Can One Still Use SBR with MyRocks?

Yes. In many cases, database applications run a restricted set of SQL statements, and it's possible to prove that lack of Gap Lock support is not a problem and data skew will not occur.

In that case, one can set

```
@@rocksdb_unsafe_for_binlog=1
```

and MyRocks will work with SBR. The user is however responsible for making sure their queries are not causing a data skew.

### Read-Free Slave

TODO

### Differences From Upstream MyRocks

MyRocks upstream (that is, Facebook's MySQL branch) has a number of unique replication enhancements. These are available in upstream's version of MyRocks but not in MariaDB's version of MyRocks.

- Read-Free Replication (see <https://github.com/facebook/mysql-5.6/wiki/Read-Free-Replication> ) TODO
- 

<<

```

unique

_check_lag_threshold

>>
. This is FB/MySQL-5.6 feature where unique checks are disabled if replication lag exceeds a certain threshold.

•
<<

slave

_gtid_info=OPTIMIZED

>>
. This is said to be:

<<quote>>
"Whether SQL threads update mysql.slave_gtid_info table. If this value "
"is OPTIMIZED, updating the table is done inside storage engines to "
"avoid MySQL layer's performance overhead",
<</quote>>

```

## 4.3.14.9 MyRocks and Group Commit with Binary log

### Contents

1. Counter Values to Watch
2. On the Value of `rocksdb_wal_group_syncs`
3. Examples

MyRocks supports group commit with the [binary log](#) ([MDEV-11934](#)).

### Counter Values to Watch

(The following is only necessary if you are studying MyRocks internals)

MariaDB's group commit counters are:

[Binlog\\_commits](#) - how many transactions were written to the binary log

[Binlog\\_group\\_commits](#) - how many group commits happened. (e.g. if each group had two transactions, this will be twice as small as [Binlog\\_commits](#))

On the RocksDB side, there is one relevant counter: [Rocksdb\\_wal\\_synced](#) - How many times RocksDB's WAL file was synced. (TODO: this is after group commit happened, right?)

### On the Value of `rocksdb_wal_group_syncs`

FB/MySQL-5.6 has a [rocksdb\\_wal\\_group\\_syncs](#) counter (The counter is provided by MyRocks, it is not a view of a RocksDB counter). It is increased in `rocksdb_flush_wal()` when doing the `rdb->FlushWAL()` call.

`rocksdb_flush_wal()` is called by MySQL's Group Commit when it wants to make the effect of several `rocksdb_prepare()` calls persistent.

So, the value of [rocksdb\\_wal\\_group\\_syncs](#) in FB/MySQL-5.6 is similar to [Binlog\\_group\\_commits](#) in MariaDB.

MariaDB doesn't have that call, each `rocksdb_prepare()` call takes care of being persistent on its own.

Because of that, [rocksdb\\_wal\\_group\\_syncs](#) is zero for MariaDB. (Currently, it is only incremented when the binlog is rotated).

### Examples

So for a workload with concurrency=50, n\_queries=10K, one gets

- `Binlog_commits`=10K
- `Binlog_group_commits`=794
- `Rocksdb_wal_synced`=8362

This is on a RAM disk

For a workload with concurrency=50, n\_queries=10K, rotating laptop hdd, one gets

- `Binlog_commits`= 10K
- `Binlog_group_commits`=1403

- Rocksdb\_wal\_synced=400

The test took 38 seconds, Number of syncs was 1400+400=1800, which gives 45 syncs/sec which looks normal for this slow rotating desktop hdd.

Note that the WAL was synced fewer times than there were binlog commit groups (?)

## 4.3.14.10 Optimizer Statistics in MyRocks

### Contents

1. [How MyRocks computes statistics](#)
1. [Are index statistics predictable?](#)
2. [Records-in-range estimates](#)
2. [ANALYZE TABLE](#)
3. [Debugging helper variables](#)

This article describes how MyRocks storage engine provides statistics to the query optimizer.

There are three kinds of statistics:

- Table statistics (number of rows in the table, average row size)
- Index cardinality (how distinct values are in the index)
- records-in-range estimates (how many rows are in a certain range "const1 < tbl.key < const2").

### How MyRocks computes statistics

MyRocks (actually RocksDB) uses LSM files which are written once and never updated. When an LSM file is written, MyRocks will compute index cardinalities and number-of-rows for the data in the file. (The file generally has rows, index records and/or tombstones for multiple tables/indexes).

For performance reasons, statistics are computed based on a fraction of rows in the LSM file. The percentage of rows used is controlled by `rocksdb_table_stats_sampling_pct`; the default value is 10%.

Before the data is dumped into LSM file, it is stored in the MemTable. MemTable doesn't allow computing index cardinalities, but it can provide an approximate number of rows in the table. Use of MemTable data for statistics is controlled by `rocksdb_force_compute_memtable_stats`; the default value is

ON

### Are index statistics predictable?

Those who create/run MTR tests, need to know whether EXPLAIN output is deterministic. For MyRocks tables, the answer is NO (just like for InnoDB).

Statistics are computed using sampling and GetApproximateMemTableStats() which means that the #rows column in the EXPLAIN output may vary slightly.

### Records-in-range estimates

MyRocks uses RocksDB's GetApproximateSizes() call to produce an estimate for the number of rows in the certain range. The data in MemTable is also taken into account by issuing a GetApproximateMemTableStats call.

### ANALYZE TABLE

ANALYZE TABLE will possibly flush the MemTable (depending on the `rocksdb_flush_memtable_on_analyze` and `rocksdb_pause_background_work` settings).

After that, it will re-read statistics from the SST files and re-compute the summary numbers (TODO: and if the data was already on disk, the result should not be different from the one we had before ANALYZE?)

### Debugging helper variables

There are a few variables that will cause MyRocks to report certain pre-defined estimate numbers to the optimizer:

- `@@rocksdb_records_in_range` - if not 0, report that any range has this many rows
- `@@rocksdb_force_index_records_in_range` - if not 0, and FORCE INDEX hint is used, report that any range has this many rows.
- `@@rocksdb_debug_optimizer_n_rows` - if not 0, report that any MyRocks table has this many rows.

## 4.3.14.11 Differences Between MyRocks Variants

MyRocks is available in

- Facebook's (FB) MySQL branch (originally based on MySQL 5.6)
- MariaDB (from 10.2 and 10.3)
- Percona Server from 5.7

This page lists differences between these variants.

## Contents

1. RocksDB Data Location
2. Compression Algorithms
3. RocksDB Version Information
4. RocksDB Version
5. Binlog Position in  
information\_schema.rocksdb\_global\_info
6. Gap Lock Detector
7. Generated Columns
8. rpl\_skip\_tx\_api
9. Details

## RocksDB Data Location

FB and Percona store RocksDB files in \$datadir/

```
.rocksdb
. MariaDB puts them in $datadir/
#rocksdb
. This is more friendly for packaging and OS scripts.
```

## Compression Algorithms

- FB's branch doesn't provide binaries. One needs to compile it with appropriate compression libraries.
- In MariaDB, available compression algorithms can be seen in the `rocksdb_supported_compression_types` variable. From [MariaDB 10.7](#), algorithms can be [installed as a plugin](#). In earlier versions, the set of supported compression algorithms depends on the platform.
  - On Ubuntu 16.04 (current LTS) it is  
Snappy, Zlib, LZ4, LZ4HC
  - On CentOS 7.4 it is  
Snappy, Zlib
  - In the bintar tarball it is  
Snappy, Zlib
- Percona Server supports:
  - Zlib, ZSTD, LZ4 (the default), LZ4HC
  - Unsupported algorithms:  
Snappy, BZip2, XPress

## RocksDB Version Information

- FB's branch provides the  
`rocksdb_git_hash`  
\*status\* variable.
- MariaDB provides the  
`@@rocksdb_git_hash`  
\*system\* variable.
- Percona Server doesn't provide either.

## RocksDB Version

- Facebook's branch uses RocksDB 5.10.0 (the version number can be found in  
`include/rocksdb/version.h`  
)

```
commit ba295cda29daee3ffe58549542804efdfd969784
Author: Andrew Kryczka <andrewkr@fb.com>
Date:   Fri Jan 12 11:03:55 2018 -0800
```

- MariaDB currently uses 5.8.0

```
commit 9a970c81af9807071bd690f4c808c5045866291a
Author: Yi Wu <yiwu@fb.com>
Date:   Wed Sep 13 17:21:35 2017 -0700
```

- Percona Server uses 5.8.0

```
commit ab0542f5ec6e7c7e405267eaa2e2a603a77d570b
Author: Maysam Yabandeh <myabandeh@fb.com>
Date:   Fri Sep 29 07:55:22 2017 -0700
```

## Binlog Position in information\_schema.rocksdb\_global\_info

- FB branch provides information\_schema.rocksdb\_global\_info type=BINLOG, NAME=(FILE, POS, GTID).
- Percona Server doesn't provide it.
- MariaDB doesn't provide it.

One use of that information is to take the output of

```
myrocks_hotbackup
```

and make it a new master.

## Gap Lock Detector

- FB branch has a "Gap Lock Detector" feature. It is at the SQL layer. It can be controlled with  
`gap_lock_XXX` variables and is disabled by default (`gap-lock-raise-error=false`, `gap-lock-write-lock=false`).
- Percona Server has gap lock checking ON but doesn't seem to have any way to control it? Queries that use Gap Lock on MyRocks fail with an error like this:

```
mysql> insert into tbl2 select * from tbl1;
ERROR 1105 (HY000): Using Gap Lock without full unique key in multi-table or multi-statement transactions
is not allowed. You need to either rewrite queries to use all unique key columns in WHERE equal conditions,
or rewrite to single-table, single-statement transaction. Query: insert into tbl2 select * from tbl1
```

- MariaDB doesn't include the Gap Lock Detector.

## Generated Columns

- Both MariaDB and Percona Server support [generated columns](#), but neither one supports them for the MyRocks storage engine (attempts to create a table will produce an error).
- [Invisible columns](#) in [MariaDB 10.3](#) are supported (as they are an SQL layer feature).

## rpl\_skip\_tx\_api

Facebook's branch has a performance feature for replication slaves,

```
rpl_skip_tx_api
```

. It is not available in MariaDB or in Percona Server.

## Details

The above comparison was made using

- FB/MySQL 5.6.35
- Percona Server 5.7.20-19-log
- [MariaDB 10.2.13](#) (MyRocks is beta)

## 4.3.14.12 MyRocks and Bloom Filters

### Contents

1. [Bloom Filter Parameters](#)
  1. [Computing Prefix Length](#)
2. [Configuring Bloom Filter](#)
3. [Checking if Bloom Filter is Useful](#)

Bloom filters are used to reduce read amplification. Bloom filters can be set on a per-column family basis (see [myrocks-column-families](#) ).

## Bloom Filter Parameters

- How many bits to use
- whole\_key\_filtering=true/false
- Whether the bloom filter is for the entire key or for the prefix. In case of a prefix, you need to look at the index definition and compute the desired prefix length.

## Computing Prefix Length

- It's 4 bytes for  
index\_nr
- Then, for fixed-size columns (integer, date[time], decimal) it is key\_length as shown by  
EXPLAIN
  - . For VARCHAR columns, determining the length is tricky (It depends on the values stored in the table. Note that MyRocks encodes VARCHARs with "Variable-Length Space-Padded Encoding" format).

## Configuring Bloom Filter

To enable 10-bit bloom filter for 8-byte prefix length for column family "cf1", put this into my.cnf:

```
rocksdb_override_cf_options='cf1={block_based_table_factory={filter_policy=bloomfilter:10:false;whole_key_filtering=0;};prefix_e  
|  
,
```

and restart the server.

Check if the column family actually uses the bloom filter:

```
select *  
from information_schema.rocksdb_cf_options  
where  
cf_name='cf1' and  
option_type IN ('TABLE_FACTORY::FILTER_POLICY','PREFIX_EXTRACTOR');
```

CF_NAME	OPTION_TYPE	VALUE
cf1	PREFIX_EXTRACTOR	rocksdb.CappedPrefix.8
cf1	TABLE_FACTORY::FILTER_POLICY	rocksdb.BuiltinBloomFilter

## Checking if Bloom Filter is Useful

Watch these status variables:

```
show status like '%bloom%';  
+-----+  
| Variable_name | Value |  
+-----+  
| Rocksdb_bloom_filter_prefix_checked | 1 |  
| Rocksdb_bloom_filter_prefix_useful | 0 |  
| Rocksdb_bloom_filter_useful | 0 |  
+-----+
```

Other useful variables are:

- rocksdb\_force\_flush\_memtable\_now
  - bloom filter is only used when reading data from disk. If you are doing testing, flush the data to disk first.
- rocksdb\_skip\_bloom\_filter\_on\_read
  - skip using the bloom filter (default is FALSE).

## 4.3.14.13 MyRocks and CHECK TABLE

MyRocks supports the [CHECK TABLE](#) command.

The command will do a number of checks to verify that the table data is self-consistent.

The details about the errors are printed into the [error log](#). If [log\\_warnings](#) > 2, the error log will also have some informational messages which can help with troubleshooting.

Besides this, RocksDB has its own (low-level) log in

```
#rocksdb/LOG  
file.
```

## 4.3.14.14 MyRocks and Data Compression

### Contents

- 1. [Supported Compression Algorithms](#)
- 2. [Compression Settings](#)
  - 1. [Checking Compression Settings](#)
  - 2. [Modifying Compression Settings](#)
  - 3. [Caveat: Syntax Errors](#)
- 3. [Checking How the Data is Compressed](#)

MyRocks supports several compression algorithms.

## Supported Compression Algorithms

Supported compression algorithms can be checked like so:

```
show variables like 'rocksdb%compress%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| rocksdb_supported_compression_types | Snappy,Zlib,LZ4,LZ4HC,ZSTDNotFinal |
+-----+-----+
```

Another way to make the check is to look into

```
#rocksdb/LOG  
file in the data directory. It should have lines like:
```

```
2019/04/12-14:08:23.869919 7f839188b540 Compression algorithms supported:  
2019/04/12-14:08:23.869920 7f839188b540 kZSTDNotFinalCompression supported: 1  
2019/04/12-14:08:23.869922 7f839188b540 kZSTD supported: 1  
2019/04/12-14:08:23.869923 7f839188b540 kXpressCompression supported: 0  
2019/04/12-14:08:23.869924 7f839188b540 kLZ4HCCompression supported: 1  
2019/04/12-14:08:23.869924 7f839188b540 kLZ4Compression supported: 1  
2019/04/12-14:08:23.869925 7f839188b540 kBZip2Compression supported: 0  
2019/04/12-14:08:23.869926 7f839188b540 kZlibCompression supported: 1  
2019/04/12-14:08:23.869927 7f839188b540 kSnappyCompression supported: 1
```

## Compression Settings

Compression is set on a per-Column Family basis. See [MyRocks Column Families](#).

### Checking Compression Settings

To check current compression settings for a column family one can use a query like so:

```
select * from information_schema.rocksdb_cf_options
where option_type like '%compression%' and cf_name='default';
```

The output will be like:

```
+-----+-----+
| CF_NAME | OPTION_TYPE | VALUE |
+-----+-----+
| default | COMPRESSION_TYPE | kSnappyCompression |
| default | COMPRESSION_PER_LEVEL | NUL |
| default | COMPRESSION_OPTS | -14:32767:0 |
| default | BOTTOMMOST_COMPRESSION | kDisableCompressionOption |
| default | TABLE_FACTORY::VERIFY_COMPRESSION | 0 |
| default | TABLE_FACTORY::ENABLE_INDEX_COMPRESSION | 1 |
+-----+-----+
```

Current column family settings will be used for the new SST files.

## Modifying Compression Settings

Compression settings are not dynamic parameters, one cannot change them by setting `rocksdb_update_cf_options`.

The procedure to change compression settings is as follows:

- Edit

```
my.cnf  
to set rocksdb_override_cf_options.
```

Example:

```
rocksdb-override-cf-options='cf1={compression=kZSTD;bottommost_compression=kZSTD;}'
```

- Restart the server.

The data will not be re-compressed immediately. However, all new SST files will use the new compression settings, so as data gets inserted/updated the column family will gradually start using the new option.

## Caveat: Syntax Errors

Please note that

```
rocksdb-override-cf-options
```

syntax is quite strict. Any typos will result in the parse error, and MyRocks plugin will not be loaded. Depending on your configuration, the server may still start. If it does start, you can use this command to check if the plugin is loaded:

```
select * from information_schema.plugins where plugin_name='ROCKSDB'
```

(note that you need the "ROCKSDB" plugin. Other auxiliary plugins like "ROCKSDB\_TRX" might still get loaded).

Another way is to detect the error is check the error log. When option parsing fails, it will contain messages like so:

```
2019-04-16 11:07:57 140283675678016 [Warning] Invalid cf config for cf1 in override options (options: cf1={compression=kLZ4Compr  
2019-04-16 11:07:57 140283675678016 [ERROR] RocksDB: Failed to initialize CF options map.  
2019-04-16 11:07:57 140283675678016 [ERROR] Plugin 'ROCKSDB' init function returned error.  
2019-04-16 11:07:57 140283675678016 [ERROR] Plugin 'ROCKSDB' registration as a STORAGE ENGINE failed.
```

## Checking How the Data is Compressed

A query to check what compression is used in the SST files that store the data for a given table (test.t1):

```
select  
SP.sst_name, SP.compression_algo  
from  
information_schema.rocksdb_sst_props SP,  
information_schema.rocksdb_ddl D,  
information_schema.rocksdb_index_file_map IFM  
where  
D.table_schema='test' and D.table_name='t1' and  
D.index_number= IFM.index_number and  
IFM.sst_name=SP.sst_name;
```

Example output:

sst_name	compression_algo
000028.sst	Snappy
000028.sst	Snappy
000026.sst	Snappy
000026.sst	Snappy

## 4.3.14.15 MyRocks and Index-Only Scans

## Contents

1. Secondary Keys Only
2. Background: Mem-Comparable Keys
3. Index-Only Support for Various Datatypes
4. Index-Only Support for Various Collations
  1. Binary (Reversible) Collations
  2. Restorable Collations
  3. All Other Collations
5. Covering Secondary Key Lookups for VARCHARs

This article is about [MyRocks](#) and index-only scans on secondary indexes. It applies to MariaDB's MyRocks, Facebook's MyRocks, and other variants.

## Secondary Keys Only

The primary key in MyRocks is always the clustered key, that is, the index record is THE table record and so it's not possible to do "index only" because there isn't anything that is not in the primary key's (Key,Value) pair.

Secondary keys may or may not support index-only scans, depending on the datatypes of the columns that the query is trying to read.

## Background: Mem-Comparable Keys

MyRocks indexes store "mem-comparable keys" (that is, the key values are compared with

`memcmp`

). For some datatypes, it is easily possible to convert between the column value and its mem-comparable form, while for others the conversion is one-way.

For example, in case-insensitive collations capital and regular letters are considered identical, i.e. '`c`' = '`C`'. For some datatypes, MyRocks stores some extra data which allows it to restore the original value back. (For the

`latin1_general_ci`

`collation` and character '`c`', for example, it will store one bit which says whether the original value was a small '`c`' or a capital letter '`C`'. This doesn't work for all datatypes, though.

## Index-Only Support for Various Datatypes

Index-only scans are supported for numeric and date/time datatypes. For CHAR and VAR[CHAR], it depends on which collation is used, see below for details.

Index-only scans are currently not supported for less frequently used datatypes, like

•

`BIT(n)`

•

`SET(...)`

•

`ENUM(...)`

*It is actually possible to add support for those, feel free to write a patch or at least make a case why a particular datatype is important*

## Index-Only Support for Various Collations

As far as Index-only support is concerned, MyRocks distinguishes three kinds of collations:

### 1. Binary (Reversible) Collations

These are

`binary`

,

`latin1_bin`

, and

`utf8_bin`

For these collations, it is possible to convert a value back from its mem-comparable form. Hence, one can restore the original value back from its index record, and index-only scans are supported.

## 2. Restorable Collations

These are collations where one can store some extra information which helps to restore the original value.

Criteria (from storage/rocksdb/rdb\_datadic.cc, rdb\_is\_collation\_supported()) are:

- The charset should use 1-byte characters (so, unicode-based collations are not included)
- strxfrm(1 byte) = {one 1-byte weight value always}
- no binary sorting
- PAD attribute

The examples are:

```
latin1_general_ci  
,  
latin1_general_cs  
,  
latin1_swedish_ci  
, etc.
```

Index-only scans are supported for these collations.

## 3. All Other Collations

For these collations, there is no known way to restore the value from its mem-comparable form, and so index-only scans are not supported.

MyRocks needs to fetch the clustered PK record to get the field value.

## Covering Secondary Key Lookups for VARCHARs

TODO: there is also this optimization:

<https://github.com/facebook/mysql-5.6/issues/303> <https://github.com/facebook/mysql-5.6/commit/f349c95848e92b5b27b44f0e57194100eb0997e7>

document it.

### 4.3.14.16 MyRocks and START TRANSACTION WITH CONSISTENT SNAPSHOT

FB/MySQL has added new syntax:

```
START TRANSACTION WITH CONSISTENT ROCKSDB|INNODB SNAPSHOT;
```

The statement returns the binlog coordinates pointing at the snapshot.

MariaDB (and Percona Server) support extension to the regular

```
START TRANSACTION WITH CONSISTENT SNAPSHOT;
```

syntax as documented in [Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT](#).

After issuing the statement, one can examine the `binlog_snapshot_file` and `binlog_snapshot_position` status variables to see the binlog position that corresponds to the snapshot.

## See Also

- [START TRANSACTION](#)
- [Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT](#)

### 4.3.14.17 MyRocks Column Families

MyRocks stores data in column families. These are similar to tablespaces. By default, the data is stored in the default column family.

One can specify which column family the data goes to by using index comments:

```
INDEX index_name(col1, col2, ...) COMMENT 'column_family_name'
```

## Contents

1. Reasons for Column Families
2. Creating a Column Family
3. Dropping a Column Family
4. Setting Column Family Parameters
  1. `rocksdb_override_cf_options`
5. Examining Column Family Parameters

If the column name starts with

`rev:`

, the column family is reverse-ordered.

## Reasons for Column Families

Storage parameters like

- Bloom filter settings
- Compression settings
- Whether the data is stored in reverse order

are specified on a per-column family basis.

## Creating a Column Family

When creating a table or index, you can specify the name of the column family for it. If the column family doesn't exist, it will be automatically created.

## Dropping a Column Family

There is currently no way to drop a column family. RocksDB supports this internally but MyRocks doesn't provide any way to do it.

## Setting Column Family Parameters

Use these variables:

- `rocksdb_default_cf_options` - a my.cnf parameter specifying default options for all column families.
- `rocksdb_override_cf_options` - a my.cnf parameter specifying per-column family option overrides.
- `rocksdb_update_cf_options` - a dynamically-settable variable which allows to change parameters online. Not all parameters can be changed.

### `rocksdb_override_cf_options`

This parameter allows one to override column family options for specific column families. Here is an example of how to set option1=value1 and option2=value for column family cf1, and option3=value3 for column family cf3:

```
rocksdb_override_cf_options='cf1={option1=value1;option2=value2};cf2={option3=value3}'
```

One can check the contents of

```
INFORMATION_SCHEMA.ROCKSDB_CF_OPTIONS
```

to see what options are available.

Options that are frequently configured are:

- Data compression. See [myrocks-and-data-compression](#).
- Bloom Filters. See [myrocks-and-bloom-filters](#).

## Examining Column Family Parameters

See the [INFORMATION\\_SCHEMA.ROCKSDB\\_CF\\_OPTIONS](#) table.

### 4.3.14.18 MyRocks in MariaDB 10.2 vs MariaDB 10.3

MyRocks storage engine itself is identical in [MariaDB 10.2](#) and [MariaDB 10.3](#).

[MariaDB 10.3](#) has a feature that should be interesting for MyRocks users. It is the `gtid_pos_auto_engines` option ([MDEV-12179](#)). This is a performance feature for replication slaves that use multiple transactional storage engines.

For further information, see [mysql.gtid\\_slave\\_pos table](#).

### 4.3.14.19 MyRocks Performance Troubleshooting

## Contents

1. Status Variables
2. SHOW ENGINE ROCKSDB STATUS
3. Performance Context

MyRocks exposes its performance metrics through several interfaces:

- Status variables
- SHOW ENGINE ROCKSDB STATUS
- RocksDB's perf context

the contents slightly overlap, but each source has its own unique information, so be sure to check all three.

## Status Variables

Check the output of

```
SHOW STATUS like 'Rocksdb%'
```

See [MyRocks Status Variables](#) for more information.

## SHOW ENGINE ROCKSDB STATUS

This produces a lot of information.

One particularly interesting part is compaction statistics. It shows the amount of data on each SST level and other details:

```
***** 4. row *****
Type: CF_COMPACTION
Name: default
Status:
** Compaction Stats [default] **
Level    Files     Size      Score Read(GB)  Rn(GB)  Rnp1(GB) Write(GB) Wnew(GB) Moved(GB) W-Amp Rd(MB/s) Wr(MB/s) Comp(sec) Comp(cn)
-----+
L0      3/0      30.16 MB  1.0      0.0      0.0      0.0      11.9      11.9      0.0      1.0      0.0      76.6      159      63
L1      5/0      247.54 MB  1.0      0.7      0.2      0.5      0.5      0.0      11.6      2.6      58.5      44.1      12
L2      112/0     2.41 GB   1.0      0.6      0.0      0.6      0.5      -0.1      11.4      43.4      55.2      45.9      11
L3      466/0     8.91 GB   0.4      0.0      0.0      0.0      0.0      0.0      8.9      0.0      0.0      0.0      0.0
Sum     586/0     11.59 GB  0.0      1.3      0.2      1.0      12.8      11.8      32.0      1.1      7.1      72.6      181      63
Int     0/0       0.00 KB   0.0      0.9      0.1      0.8      0.8      0.0      0.1      20.5      48.4      45.3      19
```

## Performance Context

RocksDB has an internal mechanism called "perf context". The counter values are exposed through two tables:

- [INFORMATION\\_SCHEMA.ROCKSDB\\_PERF\\_CONTEXT\\_GLOBAL](#) - global counters
- [INFORMATION\\_SCHEMA.ROCKSDB\\_PERF\\_CONTEXT](#) - Per-table/partition counters

By default statistics are NOT collected. One needs to set

```
rocksdb_perf_context_level
to some value (e.g. 3) to enable collection.
```

## 4.3.15 OQGRAPH

The Open Query GRAPH computation engine, or OQGRAPH as the engine itself is called, allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).

OQGRAPH Version	Introduced	Maturity
3.0	<a href="#">MariaDB 10.0.25</a>	Gamma
3.0	<a href="#">MariaDB 10.0.7</a>	Beta
2.0	<a href="#">MariaDB 5.2.1</a>	



### Installing OQGRAPH

[Installing OQGRAPH](#).



### OQGRAPH Overview

[Overview of the OQGRAPH storage engine](#).



### OQGRAPH Examples

[OQGRAPH examples](#).



## Compiling OQGRAPH

[How to compile OQGRAPH.](#)



## Building OQGRAPH Under Windows

[OQGRAPH build instructions for Windows.](#)



## OQGRAPH System and Status Variables

[List and description of OQGRAPH system and status variables.](#)

There are [5 related questions](#).

## 4.3.15.1 Installing OQGRAPH

### Contents

- 1. [Installation](#)
  - 1. [Debian and Ubuntu](#)
  - 2. [Fedora/Red Hat/CentOS](#)
  - 3. [Installing the Plugin](#)
- 2. [See Also](#)

The Open Query GRAPH computation engine, or OQGRAPH as the engine itself is called, allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).

## Installation

The OQGRAPH storage engine exists as a separate package in the repositories for [MariaDB 10.0.7](#) and later. On Ubuntu and Debian the package is called

`mariadb-oqgraph-engine-10.0`

or

`mariadb-plugin-oqgraph`

. On Red Hat, CentOS, and Fedora the package is called

`MariaDB-oqgraph-engine`

. To install the plugin, first install the appropriate package and then install the plugin using the [INSTALL SONAME](#) or [INSTALL PLUGIN](#) commands.

### Debian and Ubuntu

On Debian and Ubuntu, install the package as follows:

```
sudo apt-get install mariadb-plugin-oqgraph
```

or (for [MariaDB 10.0](#))

```
sudo apt-get install mariadb-oqgraph-engine-10.0
```

### Fedora/Red Hat/CentOS

Note that OQGRAPH v3 requires libjudy, which is not in the official Red Hat/Fedora repositories. This needs to be installed first, for example:

```
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm  
rpm -Uvh epel-release-6-8.noarch.rpm
```

Then install the package, as follows:

```
sudo yum install MariaDB-oqgraph-engine
```

### Installing the Plugin

On either system you can then launch the

`mysql`

command-line client and install the plugin in MariaDB as follows:

```
INSTALL SONAME 'ha_oqgraph';
```

## See Also

More information on this engine is found on the OpenQuery website: <https://openquery.com.au/products/graph-engine>

### 4.3.15.2 OQGRAPH Overview

#### Contents

1. [Installing](#)
2. [Creating a Table](#)
3. [Example with origin and destination nodes only](#)
4. [Manipulating Weight](#)

The Open Query GRAPH computation engine, or OQGRAPH as the engine itself is called, allows you to handle hierarchies (tree structures) and complex graphs (nodes having many connections in several directions).

OQGRAPH is unlike other storage engines, consisting of an entirely different architecture to a regular storage engine such as Aria, MyISAM or InnoDB.

It is intended to be used for retrieving hierarchical information, such as those used for graphs, routes or social relationships, in plain SQL.

## Installing

See [Installing OQGRAPH](#). Note that the [query cache](#) needs to be disabled when using OQGRAPH with [MariaDB 5.5.35](#) and before, or 10.0.8 and before (see [MDEV-5744](#)). With newer versions, the query cache can be enabled and OQGRAPH will not use it.

## Creating a Table

The following documentation is based upon [MariaDB 10.0.7](#) and OQGRAPH v3. Details have changed since older versions.

## Example with origin and destination nodes only

To create an OQGRAPH v3 table, a backing table must first be created. This backing table will store the actual data, and will be used for all INSERTs, UPDATEs and so on. It must be a regular table, not a view. Here's a simple example to start with:

```
CREATE TABLE oq_backing (
    origid INT UNSIGNED NOT NULL,
    destid INT UNSIGNED NOT NULL,
    PRIMARY KEY (origid, destid),
    KEY (destid)
);
```

Some data can be inserted into the backing table to test with later:

```
INSERT INTO oq_backing(origid, destid)
VALUES (1,2), (2,3), (3,4), (4,5), (2,6), (5,6);
```

Now the read-only OQGRAPH table is created. The CREATE statement must match the format below - any difference will result in an error.

```
CREATE TABLE oq_graph (
    latch VARCHAR(32) NULL,
    origid BIGINT UNSIGNED NULL,
    destid BIGINT UNSIGNED NULL,
    weight DOUBLE NULL,
    seq BIGINT UNSIGNED NULL,
    linkid BIGINT UNSIGNED NULL,
    KEY (latch, origid, destid) USING HASH,
    KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';
```

An older format has the latch field as a SMALLINT rather than a VARCHAR. In [MariaDB 10.0.7](#) with OQGRAPH v3, the format is still valid, but gives an error by default:

```

CREATE TABLE oq_old (
    latch SMALLINT UNSIGNED NULL,
    origid BIGINT UNSIGNED NULL,
    destid BIGINT UNSIGNED NULL,
    weight DOUBLE NULL,
    seq BIGINT UNSIGNED NULL,
    linkid BIGINT UNSIGNED NULL,
    KEY (latch, origid, destid) USING HASH,
    KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';

ERROR 1005 (HY000): Can't create table `test`.`oq_old` (errno: 140 "Wrong create options")

```

The old, deprecated format can still be used if the value of the `oqgraph_allow_create_integer_latch` system variable is changed from its default,

```

    FALSE
    , to
    TRUE

```

```

SET GLOBAL oqgraph_allow_create_integer_latch=1;

CREATE TABLE oq_old (
    latch SMALLINT UNSIGNED NULL,
    origid BIGINT UNSIGNED NULL,
    destid BIGINT UNSIGNED NULL,
    weight DOUBLE NULL,
    seq BIGINT UNSIGNED NULL,
    linkid BIGINT UNSIGNED NULL,
    KEY (latch, origid, destid) USING HASH,
    KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';
Query OK, 0 rows affected, 1 warning (0.19 sec)

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Warning | 1287 | 'latch SMALLINT UNSIGNED NULL' is deprecated and will be removed in a future release. Please use 'latch VARCHAR'
+-----+-----+

```

Data is only inserted into the backing table, not the OQGRAPH table.

Now, having created the

```

oq_graph
table linked to a backing table, it is now possible to query the
oq_graph
table directly. The
weight
field, since it was not specified in this example, defaults to
1

```

```

SELECT * FROM oq_graph;
+-----+-----+-----+-----+-----+
| latch | origid | destid | weight | seq | linkid |
+-----+-----+-----+-----+-----+
| NULL | 1 | 2 | 1 | NULL | NULL |
| NULL | 2 | 3 | 1 | NULL | NULL |
| NULL | 2 | 6 | 1 | NULL | NULL |
| NULL | 3 | 4 | 1 | NULL | NULL |
| NULL | 4 | 5 | 1 | NULL | NULL |
| NULL | 5 | 6 | 1 | NULL | NULL |
+-----+-----+-----+-----+-----+

```

The data here represents one-directional starting and ending nodes. So node 2 has paths to node 3 and node 6, while node 6 has no paths to any other node.

## Manipulating Weight

There are three fields which can be manipulated:

```
origid  
,  
destid  
(the example above uses these two), as well as  
weight  
. To create a backing table with a  
weight  
field as well, the following syntax can be used:
```

```
CREATE TABLE oq2_backing (  
    origid INT UNSIGNED NOT NULL,  
    destid INT UNSIGNED NOT NULL,  
    weight DOUBLE NOT NULL,  
    PRIMARY KEY (origid, destid),  
    KEY (destid)  
);
```

```
INSERT INTO oq2_backing(origid, destid, weight)  
VALUES (1,2,1), (2,3,1), (3,4,3), (4,5,1), (2,6,10), (5,6,2);
```

```
CREATE TABLE oq2_graph (  
    latch VARCHAR(32) NULL,  
    origid BIGINT UNSIGNED NULL,  
    destid BIGINT UNSIGNED NULL,  
    weight DOUBLE NULL,  
    seq BIGINT UNSIGNED NULL,  
    linkid BIGINT UNSIGNED NULL,  
    KEY (latch, origid, destid) USING HASH,  
    KEY (latch, destid, origid) USING HASH  
)  
ENGINE=QQGRAPH  
data_table='oq2_backing' origid='origid' destid='destid' weight='weight';
```

```
SELECT * FROM oq2_graph;  
+-----+-----+-----+-----+-----+  
| latch | origid | destid | weight | seq | linkid |  
+-----+-----+-----+-----+-----+  
| NULL | 1 | 2 | 1 | NULL | NULL |  
| NULL | 2 | 3 | 1 | NULL | NULL |  
| NULL | 2 | 6 | 10 | NULL | NULL |  
| NULL | 3 | 4 | 3 | NULL | NULL |  
| NULL | 4 | 5 | 1 | NULL | NULL |  
| NULL | 5 | 6 | 2 | NULL | NULL |  
+-----+-----+-----+-----+-----+
```

See [QQGRAPH Examples](#) for QQGRAPH usage examples.

### 4.3.15.3 OQGRAPH Examples

#### Contents

1. [Creating a Table with origid, destid Only](#)
2. [Creating a Table with Weight](#)
3. [Shortest Path](#)
4. [Possible Destinations](#)
5. [Leaf Nodes](#)
6. [Summary of Implemented Latch Commands](#)
7. [See Also](#)

#### Creating a Table with origid, destid Only

```
CREATE TABLE oq_backing (  
    origid INT UNSIGNED NOT NULL,  
    destid INT UNSIGNED NOT NULL,  
    PRIMARY KEY (origid, destid),  
    KEY (destid)  
);
```

Some data can be inserted into the backing table to test with later:

```
INSERT INTO oq_backing(origid, destid)
VALUES (1,2), (2,3), (3,4), (4,5), (2,6), (5,6);
```

Now the read-only **OQGRAPH** table is created.

From [MariaDB 10.1.2](#) onwards you can use the following syntax:

```
CREATE TABLE oq_graph
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';
```

Prior to [MariaDB 10.1.2](#), the **CREATE** statement must match the format below - any difference will result in an error.

```
CREATE TABLE oq_graph (
    latch VARCHAR(32) NULL,
    origid BIGINT UNSIGNED NULL,
    destid BIGINT UNSIGNED NULL,
    weight DOUBLE NULL,
    seq BIGINT UNSIGNED NULL,
    linkid BIGINT UNSIGNED NULL,
    KEY (latch, origid, destid) USING HASH,
    KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';
```

## Creating a Table with Weight

For the examples on this page, we'll create a second OQGRAPH table and backing table, this time with

weight  
as well.

```
CREATE TABLE oq2_backing (
    origid INT UNSIGNED NOT NULL,
    destid INT UNSIGNED NOT NULL,
    weight DOUBLE NOT NULL,
    PRIMARY KEY (origid, destid),
    KEY (destid)
);
```

```
INSERT INTO oq2_backing(origid, destid, weight)
VALUES (1,2,1), (2,3,1), (3,4,3), (4,5,1), (2,6,10), (5,6,2);
```

```
CREATE TABLE oq2_graph (
    latch VARCHAR(32) NULL,
    origid BIGINT UNSIGNED NULL,
    destid BIGINT UNSIGNED NULL,
    weight DOUBLE NULL,
    seq BIGINT UNSIGNED NULL,
    linkid BIGINT UNSIGNED NULL,
    KEY (latch, origid, destid) USING HASH,
    KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq2_backing' origid='origid' destid='destid' weight='weight';
```

## Shortest Path

A

latch  
value of  
'dijkstras'  
and an  
origid  
and  
destid  
is used for finding the shortest path between two nodes, for example:

```

SELECT * FROM oq_graph WHERE latch='breadth_first' AND origid=1 AND destid=6;
+-----+-----+-----+-----+
| latch | origid | destid | weight | seq  | linkid |
+-----+-----+-----+-----+
| dijkstras|    1 |      6 |    NULL |    0 |     1 |
| dijkstras|    1 |      6 |      1 |    1 |     2 |
| dijkstras|    1 |      6 |      1 |    2 |     6 |
+-----+-----+-----+-----+

```

Note that nodes are uni-directional, so there is no path from node 6 to node 1:

```

SELECT * FROM oq_graph WHERE latch='dijkstras' AND origid=6 AND destid=1;
Empty set (0.00 sec)

```

Using the [GROUP\\_CONCAT](#) function can produce more readable results, for example:

```

SELECT GROUP_CONCAT(linkid ORDER BY seq) AS path FROM oq_graph
WHERE latch='dijkstras' AND origid=1 AND destid=6;
+-----+
| path  |
+-----+
| 1,2,6 |
+-----+

```

Using the table

[oq2\\_graph](#), the shortest path is different:

```

SELECT GROUP_CONCAT(linkid ORDER BY seq) AS path FROM oq2_graph
WHERE latch='dijkstras' AND origid=1 AND destid=6;
+-----+
| path   |
+-----+
| 1,2,3,4,5,6 |
+-----+

```

The reason is the weight between nodes 2 and 6 is

10  
in  
[oq\\_graph2](#), so the shortest path taking into account  
weight  
is now across more nodes.

## Possible Destinations

```

SELECT GROUP_CONCAT(linkid) AS dests FROM oq_graph WHERE latch='dijkstras' AND origid=2;
+-----+
| dests   |
+-----+
| 5,4,6,3,2 |
+-----+

```

Note that this returns all possible destinations along the path, not just immediate links.

## Leaf Nodes

MariaDB 10.3.3

Support for the  
leaves  
latch value was introduced in [MariaDB 10.3.3](#).

A

latch  
value of  
'leaves'  
and either

origid  
or  
destid  
is used for finding leaf nodes at the beginning or end of a graph.

```
INSERT INTO oq_backing(origid, destid)
VALUES (1,2), (2,3), (3,5), (4,5), (5,6), (6,7), (6,8), (2,8);
```

For example, to find all reachable nodes from

origid  
that only have incoming edges:

```
SELECT * FROM oq_graph WHERE latch='leaves' AND origid=2;
+-----+-----+-----+-----+
| latch | origid | destid | weight | seq | linkid |
+-----+-----+-----+-----+
| leaves |      2 |    NULL |      4 |     2 |      7 |
| leaves |      2 |    NULL |      1 |     1 |      8 |
+-----+-----+-----+-----+
```

And to find all nodes from which a path can be found to

destid  
that only have outgoing edges:

```
SELECT * FROM oq_graph WHERE latch='leaves' AND destid=5;
+-----+-----+-----+-----+
| latch | origid | destid | weight | seq | linkid |
+-----+-----+-----+-----+
| leaves |    NULL |      5 |      3 |     2 |      1 |
| leaves |    NULL |      5 |      1 |     1 |      4 |
+-----+-----+-----+-----+
```

## Summary of Implemented Latch Commands

Latch	Alternative	additional where clause fields	Graph operation
NULL	(unspecified)	(none)	List original data
(empty string)	0	(none extra)	List all vertices in linkid column
(empty string)	0	origid	List all first hop vertices from origid in linkid column
dijkstras	1	origid, destid	Find shortest path using Dijkstras algorithm between origid and destid, with traversed vertex ids in linkid column
dijkstras	1	origid	Find all vertices reachable from origid, listed in linkid column, and report sum of weights of vertices on path to given vertex in weight
dijkstras	1	destid	Find all vertices from which a path can be found to destid, listed in linkid column, and report sum of weights of vertices on path to given vertex in weight
breadth_first	2	origid	List vertices reachable from origid in linkid column
breadth_first	2	destid	List vertices from which a path can be found to destid in linkid column
breadth_first	2	origid, destid	Find shortest path between origid and destid, report in linkid column
leaves	4	origid	List vertices reachable from origid, that only have incoming edges (from <a href="#">MariaDB 10.3.3</a> )
leaves	4	destid	List vertices from which a path can be found to destid, that only have outgoing edges (from <a href="#">MariaDB 10.3.3</a> )
leaves	4	origid, destid	Not supported, will return an empty result

Note: the use of integer latch commands is deprecated and may be phased out in a future release. Currently, numeric values in the strings are interpreted as aliases, and use of an integer column can be optionally allowed, for the latch commands column.

The use of integer latches is controlled using the [oqgraph\\_allow\\_create\\_integer\\_latch](#) system variable.

## See Also

- OQGRAPH Overview

## 4.3.15.4 Compiling OQGRAPH

To compile OQGraph v2 you need to have the boost library with the versions not earlier than 1.40 and not later than 1.55 and gcc version not earlier than 4.5.

MariaDB starting with 10.0.7

OQGraph v3 compiles fine with the newer boost libraries, but it additionally needs the Judy library installed.

When all build prerequisites are met, OQGraph is enabled and compiled automatically. To enable or disable OQGRAPH explicitly, see the generic [plugin build instructions](#).

### Finding Out Why OQGRAPH Didn't Compile

If OQGRAPH gets compiled properly, there should be a file like:

```
storage/oqgraph/ha_oqgraph.so
```

If this is not the case, then you can find out if there is any modules missing that are required by OQGRAPH by doing:

```
cmake . -LAH | grep -i oqgraph
```

## 4.3.15.5 Building OQGRAPH Under Windows

OQGRAPH v3 can be built on Windows.

MariaDB starting with 10.0.11

This has been tested using Windows 7, Microsoft Visual Studio Express 2010 (32-bit), Microsoft Windows 64-bit Platform SDK 7.1 (64-bit), the Boost library >= 1.55 and Judy 1.0.5. Probably other recent versions of Boost, Judy or MSVC may work but these combinations have not been tested.

- Download the source package for Boost 1.55 from the Boost project website, <http://www.boost.org>
- Download the source package for Judy 1.05 via <http://judy.sourceforge.net/>
- Follow the documented instructions for building under Windows from the command line: [Building\\_MariaDB\\_on\\_Windows](#)
  - Ensure that the following variable is set to CMAKE:  
`JUDY_ROOT=path\to\judy\unzipped`
  - See also comments in  
`storage/oqgraph/cmake/FindJudy.cmake`

## 4.3.15.6 OQGRAPH System and Status Variables

### Contents

1. [System Variables](#)
  1. [oqgraph\\_allow\\_create\\_integer\\_latch](#)
2. [Status Variables](#)
  1. [Oqgraph\\_boost\\_version](#)
  2. [Oqgraph\\_compat\\_mode](#)
  3. [Oqgraph\\_verbose\\_debug](#)

This page documents system and status variables related to the [OQGRAPH storage engine](#). See [Server Status Variables](#) and [Server System Variables](#) for complete list of all system and status variables.

### System Variables

`oqgraph_allow_create_integer_latch`

- **Description:** Created when the OQGRAPH storage engine is installed, if set to

```
1
(
0
is default), permits the
latch
```

field to be an integer (see [OQGRAPH Overview](#) ).

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**

0

- **Introduced:** [MariaDB 10.0.7](#)

## Status Variables

### Oqgraph\_boost\_version

- **Description:** OQGRAPH boost version.

- **Scope:** Global, Session

- **Data Type:**

string

- **Introduced:** [MariaDB 10.0.7](#) with the [OQGRAPH](#) storage engine.

### Oqgraph\_compat\_mode

- **Description:** Whether or not legacy tables with integer latches are supported.

- **Scope:** Global, Session

- **Data Type:**

string

- **Introduced:** [MariaDB 10.1.5](#)

### Oqgraph\_verbose\_debug

- **Description:** Whether or not verbose debugging is enabled. If it is, performance may be adversely impacted

- **Scope:** Global, Session

- **Data Type:**

string

- **Introduced:** [MariaDB 10.1.5](#)

## 4.3.16 S3 Storage Engine

MariaDB starting with [10.5](#)

The [S3 storage engine](#) has been available since [MariaDB 10.5.4](#) .

S3 is a read-only storage engine that stores its data in Amazon S3.



### Using the S3 Storage Engine

[Using the S3 storage engine](#).



### Testing the Connections to S3

[Steps to help verify where an S3 problem could be.](#)



### S3 Storage Engine Internals

[The S3 storage engine is based on the Aria code.](#)



### aria\_s3\_copy

[Copies an Aria table to and from S3.](#)



## S3 Storage Engine Status Variables

*S3 Storage Engine-related status variables.*



## S3 Storage Engine System Variables

*S3 Storage Engine-related system variables.*

There are [2 related questions](#).

### 4.3.16.1 Using the S3 Storage Engine

MariaDB starting with [10.5](#)

The [S3 storage engine](#) has been available since [MariaDB 10.5.4](#).

#### Contents

1. [Installing the Plugin](#)
2. [Moving Data to S3](#)
3. [New Options for ALTER TABLE](#)
4. [mysqld Startup Options for S3](#)
5. [Typical my.cnf Entry for connecting to Amazon S3 service](#)
6. [Typical my.cnf entry for connecting to a minio S3 server](#)
7. [Typical Usage Case for S3 Tables](#)
8. [Operations Allowed on S3 Tables](#)
9. [Discovery](#)
10. [Replication](#)
11. [aria\\_s3\\_copy](#)
12. [mysqldump](#)
13. [ANALYZE TABLE](#)
14. [CHECK TABLE](#)
15. [Current Limitations](#)
  1. [Limitations in ALTER .. PARTITION](#)
16. [Performance Considerations](#)
  1. [Discovery](#)
  2. [Caching](#)
  3. [Things to Try to Increase Performance](#)
17. [Future Development Ideas](#)
18. [Troubleshooting S3 on SELinux](#)
19. [See Also](#)

The [S3 storage engine](#) is read only and allows one to archive MariaDB tables in Amazon S3, or any third-party public or private cloud that implements S3 API (of which there are many), but still have them accessible for reading in MariaDB.

### Installing the Plugin

As of [MariaDB 10.5.7](#), the S3 storage engine is currently [gamma maturity](#), so the following step can be omitted.

On earlier releases, when it was [alpha maturity](#), it will not load by default on a stable release of the server due to the default value of the [plugin\\_maturity](#) variable. Set to

`alpha`

(or below) in your config file to permit installation of the plugin:

```
[mysqld]
plugin-maturity = alpha
```

and restart the server.

Now [install the plugin library](#), for example:

```
INSTALL SONAME 'ha_s3';
```

If the library is not available, for example:

```
INSTALL SONAME 'ha_s3';
ERROR 1126 (HY000): Can't open shared library '/var/lib/mysql/lib64/mysql/plugin/ha_s3.so'
(errno: 13, cannot open shared object file: No such file or directory)
```

you may need to install a separate package for the S3 storage engine, for example:

```
shell> yum install MariaDB-s3-engine
```

## Moving Data to S3

To move data from an existing table to S3, one can run:

```
ALTER TABLE old_table ENGINE=S3
```

To get data back to a 'normal' table one can do:

```
ALTER TABLE s3_table ENGINE=INNODB
```

## New Options for `ALTER TABLE`

- `S3_BLOCK_SIZE`  
: Set to 4M as default. This is the block size for all index and data pages stored in S3.
- `COMPRESSION_ALGORITHM`  
: Set to 'none' as default. Which compression algorithm to use for block stored in S3. Options are:  
none  
or  
zlib

`ALTER TABLE` can be used on S3 tables as normal to add columns or change column definitions.

## mysqld Startup Options for S3

To be able to use S3 for storage one \*must\* define how to access S3 and where data are stored in S3:

- `s3_access_key` : The AWS access key to access your data
- `s3_secret_key` : The AWS secret key to access your data
- `s3_bucket` : The AWS bucket where your data should be stored. All MariaDB table data is stored in this bucket.
- `s3_region` : The AWS region where your data should be stored.

If you are using an S3 service that is using HTTP to connect (like <https://min.io/>) you also need to set the following variables:

- `s3_port` : Port number to connect to (0 means use default)
- `s3_use_http` : If true, force use of HTTP protocol

If you are going to use a primary-replica setup, you should look at the following variables:

- `s3_replicate_alter_as_create_select` : When converting an S3 table to local table, log all rows in binary log. Defaults to  
`TRUE`  
. This allows the replica to replicate  
`CREATE TABLE .. SELECT FROM s3_table`  
even if the replica doesn't have access to the original  
`s3_table`
- `s3_slave_ignore_updates` : Should be set if primary and replica share the same S3 instance. This tells the replica that it can ignore any updates to the S3 tables as they are already applied on the primary. Defaults to  
`FALSE`

The above defaults assume that the primary and replica don't share the same S3 instance.

Other, less critical options, are:

- `s3_host_name` : Hostname for the S3 service. "s3.amazonaws.com", Amazon S3 service, by default.
- `s3_protocol_version` : Protocol used to communicate with S3. One of "Auto", "Amazon" or "Original" where "Auto" is the default. If you get errors like "8 Access Denied" when you are connecting to another service provider, then try to change this option. The reason for this variable is that Amazon has changed some parts of the S3 protocol since they originally introduced it but other service providers are still using the original protocol.
- `s3_block_size` : Set to 4M as default. This is the default block size for a table, if not specified in `CREATE TABLE`.
- `s3_pagecache_buffer_size` : Default 128M. The size of the buffer used for data and index blocks for S3 tables. Increase this to get better index handling (for all reads and multiple writes) to as much as you can afford.

Last some options you probably don't have to ever touch:

- `s3_pagecache_age_threshold` : Default 300: This characterizes the number of hits a hot block has to be untouched until it is considered aged enough to be downgraded to a warm block. This specifies the percentage ratio of that number of hits to the total number of blocks in the page

cache.

- **s3\_pagecache\_division\_limit** : Default 100. The minimum percentage of warm blocks in key cache.
- **s3\_pagecache\_file\_hash\_size** : Default 512. Number of hash buckets for open files. If you have a lot of S3 files open you should increase this for faster flush of changes. A good value is probably 1/10 of number of possible open S3 files.
- **s3\_debug** : Default 0. Generates a trace file from libmarias3 on stderr (mysqld.err) for debugging the S3 protocol.

## Typical my.cnf Entry for connecting to Amazon S3 service

```
[mariadb]
s3=ON
s3-bucket=mariadb
s3-access-key=xxxx
s3-secret-key=xxx
s3-region=eu-north-1
s3-host-name=s3.amazonaws.com
# The following is useful if you want to use minio as a S3 server. (https://min.io/)
#s3-port=9000
#s3-use-htp=ON

# Primary and replica share same S3 tables.
s3-slave-ignore-updates=1

[aria_s3_copy]
s3-bucket=mariadb
s3-access-key=xxxx
s3-secret-key=xxx
s3-region=eu-north-1
s3-host-name=s3.amazonaws.com
# The following is useful if you want to use minio as a S3 server. (https://min.io/)
#s3-port=9000
#s3-use-htp=ON
```

## Typical my.cnf entry for connecting to a [minio](#) S3 server

```
[mariadb]
s3=ON
s3-host-name="127.0.0.1"
s3-bucket=storage-engine
s3-access-key=minio
s3-secret-key=minioadmin
s3-port=9000
s3-use-htp=ON

[aria_s3_copy]
s3=ON
s3-host-name="127.0.0.1"
s3-bucket=storage-engine
s3-access-key=minio
s3-secret-key=minioadmin
s3-port=9000
s3-use-htp=ON
```

## Typical Usage Case for S3 Tables

The typical use case would be that there exists tables that after some time would become fairly inactive, but are still important so that they can not be removed. In that case, an option is to move such a table to an archiving service, which is accessible through an S3 API.

Notice that S3 means the Cloud Object Storage API defined by Amazon AWS. Often the whole of Amazon's Cloud Object Storage is referred to as S3. In the context of the S3 archive storage engine, it refers to the API itself that defines how to store objects in a cloud service, being it Amazon's or someone else's. OpenStack for example provides an S3 API for storing objects.

The main benefit of storing things in an S3 compatible storage is that the cost of storage is much cheaper than many other alternatives. Many S3 implementations also provide reliable long-term storage.

## Operations Allowed on S3 Tables

- [ALTER TABLE](#) S3 supports all types, keys and other options that are supported by the [Aria](#) engine. One can also perform [ALTER TABLE](#) on an S3 table to add or modify columns etc.
- [DROP TABLE](#)
- [SELECT](#) Any SELECT operations you can perform on a normal table should work with an S3 table.
- [SHOW TABLES](#) will show all tables that exist in the current defined S3 location.

- S3 tables can be part of [partitions](#) . See Discovery below.

## Discovery

The S3 storage engine supports full [MariaDB discovery](#) . This means that if you have the S3 storage engine enabled and properly configured, the table stored in S3 will automatically be discovered when it's accessed with `SHOW TABLES` , `SELECT` or any other operation that tries to access it. In the case of `SELECT`, the .frm file from S3 will be copied to the local storage to speed up future accesses.

When an S3 table is opened for the first time (it's not in the table cache) and there is a local .frm file, the S3 engine will check if it's still relevant, and if not, update or delete the .frm file.

This means that if the table definition changes on S3 and it's in the local cache, one has to execute

[FLUSH TABLES](#)

to get MariaDB to notice the change and update the .frm file.

If partitioning S3 tables are used, the partition definitions will also be stored on S3 storage and will be discovered by other servers.

Discovery of S3 tables is not done for tables in the [mysql databases](#) to make mysqld boot faster and more securely.

## Replication

S3 works with [replication](#) . One can use replication in two different scenarios:

- The primary and replica share the same S3 storage. In this case the primary will make all changes to the S3 data and the replica will ignore any changes in the replication stream to S3 data . This scenario is achieved by setting `s3_slave_ignore_updates` to 1.
- The primary and replica don't share the same S3 storage or the replica uses another storage engine for the S3 tables. This scenario is achieved by setting `s3_slave_ignore_updates` to 0.

## aria\_s3\_copy

[aria\\_s3\\_copy](#) is an external tool that one can use to copy [Aria](#) tables to and from S3. Use

```
aria_s3_copy --help
to get the options of how to use it.
```

## mysqldump

- [mysqldump](#) will by default ignore S3 tables. If

```
mysqldump
is run with the
--copy-s3-tables
option, the resulting file will contain a CREATE statement for a similar Aria table, followed by the table data and ending with an
ALTER TABLE xxx ENGINE=S3
```

## ANALYZE TABLE

As of [MariaDB 10.5.14](#) ,

[ANALYZE TABLE](#)

is supported for S3 tables. As the S3 tables are read-only, a normal `ANALYZE TABLE` will not do anything. However using `ANALYZE TABLE table_name PERSISTENT FOR...` will now work.

## CHECK TABLE

As of [MariaDB 10.5.14](#) ,

[CHECK TABLE](#)

will work. As S3 tables are read only it is very unlikely that they can become corrupted. The only known way an S3 table could be corrupted if either the original table copied to S3 was corrupted or the process of copying the original table to S3 was somehow interrupted.

## Current Limitations

- [mysql-test-run](#) doesn't by default test the S3 engine as we can't embed AWS keys into mysql-test-run.
- Replicas should not access S3 tables while they are ALTERed! This is because there is no locking implemented to S3 between servers. However, after a table (either the original S3 table or the partitioned S3 table) is changed on the primary, the replica will notice this on the next access and update its local definition.

## Limitations in [ALTER .. PARTITION](#)

All [ALTER PARTITION](#) operations are supported on S3 partitioning tables except:

- REBUILD PARTITION
- TRUNCATE PARTITION
- REORGANIZE PARTITION

## Performance Considerations

Depending on your connection speed to your S3 provider, there can be some notable slowdowns in some operations.

### Discovery

As S3 is supporting discovery (automatically making tables available that are in S3) this can cause some small performance problems if the S3 engine is enabled. Partitioning S3 tables also support discovery.

- CREATE TABLE is a bit slower as the S3 engine has to check if the to-be-created table is already S3.
- Queries on information\_schema tables are slower as S3 has to check if there is new tables in S3.
- DROP of non existing tables are slower as S3 has to check if the table is in S3.

There are no performance degradation's when accessing existing tables on the server. Accessing the S3 table the first time will copy the .frm file from S3 to the local disk, speeding up future accesses to the table.

### Caching

- Accessing a table on S3 can take some time , especially if you are using big packets ( [s3\\_block\\_size](#) ). However the second access to the same data should be fast as it's then cached in the S3 page cache.

## Things to Try to Increase Performance

If you have performance problems with the S3 engine, here are some things you can try:

- Decreasing [s3\\_block\\_size](#) . This can be done both globally and per table.
- Use COMPRESSION\_ALGORITHM=zlib when creating the table. This will decrease the amount of data transferred from S3 to the local cache.
- Increasing the size of the s3 page cache: [s3\\_pagecache\\_buffer\\_size](#)

Try also to execute the query twice to check if the problem is that the data was not properly cached. When data is cached locally the performance should be excellent.

## Future Development Ideas

- Store aws keys and region in the mysql.servers table (as [Spider](#) and [FederatedX](#) ). This will allow one to have different tables on different S3 servers.
- Store s3 bucket, access\_key and secret key in a cache to better be able to reuse connections. This would save some memory and make some S3 accesses a bit faster as we could reuse old connections.

## Troubleshooting S3 on SELinux

If you get errors such as:

```
ERROR 3 (HY000): Got error from put_object(bubu/produkt/frm): 5 Couldn't connect to server
```

one reason could be that your system doesn't allow MariaDB to connect to ports other than 3306. To procedure to enable other ports is the following:

Search for the ports allowed for MariaDB:

```
$ sudo semanage port -l | grep mysqld_port_t
mysqld_port_t          tcp    1186, 3306, 63132-63164
```

Say you want to allow MariaDB to connect to port 32768:

```
$ sudo semanage port -a -t mysqld_port_t -p tcp 32768
```

You can verify that the new port, 32768, is now allowed for MariaDB:

```
$ sudo semanage port -l | grep mysqld_port_t
mysqld_port_t           tcp    32768,1186, 3306, 63132-63164
```

## See Also

- [S3 storage engine internals](#)

### 4.3.16.2 Testing the Connections to S3

MariaDB starting with [10.5](#)

The [S3 storage engine](#) has been available since [MariaDB 10.5.4](#).

#### Contents

1. [S3 Connection Variables](#)
  1. [Using aria\\_s3\\_copy to Test the Connection](#)
  2. [Using mysql-test-run to Test the Connection and the S3 Storage Engine](#)
2. [What to Do Next](#)
3. [See Also](#)

If you can't get the S3 storage engine to work, here are some steps to help verify where the problem could be.

## S3 Connection Variables

In most cases the problem is to correctly set the S3 connection variables.

The variables are:

- [s3\\_access\\_key](#) : The AWS access key to access your data
- [s3\\_secret\\_key](#) : The AWS secret key to access your data
- [s3\\_bucket](#) : The AWS bucket where your data should be stored. All MariaDB table data is stored in this bucket.
- [s3\\_region](#) : The AWS region where your data should be stored.
- [s3\\_host\\_name](#) : Hostname for the S3 service.
- [s3\\_protocol\\_version](#) : Protocol used to communicate with S3. One of "Amazon" or "Original"

There are several ways to ensure you get them right:

### Using aria\_s3\_copy to Test the Connection

[aria\\_s3\\_copy](#) is a tool that allows you to copy [aria](#) tables to and from S3. It's useful for testing the connection as it allows you to specify all s3 options on the command line.

Execute the following sql commands to create a trivial sql table:

```
use test;
create table s3_test (a int) engine=aria row_format=page transactional=0;
insert into s3_test values (1),(2);
flush tables s3_test;
```

Now you can use the [aria\\_s3\\_copy](#) tool to copy this to S3 from your shell/the command line:

```
shell> cd mariadb-data-directory/test
shell> aria_s3_copy --op=to --verbose --force --**other*options* s3_test.frm

Copying frm file s3_test.frm
Copying aria table: test.s3_test to s3
Creating aria table information test/s3_test/aria
Copying index information test/s3_test/index
Copying data information test/s3_test/data
```

As you can see from the above, [aria\\_s3\\_copy](#) is using the current directory as the database name.

You can also set the [aria\\_s3\\_copy](#) options in your my.cnf file to avoid some typing.

### Using mysql-test-run to Test the Connection and the S3 Storage Engine

One can use the [MariaDB test system](#) to run all default S3 test against your S3 storage.

To do that you have to locate the

```
mysql-test  
directory in your system and  
cd  
to it.
```

The config file for the S3 test system can be found at

```
suite/s3/my.cnf
```

. To enable testing you have to edit this file and add the s3 connection options to the end of the file. It should look something like this after editing:

```
!include include/default_mysqld.cnf  
!include include/default_client.cnf  
  
[mysqld.1]  
s3=ON  
#s3-host-name=s3.amazonaws.com  
#s3-protocol-version=Amazon  
s3-bucket=MariaDB  
s3-access-key=  
s3-secret-key=  
s3-region=
```

You must give values for

```
s3-access-key  
,  
s3-secret-key  
and  
s3-region  
that reflects your S3 provider. The  
s3-bucket  
name is defined by your administrator.
```

If you are not using Amazon Web Services as your S3 provider you must also specify

```
s3-hostname  
and possibly change  
s3-protocol-version  
to "Original".
```

Now you can test the configuration:

```
shell> cd **mysql-test** directory  
shell> ./mysql-test-run --suite=s3  
...  
s3.no_s3 [ pass ] 5  
s3.alter [ pass ] 11073  
s3.arguments [ pass ] 2667  
s3.basic [ pass ] 2757  
s3.discovery [ pass ] 7851  
s3.select [ pass ] 1325  
s3.unsupported [ pass ] 363
```

Note that there may be more tests in your output as we are constantly adding more tests to S3 when needed.

## What to Do Next

When you got the connection to work, you should add the options to your global my.cnf file. Now you can start testing S3 from your [mysql command client](#) by converting some existing table to S3 with [ALTER TABLE ... ENGINE=S3](#).

## See Also

- [Using the S3 Storage Engine](#)
- [Using MinIO with mysql-test-run to test the S3 storage engine](#)

### 4.3.16.3 S3 Storage Engine Internals

MariaDB starting with [10.5](#)

The [S3 storage engine](#) has been available since [MariaDB 10.5.4](#).

## Contents

1. [ALTER TABLE](#)
2. [Partitioning Tables](#)
3. [Big Reads](#)
4. [Compression](#)
5. [Structure Stored on S3](#)
6. [Using the awsctl Python Tool to Examine Data](#)
  1. [Installing awsctl on Linux](#)
  2. [Using the awsctl Tool](#)
7. [See Also](#)

The [S3 storage engine](#) is based on the [Aria](#) code. Internally the S3 storage inherits from the Aria code, with hooks that change reads, so that instead of reading data from the local disk it reads things from S3.

The S3 engine uses its own page cache, modified to be able to handle reading blocks from S3 (of size

`s3_block_size`

). Internally the S3 page cache uses pages of [aria-block-size](#) for splitting the blocks read from S3.

## ALTER TABLE

[ALTER TABLE](#) will first create a local table in the normal Aria on disk format and then move both index and data to S3 in buckets of [S3\\_BLOCK\\_SIZE](#). The .frm file is also copied to S3 for discovery to support discovery for other MariaDB servers. One can also use [ALTER TABLE](#) to change the structure of an S3 table.

## Partitioning Tables

Starting from [MariaDB 10.5.3](#), S3 tables can also be used with [Partitioning tables](#). All [ALTER PARTITION](#) operations are supported except:

- REBUILD PARTITION
- TRUNCATE PARTITION
- REORGANIZE PARTITION

## Big Reads

One of the properties of many S3 implementations is that they favor large reads. It's said that 4M gives the best performance, which is why the default value for

`S3_BLOCK_SIZE`  
is 4M.

## Compression

If compression (  
`COMPRESSION_ALGORITHM=zlib`  
) is used, then all index blocks and data blocks are compressed. The  
.frm  
file and Aria definition header (first page/pages in the index file) are not compressed as these are used by discovery/open.

If compression is used, then the local block size is

`S3_BLOCK_SIZE`  
, but the block stored in S3 will be the size of the compressed block.

Typical compression we have seen is in the range of 80% saved space.

## Structure Stored on S3

The table will be copied in S3 into the following locations:

`frm` file (for discovery):  
`s3_bucket/database/table/frm`

First index block (contains description of the Aria file):  
`s3_bucket/database/table/aria`

Rest of the index file:  
`s3_bucket/database/table/index/block_number`

Data file:  
`s3_bucket/database/table/data/block_number`

block\_number is a 6-digit decimal number, prefixed with 0 (Can be larger than 6 numbers, the prefix is just for nice output)

## Using the awsctl Python Tool to Examine Data

### Installing awsctl on Linux

```
# install python-pip (on an OpenSuse distribution)
# use the appropriate command for your distribution
zypper install python-pip
pip install --upgrade pip

# the following installs awscli tools in ~/.local/bin
pip install --upgrade --user awscli
export PATH=~/./local/bin:$PATH

# configure your aws credentials
aws configure
```

### Using the awsctl Tool

One can use the

```
aws
python tool to see how things are stored on S3:
```

```
shell> aws s3 ls --recursive s3://mariadb-bucket/
2019-05-10 17:46:48      8192 foo/test1/aria
2019-05-10 17:46:49    3227648 foo/test1/data/000001
2019-05-10 17:46:48      942 foo/test1/frm
2019-05-10 17:46:48   1015808 foo/test1/index/000001
```

To delete an obsolete table

```
foo.test1
one can do:
```

```
shell> ~/.local/bin/aws s3 rm --recursive s3://mariadb-bucket/foo/test1
delete: s3://mariadb-bucket/foo/test1/aria
delete: s3://mariadb-bucket/foo/test1/data/000001
delete: s3://mariadb-bucket/foo/test1/frm
delete: s3://mariadb-bucket/foo/test1/index/000001
```

## See Also

- [Using the S3 storage engine](#)

### 4.3.16.4 aria\_s3\_copy

MariaDB starting with [10.5](#)

The [S3 storage engine](#) has been available since [MariaDB 10.5.4](#).

`aria_s3_copy`  
is a tool for copying an [Aria](#) table to and from [S3](#).

The Aria table must be non transactional and have [ROW\\_FORMAT=PAGE](#).

For

```
aria_s3_copy
to work reliably, the table should not be changed by the MariaDB server during the copy, and one should have first performed FLUSH TABLES to ensure that the table is properly closed.
```

Example of properly created Aria table:

```
create table test1 (a int) transactional=0 row_format=PAGE engine=aria;
```

Note that [ALTER TABLE table\\_name ENGINE=S3](#) will work for any kind of table. This internally converts the table to an Aria table and then moves it to S3 storage.

## Main Arguments

Option	Description
-?, --help	Display this help and exit.
-k, --s3-access-key=name	AWS access key ID
-r, --s3-region=name	AWS region
-K, --s3-secret-key=name	AWS secret access key ID
-b, --s3-bucket=name	AWS prefix for tables
-h, --s3-host-name=name	Host name to S3 provider
-c, --compress	Use compression
-o, --op=name	Operation to execute. One of 'from_s3', 'to_s3' or 'delete_from_s3'
-d, --database=name	Database for copied table (second prefix). If not given, the directory of the table file is used
-B, --s3-block-size=#	Block size for data/index blocks in s3
-L, --s3-protocol-version=name	Protocol used to communication with S3. One of "Amazon" or "Original".
-f, --force	Force copy even if target exists
-v, --verbose	Write more information
-V, --version	Print version and exit.
-#, --debug[=name]	Output debug log. Often this is 'd:t:o,filename'.
--s3-debug	Output debug log from marias3 to stdout

## Typical Configuration in a my.cnf File

```
[aria_s3_copy]
s3-bucket=mariadb
s3-access-key=xxxx
s3-secret-key=xxx
s3-region=eu-north-1
#s3-host-name=s3.amazonaws.com
#s3-protocol-version=Amazon
verbose=1
op=to
```

## Example Usage

The following code will copy an existing Aria table named

```
test1
to S3. If the
--database
option is not given, then the directory name where the table files exist will be used as the database.
```

```
shell> aria_s3_copy --force --op=to --database=foo --compress --verbose --s3_block_size=4M test1
Delete of aria table: foo.test1
Delete of index information foo/test1/index
Delete of data information foo/test1/data
Delete of base information and frm
Copying frm file test1.frm
Copying aria table: foo.test1 to s3
Creating aria table information foo/test1/aria
Copying index information foo/test1/index
.
Copying data information foo/test1/data
.
```

When using

```
--verbose
,
aria_s3_copy
will write a dot for each #/79 part of the file copied.
```

## See Also

[Using the S3 storage engine](#) . This page has examples of .my.cnf entries for using  
`aria_s3_copy`

## 4.3.16.5 S3 Storage Engine Status Variables

MariaDB starting with [10.5](#)

The [S3 storage engine](#) has been available since [MariaDB 10.5.4](#) .

### Contents

1. [S3\\_pagecache\\_blocks\\_not\\_flushed](#)
2. [S3\\_pagecache\\_blocks\\_unused](#)
3. [S3\\_pagecache\\_blocks\\_used](#)
4. [S3\\_pagecache\\_reads](#)

This page documents status variables related to the [S3 storage engine](#) . See [Server Status Variables](#) for a complete list of status variables that can be viewed with `SHOW STATUS` .

See also the [Full list of MariaDB options, system and status variables](#) .

### S3\_pagecache\_blocks\_not\_flushed

- **Description:**
- **Scope:**
- **Data Type:**
- **Introduced:** [MariaDB 10.5](#)

### S3\_pagecache\_blocks\_unused

- **Description:**
- **Scope:**
- **Data Type:**
- **Introduced:** [MariaDB 10.5](#)

### S3\_pagecache\_blocks\_used

- **Description:**
- **Scope:**
- **Data Type:**
- **Introduced:** [MariaDB 10.5](#)

### S3\_pagecache\_reads

- **Description:**
- **Scope:**
- **Data Type:**
- **Introduced:** [MariaDB 10.5](#)

## 4.3.16.6 S3 Storage Engine System Variables

MariaDB starting with [10.5](#)

The [S3 storage engine](#) has been available since [MariaDB 10.5.4](#) .

## Contents

1. [Variables](#)
  1. [s3\\_access\\_key](#)
  2. [s3\\_block\\_size](#)
  3. [s3\\_bucket](#)
  4. [s3\\_debug](#)
  5. [s3\\_host\\_name](#)
  6. [s3\\_pagecache\\_age\\_threshold](#)
  7. [s3\\_pagecache\\_buffer\\_size](#)
  8. [s3\\_pagecache\\_division\\_limit](#)
  9. [s3\\_pagecache\\_file\\_hash\\_size](#)
  10. [s3\\_port](#)
  11. [s3\\_protocol\\_version](#)
  12. [s3\\_region](#)
  13. [s3\\_replicate\\_alter\\_as\\_create\\_select](#)
  14. [s3\\_secret\\_key](#)
  15. [s3\\_slave\\_ignore\\_updates](#)
  16. [s3\\_use\\_http](#)
2. [See Also](#)

This page documents system variables related to the [S3 storage engine](#).

See [Server System Variables](#) for a complete list of system variables and instructions on setting system variables.

Also see the [Full list of MariaDB options, system and status variables](#)

## Variables

### s3\_access\_key

- **Description:** The AWS access key to access your data. See [mysqld startup options for S3](#).

- **Commandline:**

```
--s3-access-key=val
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:** String
- **Default Value:** (Empty)
- **Introduced:** [MariaDB 10.5.4](#)

### s3\_block\_size

- **Description:** The default block size for a table, if not specified in [CREATE TABLE](#). Set to 4M as default. See [mysqld startup options for S3](#).

- **Commandline:**

```
--s3-block-size=#
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:** Numeric
- **Default Value:**

4194304

- **Range:**

4194304

to

16777216

- **Introduced:** [MariaDB 10.5.4](#)

### s3\_bucket

- **Description:** The AWS bucket where your data should be stored. All MariaDB table data is stored in this bucket. See [mysqld startup options for S3](#)

- **Commandline:**

```
--s3-bucket=val
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:** String

- **Default Value:**

MariaDB

- **Introduced:** MariaDB 10.5.4

---

## s3\_debug

- **Description:** Generates a trace file from libmarias3 on stderr (mysqld.err) for debugging the S3 protocol.

- **Commandline:**

```
--s3-debug{=0|1}
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:** Boolean

- **Valid Values:** 0 or 1

- **Default Value:**

0

- **Introduced:** MariaDB 10.5.4

---

## s3\_host\_name

- **Description:** Hostname for the S3 service. "s3.amazonaws.com", Amazon S3 service, by default

- **Commandline:**

```
--s3-host-name=val
```

- **Scope:** Global;

- **Dynamic:** No

- **Data Type:** String

- **Default Value:**

s3.amazonaws.com

- **Introduced:** MariaDB 10.5.4

---

## s3\_pagecache\_age\_threshold

- **Description:** This characterizes the number of hits a hot block has to be untouched until it is considered aged enough to be downgraded to a warm block. This specifies the percentage ratio of that number of hits to the total number of blocks in the page cache.

- **Commandline:**

```
--s3-pagecache-age-threshold=val
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:** Numeric

- **Default Value:**

300

- **Range:**

100

to

18446744073709551615

- **Introduced:** MariaDB 10.5.4

---

## s3\_pagecache\_buffer\_size

- **Description:** The size of the buffer used for index blocks for S3 tables. Increase this to get better index handling (for all reads and multiple writes) to as much as you can afford. Size can be adjusted in blocks of 8192

- **Commandline:**

```
--s3-pagecache-buffer-size=val
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:** Numeric
- **Default Value:**

134217728  
(128M)

- **Range:**

33554432  
to  
18446744073709551615

- **Introduced:** MariaDB 10.5.4
- 

## s3\_pagecache\_division\_limit

- **Description:** The minimum percentage of warm blocks in key cache.

- **Commandline:**

```
--s3-pagecache-division-limit=val
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:** Numeric
- **Default Value:**

100

- **Range:**

1  
to  
100

- **Introduced:** MariaDB 10.5.4
- 

## s3\_pagecache\_file\_hash\_size

- **Description:** Number of hash buckets for open files. Default 512. If you have a lot of S3 files open you should increase this for faster flush of changes. A good value is probably 1/10 of number of possible open S3 files.

- **Commandline:**

```
--s3-pagecache-file-hash-size=#
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:** Numeric
- **Default Value:**

512

- **Range:**

32  
to  
16384

- **Introduced:** MariaDB 10.5.4
- 

## s3\_port

- **Description:** The TCP port number on the S3 host to connect to. A value of 0 means determine automatically.

- **Commandline:**

```
--s3-port=#
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:** Numeric

- **Default Value:**

```
0
```

- **Range:**

```
0
```

```
to
```

```
65535
```

- **Introduced:** [MariaDB 10.5.7](#)
- 

## s3\_protocol\_version

- **Description:** Protocol used to communicate with S3. One of "Auto", "Amazon" or "Original" where "Auto" is the default. If you get errors like "8 Access Denied" when you are connecting to another service provider, then try to change this option. The reason for this variable is that Amazon has changed some parts of the S3 protocol since they originally introduced it but other service providers are still using the original protocol.

- **Commandline:**

```
--s3-protocol-version=val
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:** Enum

- **Valid Values:**

```
Auto
```

```
,
```

```
Amazon
```

```
or
```

```
Original
```

- **Default Value:**

```
Auto
```

- **Introduced:** [MariaDB 10.5.4](#)
- 

## s3\_region

- **Description:** The AWS region where your data should be stored. See [mysqld startup options for S3](#).

- **Commandline:**

```
--s3-region=val
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:** String

- **Default Value:** (Empty)

- **Introduced:** [MariaDB 10.5.4](#)
- 

## s3\_replicate\_alter\_as\_create\_select

- **Description:** When converting S3 table to local table, log all rows in binary log. This allows the slave to replicate

```
CREATE TABLE .. SELECT FROM s3_table  
even if the slave doesn't have access to the original  
s3_table
```

- **Commandline:**

```
--s3-replicate-alter-as-create-select{=0|1}
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:** Boolean
- **Default Value:**

1

- **Introduced:** [MariaDB 10.5.4](#)
- 

### s3\_secret\_key

- **Description:** The AWS secret key to access your data. See [mysqld startup options for S3](#).
- **Commandline:**

```
--s3-secret-key=val
```

- **Scope:** Global
  - **Dynamic:** No
  - **Data Type:** String
  - **Default Value:** (Empty)
  - **Introduced:** [MariaDB 10.5.4](#)
- 

### s3\_slave\_ignore\_updates

- **Description:** Should be set if master and slave share the same S3 instance. This tells the slave that it can ignore any updates to the S3 tables as they are already applied on the master.

- **Commandline:**

```
--s3-slave-ignore-updates{=0|1}
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:** Boolean
- **Default Value:**

0

- **Introduced:** [MariaDB 10.5.4](#)
- 

### s3\_use\_http

- **Description:** If enabled, HTTP will be used instead of HTTPS.

- **Commandline:**

```
--s3-use-http{=0|1}
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:** Boolean
- **Default Value:**

0

- **Introduced:** [MariaDB 10.5.7](#)
- 

## See Also

[Using the S3 Storage Engine](#)

## 4.3.17 Sequence Storage Engine

## Contents

1. [Installing](#)
2. [Usage and Examples](#)
3. [Table Name Conflicts](#)
4. [Resources](#)

This article is about the Sequence storage engine. For details about sequence objects, see [Sequences](#).

A **Sequence** engine allows the creation of ascending or descending sequences of numbers (positive integers) with a given starting value, ending value and increment.

It creates completely virtual, ephemeral tables automatically when you need them. There is no way to create a Sequence table explicitly. Nor are they ever written to disk or create

.frm files. They are read-only, [transactional](#), and [support XA](#).

## Installing

Until [MariaDB 10.0](#), the **Sequence** engine is usually distributed as a dynamic plugin, not part of the server binary. To be able to use it, you need to install it first:

```
INSTALL SONAME "ha_sequence";
```

From [MariaDB 10.1](#), the Sequence engine is installed by default.

If it has been correctly installed, [SHOW ENGINES](#) will list the Sequence storage engine as supported:

```
SHOW ENGINES\G
...
***** 5. row *****
  Engine: MyISAM
  Support: YES
  Comment: MyISAM storage engine
Transactions: NO
      XA: NO
  Savepoints: NO
***** 6. row *****
  Engine: SEQUENCE
  Support: YES
  Comment: Generated tables filled with sequential values
Transactions: YES
      XA: YES
  Savepoints: YES
***** 7. row *****
  Engine: MRG_MyISAM
  Support: YES
  Comment: Collection of identical MyISAM tables
Transactions: NO
      XA: NO
  Savepoints: NO
...
...
```

## Usage and Examples

To use a Sequence table, you simply select from it, as in

```
SELECT * FROM seq_1_to_5;
+-----+
| seq |
+-----+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+-----+
```

To use a sequence in a statement, you select from the table named by a pattern **seq\_** FROM

```
_to_
TO
or seq_
FROM
_to_
TO
_step_
STEP
```

In the case of an odd step, the sequence will commence with the

```
FROM
, and end at the final result before
TO
```

```
SELECT * FROM seq_1_to_15_step_3;
+---+
| seq |
+---+
| 1 |
| 4 |
| 7 |
| 10 |
| 13 |
+---+
```

A sequence can go backwards too. In this case the final value will always be the

```
TO
value, so that a descending sequence has the same values as an ascending sequence:
```

```
SELECT * FROM seq_5_to_1_step_2;
+---+
| seq |
+---+
| 5 |
| 3 |
| 1 |
+---+
```

```
SELECT * FROM seq_15_to_1_step_3;
+---+
| seq |
+---+
| 13 |
| 10 |
| 7 |
| 4 |
| 1 |
+---+
```

```
SELECT * FROM seq_15_to_2_step_3;
+---+
| seq |
+---+
| 14 |
| 11 |
| 8 |
| 5 |
| 2 |
+---+
```

This engine is particularly useful with joins and subqueries. For example, this query finds all prime numbers below 50:

```

SELECT seq FROM seq_2_to_50 s1 WHERE 0 NOT IN
    (SELECT s1.seq % s2.seq FROM seq_2_to_50 s2 WHERE s2.seq <= sqrt(s1.seq));
+----+
| seq |
+----+
| 2 |
| 3 |
| 5 |
| 7 |
| 11 |
| 13 |
| 17 |
| 19 |
| 23 |
| 29 |
| 31 |
| 37 |
| 41 |
| 43 |
| 47 |
+----+

```

And almost (without 2, the only even prime number) the same result with joins:

```

SELECT s1.seq FROM seq_2_to_50 s1 JOIN seq_2_to_50 s2
WHERE s1.seq > s2.seq AND s1.seq % s2.seq <> 0
GROUP BY s1.seq HAVING s1.seq - COUNT(*) = 2;
+----+
| seq |
+----+
| 3 |
| 5 |
| 7 |
| 11 |
| 13 |
| 17 |
| 19 |
| 23 |
| 29 |
| 31 |
| 37 |
| 41 |
| 43 |
| 47 |
+----+

```

Sequence tables can also be useful in date calculations. For example, to find the day of the week that a particular date has fallen on over a 40 year period (perhaps for birthday planning ahead!):

```

SELECT DAYNAME('1980-12-05' + INTERVAL (seq) YEAR) day,
    '1980-12-05' + INTERVAL (seq) YEAR date FROM seq_0_to_40;
+-----+-----+
| day      | date       |
+-----+-----+
| Friday   | 1980-12-05 |
| Saturday | 1981-12-05 |
| Sunday   | 1982-12-05 |
...
| Friday   | 2014-12-05 |
| Saturday | 2015-12-05 |
| Monday   | 2016-12-05 |
| Tuesday  | 2017-12-05 |
| Wednesday| 2018-12-05 |
| Thursday | 2019-12-05 |
| Saturday | 2020-12-05 |
+-----+-----+

```

Although Sequence tables can only directly make use of positive integers, they can indirectly be used to return negative results by making use of the [CAST](#) statement. For example:

```
SELECT CAST(seq AS INT) - 5 x FROM seq_5_to_1;
+----+
| x |
+----+
| 0 |
| -1 |
| -2 |
| -3 |
| -4 |
+----+
```

`CAST` is required to avoid a

```
BIGINT UNSIGNED value is out of range  
error.
```

Sequence tables, while virtual, are still tables, so they must be in a database. This means that a default database must be selected (for example, via the `USE` command) to be able to query a Sequence table. The `information_schema` database cannot be used as the default for a Sequence table.

## Table Name Conflicts

If the SEQUENCE storage engine is installed, it is not possible to create a table with a name which follows the SEQUENCE pattern:

```
CREATE TABLE seq_1_to_100 (col INT) ENGINE = InnoDB;  
ERROR 1050 (42S01): Table 'seq_1_to_100' already exists
```

However, a SEQUENCE table can be converted to another engine and the new table can be referred in any statement:

```
ALTER TABLE seq_1_to_100 ENGINE = BLACKHOLE;  
  
SELECT * FROM seq_1_to_100;  
Empty set (0.00 sec)
```

While a SEQUENCE table cannot be dropped, it is possible to drop the converted table. The SEQUENCE table with the same name will still exist:

```
DROP TABLE seq_1_to_100;  
  
SELECT COUNT(*) FROM seq_1_to_100;  
+----+  
| COUNT(*) |  
+----+  
| 100 |  
+----+  
1 row in set (0.00 sec)
```

A temporary table with a SEQUENCE-like name can always be created and used:

```
CREATE TEMPORARY TABLE seq_1_to_100 (col INT) ENGINE = InnoDB;  
  
SELECT * FROM seq_1_to_100;  
Empty set (0.00 sec)
```

## Resources

- [Sometimes its the little things](#) - Dean Ellis tries out the Sequence engine.
- [MariaDB's Sequence Storage Engine](#) - Federico Razzoli writes more in-depth on the engine.

## 4.3.18 SphinxSE

SphinxSE is a storage engine that talks to searchd (Sphinx daemon) to enable text searching.



### About SphinxSE

[About the Sphinx Storage Engine](#)



### Installing Sphinx

[Installing the Sphinx daemon](#)



### Configuring Sphinx

[Before you can get Sphinx working with the Sphinx Storage Engine on MariaDB...](#)



## Installing and Testing SphinxSE with MariaDB

[How to install and test SphinxSE](#)



## Sphinx Status Variables

[Sphinx status variables.](#)

There are [3 related questions](#).

### 4.3.18.1 About SphinxSE

#### Contents

1. [Versions of SphinxSE in MariaDB](#)
2. [Enabling SphinxSE in MariaDB](#)
3. [Using SphinxSE](#)
  1. [Basic Usage](#)
  2. [Search Options](#)
  3. [SHOW ENGINE SPHINX STATUS](#)
  4. [JOINS with SphinxSE](#)
4. [Building snippets \(excerpts\) via MariaDB](#)
5. [More Information](#)

The Sphinx storage engine (SphinxSE) is a storage engine that talks to searchd (the Sphinx daemon) to enable text searching. Sphinx and SphinxSE is used as a faster and more customizable alternative to MariaDB's [built-in full-text search](#).

Sphinx does not depend on MariaDB, and can run independently, but SphinxSE provides a convenient interface to the underlying Sphinx daemon.

#### Versions of SphinxSE in MariaDB

SphinxSE Version	Introduced	Maturity
SphinxSE 2.2.6	<a href="#">MariaDB 10.0.15</a>	Stable
SphinxSE 2.1.9	<a href="#">MariaDB 10.0.14</a>	Stable
SphinxSE 2.0.4	<a href="#">MariaDB 5.5</a>	
SphinxSE 0.99	<a href="#">MariaDB 5.2</a> and <a href="#">MariaDB 5.3</a>	

#### Enabling SphinxSE in MariaDB

The Sphinx storage engine is included in the source, binaries, and packages of MariaDB. SphinxSE is built as a dynamically loadable .so plugin. To use it, you need to perform a one-time install:

```
INSTALL SONAME 'ha_sphinx';
```

MariaDB until 10.0

In Debian/Ubuntu packages SphinxSE is statically compiled into the MariaDB server, there is no need to use the

```
INSTALL SONAME  
statement.
```

Once installed, SphinxSE will show up in the list of installed storage engines:

```
SHOW ENGINES;  
+-----+-----+-----+-----+-----+  
| Engine | Support | Comment | Transactions | XA | Savepoints |  
+-----+-----+-----+-----+-----+  
| SPHINX | YES | Sphinx storage engine 0.9.9 | NO | NO | NO |  
+-----+-----+-----+-----+-----+
```

This is a one-time step and will not need to be performed again.

**Note:** SphinxSE is just the storage engine part of Sphinx. You will have to [install Sphinx](#) itself in order to make use of SphinxSE in MariaDB.

Despite the name, SphinxSE does not actually store any data itself. It is actually a built-in client which allows MariaDB to talk to Sphinx, run search

queries, and obtain search results. All indexing and searching happen outside MariaDB.

Some SphinxSE applications include:

- easier porting of MariaDB/MySQL FTS applications to Sphinx
- allowing Sphinx use with programming languages for which native APIs are not available yet
- optimizations when additional Sphinx result set processing on the MariaDB side is required (eg. JOINs with original document tables, additional MariaDB-side filtering, and etc...)

## Using SphinxSE

### Basic Usage

To search via SphinxSE, you would need to create a special

```
ENGINE=SPHINX  
"search table", and then  
SELECT  
from it with full text query put into the  
WHERE  
clause for query column.
```

Here is an example create statement and search query:

```
CREATE TABLE t1  
(  
    id      BIGINT UNSIGNED NOT NULL,  
    weight  INTEGER NOT NULL,  
    query   VARCHAR(3072) NOT NULL,  
    group_id INTEGER,  
    INDEX(query)  
) ENGINE=SPHINX CONNECTION="sphinx://127.0.0.1:9312/test1";  
  
SELECT * FROM t1 WHERE query='test it;mode=any';
```

The first three columns of the search table must have a type of

```
BIGINT  
for the 1st column (document id),  
INTEGER  
or  
BIGINT  
for the 2nd column (match weight), and  
VARCHAR  
or  
TEXT
```

for the 3rd column (your query), respectively. This mapping is fixed; you cannot omit any of these three required columns, or move them around, or change types. Also, the query column must be indexed; all the others must be kept unindexed. Column names are ignored so you can use arbitrary ones.

Additional columns must be either

```
INTEGER  
,  
TIMESTAMP  
,  
BIGINT  
,  
VARCHAR  
, or  
FLOAT  
. They will be bound to the attributes provided in the Sphinx result set by name, so their names must match the attribute names specified in sphinx.conf  
. If there's no such attribute name in the Sphinx search results, the additional columns will have  
NULL  
values.
```

Special "virtual" attribute names can also be bound to SphinxSE columns.

```
_sph_  
needs to be used instead of  
@  
for that. For instance, to obtain the values of '  
@groupby  
','  
@count
```

```
', or
@distinct
' virtual attributes, use '
_sph_groupby
',
_sph_count
' or '
_sph_distinct
' column names, respectively.
```

The

```
CONNECTION
string parameter is used to specify the default
searchd
host, port, and indexes for queries issued using this table. If no connection string is specified in
CREATE
TABLE
, index name'
*
'(ie. search all indexes) and'
127.0.0.1:9312
' are assumed. The connection string syntax is as follows:
```

```
CONNECTION="sphinx://HOST:PORT/INDEXNAME"
```

You can change the default connection string later like so:

```
ALTER TABLE t1 CONNECTION="sphinx://NEWHOST:NEWPORT/NEWINDEXNAME";
```

You can also override all these parameters per-query.

**Note:** To use Linux sockets you can modify the searchd section of the Sphinx configuration file, setting the listen parameter to a socket file. Instruct SphinxSE about the socket using CONNECTION="unix: /path/to/socket[:index]".

## Search Options

As seen in the example above, both query text and search options should be put into the '

```
WHERE
' clause of the search query column (i.e. the 3rd column); the options are separated by semicolons (';
') and separate names from values using an equals sign ('=
'). Any number of options can be specified. Available options are:
```

- query - query text;
- mode - matching mode. Must be one of "all", "any", "phrase", "boolean", or "extended". Default is "all";
- sort - match sorting mode. Must be one of "relevance", "attr\_desc", "attr\_asc", "time\_segments", or "extended". In all modes besides "relevance" attribute name (or sorting clause for "extended") is also required after a colon:

```
... WHERE query='test;sort=attr_asc:group_id';
... WHERE query='test;sort=extended:@weight desc, group_id asc';
```

- offset - offset into result set, default is 0;
- limit - amount of matches to retrieve from result set, default is 20;
- index - names of the indexes to search:

```
... WHERE query='test;index=test1';
... WHERE query='test;index=test1,test2,test3';
```

- minid, maxid - min and max document ID to match;
- weights - comma-separated list of weights to be assigned to Sphinx full-text fields:

```
... WHERE query='test;weights=1,2,3';
```

- filter, !filter - comma-separated attribute name and a set of values to match:

```
# only include groups 1, 5 and 19
... WHERE query='test;filter=group_id,1,5,19;';

# exclude groups 3 and 11
... WHERE query='test;!filter=group_id,3,11;';
```

- range, !range - comma-separated attribute name, min and max value to match:

```
# include groups from 3 to 7, inclusive
... WHERE query='test;range=group_id,3,7;';

# exclude groups from 5 to 25
... WHERE query='test;!range=group_id,5,25;';
```

- maxmatches - per-query max matches value:

```
... WHERE query='test;maxmatches=2000;';
```

- groupby - group-by function and attribute:

```
... WHERE query='test;groupby=day:published_ts;';
... WHERE query='test;groupby=attr:group_id';
```

- groupsort - group-by sorting clause:

```
... WHERE query='test;groupsort=@count desc;';
```

- indexweights - comma-separated list of index names and weights to use when searching through several indexes:

```
... WHERE query='test;indexweights=idx_exact,2, idx_stemmed,1;';
```

- comment - a string to mark this query in query log (mapping to \$comment parameter in Query() API call):

```
... WHERE query='test;comment=marker001;';
```

- select - a string with expressions to compute (mapping to SetSelect() API call):

```
... WHERE query='test;select=2*a+3*b as myexpr;';
```

**Note:** It is much more efficient to allow Sphinx to perform sorting, filtering, and slicing of the result set than to raise max matches count and use 'WHERE

```
''
ORDER BY
', and'
LIMIT
```

' clauses on the MariaDB side. This is for two reasons:

1. Sphinx does a number of optimizations and performs better than MariaDB/MySQL on these tasks.
2. Less data would need to be packed by

```
searchd
, and transferred and unpacked by SphinxSE.
```

## SHOW ENGINE SPHINX STATUS

Starting with version 0.9.9-rc1, additional query info besides the result set can be retrieved with the 'SHOW ENGINE SPHINX STATUS' statement:

```
SHOW ENGINE SPHINX STATUS;
+-----+-----+
| Type   | Name    | Status          |
+-----+-----+
| SPHINX | stats   | total: 25, total found: 25, time: 126, words: 2 |
| SPHINX | words   | sphinx:591:1256 soft:11076:15945
+-----+-----+
```

This information can also be accessed through status variables. Note that this method does not require super-user privileges.

```
SHOW STATUS LIKE 'sphinx %';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| sphinx_total | 25 |
| sphinx_total_found | 25 |
| sphinx_time | 126 |
| sphinx_word_count | 2 |
| sphinx_words | sphinx:591:1256 soft:11076:15945 |
+-----+-----+
```

## JOINs with SphinxSE

You can perform

`JOIN`

`s` on a SphinxSE search table and tables using other engines. Here's an example with "documents" from example.sql:

```
SELECT content, date_added FROM test.documents docs
  JOIN t1 ON (docs.id=t1.id)
 WHERE query="one document;mode=any";
+-----+-----+
| content | docdate |
+-----+-----+
| this is my test document number two | 2006-06-17 14:04:28 |
| this is my test document number one | 2006-06-17 14:04:28 |
+-----+-----+

SHOW ENGINE SPHINX STATUS;
+-----+-----+
| Type | Name | Status |
+-----+-----+
| SPHINX | stats | total: 2, total found: 2, time: 0, words: 2 |
| SPHINX | words | one:1:2 document:2:2 |
+-----+-----+
```

## Building snippets (excerpts) via MariaDB

Starting with version 0.9.9-rc2, SphinxSE also includes a UDF function that lets you create snippets through MariaDB. The functionality is fully similar to the [BuildExcerpts](#) API call but is accessible through MariaDB+SphinxSE.

### MariaDB until 5.5

The binary that provides the UDF is named `sphinx.so` and is automatically built and installed to the proper location along with SphinxSE itself. Register the UDF using the following statement:

```
CREATE FUNCTION sphinx_snippets RETURNS STRING SONAME 'sphinx.so';
```

### MariaDB until 10.0

The UDF is packed together with the storage engine, in the same binary named `ha_sphinx.so`. Register the UDF using the following statement:

```
CREATE FUNCTION sphinx_snippets RETURNS STRING SONAME 'ha_sphinx.so';
```

The function name must be '

`sphinx_snippets`  
' , you can not use an arbitrary name. Function arguments are as follows:

```
Prototype: function sphinx_snippets ( document, index, words, [options] );
```

Document and words arguments can be either strings or table columns. Options must be specified like this: `<code>'value' AS option_name</code>`. For a list of supported options, refer to the [BuildExcerpts\(\)](#) API call. The only UDF-specific additional option is named 'sphinx' and lets you specify searchd location (host and port).

Usage examples:

```

SELECT sphinx_snippets('hello world doc', 'main', 'world',
    'sphinx://192.168.1.1/' AS sphinx, true AS exact_phrase,
    '[b]' AS before_match, '[/b]' AS after_match)
FROM documents;

SELECT title, sphinx_snippets(text, 'index', 'mysql php') AS text
    FROM sphinx, documents
    WHERE query='mysql php' AND sphinx.id=documents.id;

```

## More Information

More information on Sphinx and SphinxSE is available on the [Sphinx website](#).

### 4.3.18.2 Installing Sphinx

In order to use the [Sphinx Storage Engine](#), it is necessary to install the Sphinx daemon.

Many Linux distributions have Sphinx in their repositories. These can be used to install Sphinx instead of following the instructions below, but these are usually quite old versions and don't all include API's for easy integration. Ubuntu users can use the updated repository at <https://launchpad.net/~builds/+archive/sphinxsearch-rel21> (see instructions below). Alternatively, download from <http://sphinxsearch.com/downloads/release/>

## Debian and Ubuntu

Ubuntu users can make use of the repository, as follows:

```

sudo add-apt-repository ppa:builds/sphinxsearch-rel21
sudo apt-get update
sudo apt-get install sphinxsearch

```

Alternatively, install as follows:

- The Sphinx package and daemon are named sphinxsearch
- sudo apt-get install unixodbc libpq5 mariadb-client
- sudo dpkg -i sphinxsearch\*.deb

- [Configure Sphinx](#) as required
- You may need to check  
`/etc/default/sphinxsearch`  
to see that  
`START=yes`

- Start with  
`sudo service sphinxsearch start`  
(and stop with  
`sudo service sphinxsearch stop`)

## Red Hat and CentOS

- The package name is sphinx and the daemon searchd
- sudo yum install postgresql-libs unixODBC
- sudo rpm -Uhv sphinx\*.rpm
- [Configure Sphinx](#) as required
- service searchd start

# Windows

- Unzip and extract the downloaded zip file
- Move the extracted directory to  
C:\Sphinx
- Configure Sphinx as required
- Install as a service:
  - C:\Sphinx\bin> C:\Sphinx\bin\searchd --install --config C:\Sphinx\sphinx.conf.in --servicename SphinxSearch

Once Sphinx has been installed, it will need to be [configured](#).

Full instructions, including details on compiling Sphinx yourself, are available at <http://sphinxsearch.com/docs/current.html>.

## 4.3.18.3 Configuring Sphinx

Before you can get Sphinx working with the Sphinx Storage Engine on MariaDB, you need to configure it.

- The default configuration file is called

```
sphinx.conf  
, usually located in  
/etc/sphinxsearch  
(Debian/Ubuntu),  
/etc/sphinx/sphinx.conf.  
(Red Hat/CentOS) or  
C:\Sphinx\sphinx.conf  
(Windows).
```

If it doesn't already exist, you can use the sample configuration file,

```
sphinx.conf.dist  
. There is also sample data supplied that we can use for testing. Load the sample data (which creates two tables,  
documents  
and  
tags  
in the  
test  
database), for example:
```

```
mysql -u test < /usr/local/sphinx/etc/example.sql  
(Red Hat, CentOS)  
mysql -u test < /usr/share/doc/sphinxsearch/example-conf/example.sql  
(Debian/Ubuntu)
```

The sample configuration file documents the available options. You will need to make at least a few changes. A MariaDB user with permission to access the database must be created. For example:

```
CREATE USER 'sphinx'@localhost  
IDENTIFIED BY 'sphinx_password';  
GRANT SELECT on test.* to 'sphinx'@localhost;
```

Add these details to the

```
mysql  
section of the config file:
```

```
sql_host = localhost  
sql_user = sphinx  
sql_pass = sphinx_password  
sql_db = test  
sql_port = 3306
```

On Windows, the

```
path  
and  
pid  
lines will need to be changed to reflect a valid path, usually as follows:
```

```
path = C:\Sphinx\docsidx  
...  
pid_file = C:\Sphinx\sphinx.pid
```

The query in the configuration files is the query that will be used for building the index. In the sample data, this is:

```
sql_query = \
    SELECT id, group_id, UNIX_TIMESTAMP(date_added) AS date_added, title, content \
    FROM documents
```

## 4.3.18.4 Installing and Testing SphinxSE with MariaDB

To use SphinxSE with MariaDB you need to first [download and install Sphinx](#).

Complete Sphinx documentation is available on the [Sphinx website](#).

### Tips for Installing Sphinx

#### libexpat

One library we know you will need on Linux before you can install Sphinx is

```
libexpat
. If it is not installed, you will get the warning
checking for libexpat... not found
. On Suse the package is called
libexpat-devel
, on Ubuntu the package is called
libexpat1-dev
.
```

#### MariaDB detection

If you run into problems with MariaDB not being detected, use the following options:

```
--with-mysql      compile with MySQL support (default is enabled)
--with-mysql-includes  path to MySQL header files
--with-mysql-libs    path to MySQL libraries
```

The above will tell the

```
configure
script where your MySQL/MariaDB installation is.
```

### Testing Sphinx

After installing Sphinx, you can check that things are working in MariaDB by doing the following:

```
cd installation-dir/mysql-test
./mysql-test-run --suite=sphinx
```

If the above test doesn't pass, check the logs in the

```
'var'
directory. If there is a problem with the sphinx installation, the reason can probably be found in the log file at:
var/log/sphinx.sphinx/searchd/sphinx.log
.
```

## 4.3.18.5 Sphinx Status Variables

### Contents

1. [Sphinx\\_error](#)
2. [Sphinx\\_time](#)
3. [Sphinx\\_total](#)
4. [Sphinx\\_total\\_found](#)
5. [Sphinx\\_word\\_count](#)
6. [Sphinx\\_words](#)

This page documents status variables related to the [Sphinx storage engine](#). See [Server Status Variables](#) for a complete list of status variables that can be viewed with [SHOW STATUS](#).

See also the [Full list of MariaDB options, system and status variables](#).

Sphinx\_error

- **Description:** See [SHOW ENGINE SPHINX STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

Sphinx\_time

- **Description:** See [SHOW ENGINE SPHINX STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

Sphinx\_total

- **Description:** See [SHOW ENGINE SPHINX STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

Sphinx\_total\_found

- **Description:** See [SHOW ENGINE SPHINX STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

Sphinx\_word\_count

- **Description:** See [SHOW ENGINE SPHINX STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

Sphinx\_words

- **Description:** See [SHOW ENGINE SPHINX STATUS](#) .
  - **Scope:** Global, Session
  - **Data Type:**  
    numeric
- 

## 4.3.19 Spider

The Spider storage engine supports partitioning and [xa transactions](#) , and allows tables of different MariaDB instances to be handled as if they were on the same instance.

## Versions of Spider in MariaDB

Spider Version	Introduced	Maturity
Spider 3.3.15	MariaDB 10.5.7 , MariaDB 10.4.6	Stable
Spider 3.3.15	MariaDB 10.5.4	Gamma
Spider 3.3.14	MariaDB 10.4.3 , MariaDB 10.3.13	Stable
Spider 3.3.13	MariaDB 10.3.7	Stable
Spider 3.3.13	MariaDB 10.3.3	Gamma
Spider 3.2.37	MariaDB 10.1.10 , MariaDB 10.0.23	Gamma
Spider 3.2.21	MariaDB 10.1.5 , MariaDB 10.0.18	Gamma
Spider 3.2.18	MariaDB 10.0.17	Gamma
Spider 3.2.11	MariaDB 10.0.14	Gamma
Spider 3.2.4	MariaDB 10.0.12	Gamma
Spider 3.2	MariaDB 10.0.11	Gamma
Spider 3.0	MariaDB 10.0.4	Beta

## Spider Documentation

See the [spider-2.0-doc](#) repository for complete, older, documentation.

[Presentation for new sharding features in Spider 3.3](#) .



### Spider Storage Engine Overview

*Storage engine with sharding features.*



### Spider Installation

*Setting up Spider.*



### Spider Storage Engine Core Concepts

*Key Spider concepts*



### Spider Use Cases

*Basic working examples for Spider*



### Spider Cluster Management

*Spider Cluster Management*



### Spider Feature Matrix

*Matrix of Spider features*



### Spider Server System Variables

*System variables for the Spider storage engine.*



### Spider Table System Variables

*Spider variables available in the CREATE TABLE ... COMMENT clause*



### Spider Status Variables

*Spider server status variables.*



### Spider Functions

*User-defined functions available with the Spider storage engine.*



### Spider mysql Database Tables

*System tables related to the Spider storage engine.*



### Information Schema SPIDER\_ALLOC\_MEM Table

*Information about Spider's memory usage.*



### Information Schema SPIDER\_WRAPPER\_PROTOCOLS Table

*Installed along with the Spider storage engine.*



### Spider Differences Between SpiderForMySQL and MariaDB

*Spider differences between MySQL and MariaDB*



## **Spider Case Studies**

*List of clients using Spider*



## **Spider Benchmarks**

*Benchmarks for Spider*



## **Spider FAQ**

*Frequently-asked questions about the Spider storage engine*

There are 5 related questions .

### **4.3.19.10 Spider Functions**

#### **4.3.19.10.1 SPIDER\_BG\_DIRECT\_SQL**

#### **4.3.19.10.2 SPIDER\_COPY\_TABLES**

#### **4.3.19.10.3 SPIDER\_DIRECT\_SQL**

#### **4.3.19.10.4 SPIDER\_FLUSH\_TABLE\_MON\_CACHE**

### **4.3.19.11 Spider mysql Database Tables**

#### **4.3.19.11.1 mysqlspider\_link\_failed\_log Table**

#### **4.3.19.11.2 mysqlspider\_link\_mon\_servers Table**

#### **4.3.19.11.3 mysqlspider\_tables Table**

#### **4.3.19.11.4 mysqlspider\_table\_crd Table**

#### **4.3.19.11.5 mysqlspider\_table\_position\_for\_recovery Table**

#### **4.3.19.11.6 mysqlspider\_table\_sts Table**

#### **4.3.19.11.7 mysqlspider\_xa Table**

#### **4.3.19.11.8 mysqlspider\_xa\_failed\_log Table**

#### **4.3.19.11.9 mysqlspider\_xa\_member Table**

### **4.3.19.12 Information Schema SPIDER\_ALLOC\_MEM Table**

### **4.3.19.13 Information Schema SPIDER\_WRAPPER\_PROTOCOLS Table**

### **4.3.19.14 Spider Differences Between SpiderForMySQL and MariaDB**

## Contents

1. SQL Syntax
2. Features

## SQL Syntax

- With

SpiderForMySQL  
, the **CREATE TABLE** statement uses  
CONNECTION  
to define spider table variables whereas MariaDB uses  
COMMENT

## Features

- 

**HANDLER**

can not be translated to SQL in MariaDB

- Concurrent background search is not yet implemented in MariaDB
- Vertical partitioning storage engine VP is not implemented in MariaDB

- 

**CREATE TABLE**

can use **table discovery** in MariaDB

- 

**JOIN**

performance improvement using

**join\_cache\_level**

>1

and

**join\_buffer\_size**

in MariaDB

## 4.3.19.15 Spider Case Studies

A list of users or clients that are using Spider and agree to be referenced:

- Tencent Games. They handle 100TB data on 396 Spider nodes and 2800 data nodes. They use this cluster for their online games.
- Kadokawa Corporation
- MicroAd, Inc.
- Sansan, Inc.
- teamLab Inc.
- CCM Benchmark <http://www.slideshare.net/skysql/ccm-escape-case-study-skysql-paris-meetup-17122013>
- Softlink <http://fr.slideshare.net/skysql/galaxy-big-data-with-mariadb>
- Gavo <http://wiki.ivoa.net/internal/IVOA/InterOpMay2014NewTechnologies/Spider-MariaDB.pdf>
- Blablacar Using for storing various logs
- Believe Digital Using for back office analytics queries to aggregate multi billions tables in real time

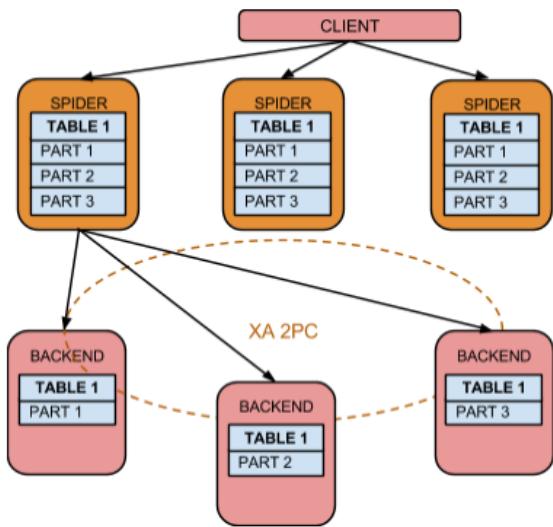
## 4.3.19.16 Spider Benchmarks

This is best run on a cluster of 3 nodes intel NUC servers 12 virtual cores model name : Intel ® Core(TM) i3-3217U CPU @ 1.80GHz

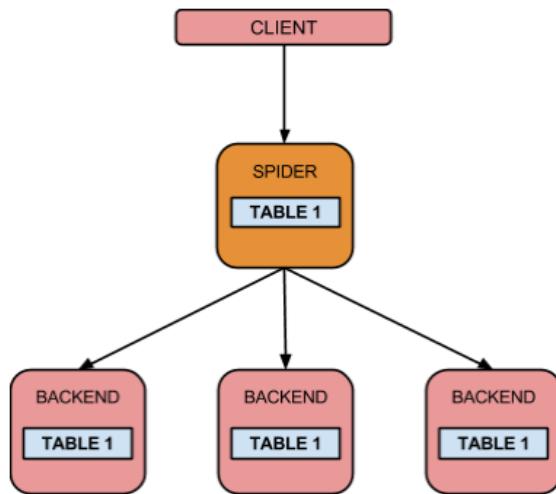
All nodes have been running a mysqlslap client attached to the local spider node in the best run.

```
/usr/local/skysql/mysql-client/bin/mysqlslap --user=skysql --password=skyvodka --host=192.168.0.201 --port=5012 -i1000000 -c32 -
```

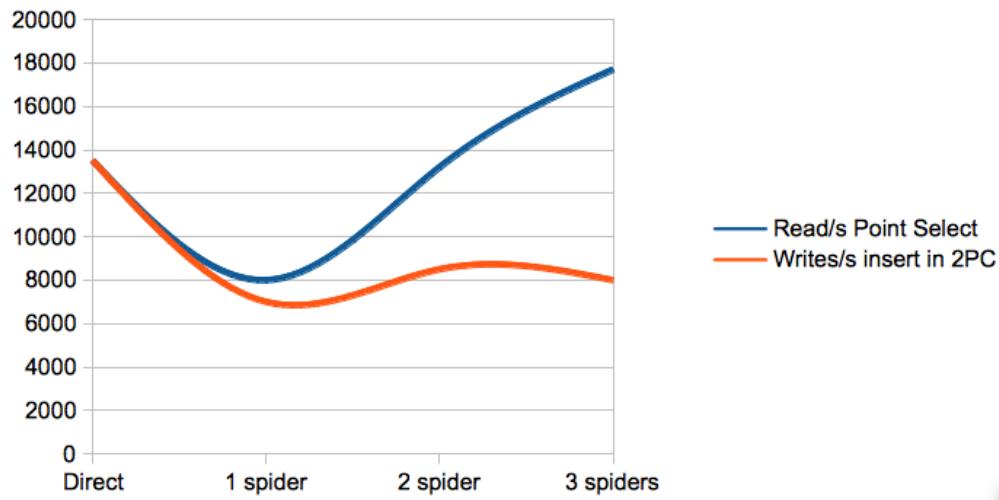
```
spider_conn_recycle_mode=1;
```



The read point select is produce with a 10M rows sysbench table

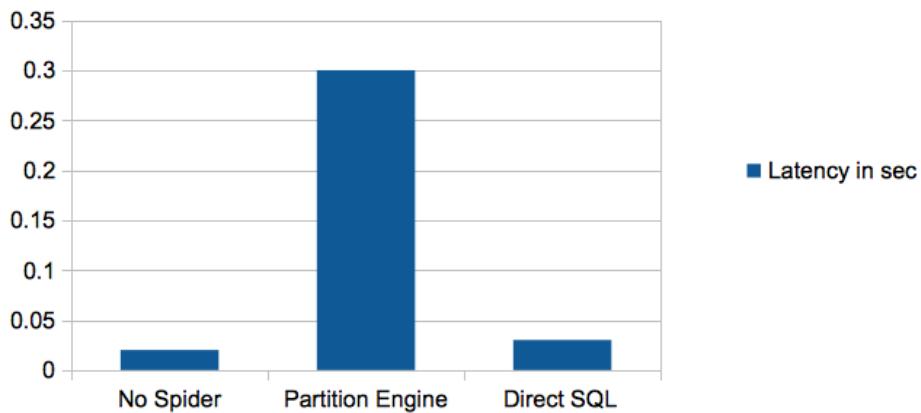


The write insert a single string into a memory table



Before Engine Condition Push Down patch .

## 1000 point select IN(...)



Spider can benefit by 10% additional performance with Independent Storage Engine Statistics.

```
set global use_stat_tables='preferably';
USE backend;
ANALYZE TABLE sbtest;
```

## 4.3.19.17 Spider FAQ

### Contents

1. [What does "\[ERROR\] mysqld: Can't find record in 'spider\\_tables'" mean?](#)
2. [Are there minimum Spider settings?](#)
3. [What does "select spider\\_ping\\_table\(\)" in the general log mean?](#)
4. [Do I need a primary key on physical tables?](#)
5. [Can I use Spider on top of Galera shards?](#)
6. [What are the most used architectures for Spider HA?](#)
7. [What are the most used architectures for Spider Map Reduce?](#)
8. [What about Grants on shards?](#)

### What does "[ERROR] mysqld: Can't find record in 'spider\_tables'" mean?

This happens when you have a Spider table defined that does not point to an existing table on a data node.

### Are there minimum Spider settings?

```
myisam-recover=FORCE,BACKUP
```

MariaDB until 10.1.1

```
optimizer_switch='engine_condition_pushdown=on'
```

MariaDB until 10.3.7

When using spider\_autoincrement\_mode = 0, partitioned Spider tables work as spider\_autoincrement\_mode = 1 see : [MDEV-21404](#)

### What does "select spider\_ping\_table()" in the general log mean?

This is used by Spider monitoring to ask other monitoring nodes the status of a table.

### Do I need a primary key on physical tables?

Not having a primary key will generate errors for resynchronizing tables via spider\_copy\_table().

## Can I use Spider on top of Galera shards?

Yes, XA transactions can be disabled from Spider. Until Galera 4.0 fully supports xa transactions, spider can point to a maxscale proxy that can manage transparent node election in case of failure inside a shard group. Note that disabling XA will break cross shard WRITES in case of transaction ROLLBACK. This architecture need to be used with care if you have a highly transactional workload that can generate cross shard deadlocks.

## What are the most used architectures for Spider HA?

- Delegation of shard node replication using asynchronous replication and slave election with GTID.
- Delegation of shard node replication via active passive HA solutions.
- Shard builds via replication into Spider tables is interesting when you can route READS to a pool of Spider nodes reattaching the shards.

## What are the most used architectures for Spider Map Reduce?

- Map reduce in Spider is limited to a single table. Building spider on top of some views can eliminate the need to use joins.
- Replication to universal tables to every shard is commonly used to enable the views on each shard.

## What about Grants on shards?

- When using MRR and BKA (and you do so with network storage), when Spider needs to create temporary tables on the backends, use the CREATE TEMPORARY TABLES privilege. Spider can still switch to a lower performance solution using `spider_bka_mode=2`, or Query push down or range predicate using `spider_bka_mode=0`

## 4.3.20 Information Schema ENGINES Table

## 4.3.21 PERFORMANCE\_SCHEMA Storage Engine

## 4.3.22 Storage Engine Development

### 4.3.22.1 Storage Engine FAQ

#### Are storage engines designed for MySQL compatible with MariaDB?

In most cases, yes. MariaDB tries to keep API compatibility with MySQL, even across major versions.

#### Will storage engines created for MariaDB work in MySQL?

It will mostly work. It would need #ifdef's to adjust to MySQL-5.6 API, for example, for multi-read-range API, for table discovery API, etc. But most of the code will work as is, without any changes.

#### Do storage engine binaries need to be recompiled for MariaDB?

Yes. You will need to recompile the storage engine against the exact version of MySQL or MariaDB you intend to run it on. This is due to the version of the server being stored in the storage engine binary, and the server will refuse to load it if it was compiled for a different version.

## 4.3.22.2 Engine-defined New Table/Field/Index Attributes

### Contents

1. [API](#)
2. [SQL](#)
3. [See Also](#)

In MariaDB, a storage engine can allow the user to specify additional attributes per index, field, or table. The engine needs to declare what attributes it introduces.

## API

There are three new members in the

`handlerton`

structure, they can be set in the engine's initialization function as follows:

```
example_hton->table_options= example_table_option_array;  
example_hton->field_options= example_field_option_array;  
example_hton->index_options= example_index_option_array;
```

The arrays are declared statically, as in the following example:

```
static MYSQL_THDVAR ULONG(varopt_default, PLUGIN_VAR_RQCMDARG,
    "default value of the VAROPT table option", NULL, NULL, 5, 0, 100, 0);

struct ha_table_option_struct
{
    char *strparam;
    ulonglong ullparam;
    uint enumparam;
    bool boolparam;
    ulonglong varparam;
};

ha_create_table_option example_table_option_list[]=
{
    HA_TOPTION_NUMBER("NUMBER", ullparam, UINT_MAX32, 0, UINT_MAX32, 10),
    HA_TOPTION_STRING("STR", strparam),
    HA_TOPTION_ENUM("ONE_OR_TWO", enumparam, "one,two", 0),
    HA_TOPTION_BOOL("YESNO", boolparam, 1),
    HA_TOPTION_SYSVAR("VAROPT", varopt, varparam),
    HA_TOPTION_END
};
```

The engine declares a structure

```
ha_table_option_struct
```

that will hold values of these new attributes.

And it describes these attributes to MySQL by creating an array of

```
HA_TOPTION_
```

```
*
```

macros. Note a detail: these macros expect a structure called

```
ha_table_option_struct
```

, if the structure is called differently, a

```
#define
```

will be needed.

There are five supported kinds of attributes:

macro name	attribute value type	corresponding C type	additional parameters of a macro
HA_TOPTION_NUMBER	an integer number	unsigned long long	a default value, minimal allowed value, maximal allowed value, a factor, that any allowed should be a multiple of.
HA_TOPTION_STRING	a string	char *	none. The default value is a null pointer.
HA_TOPTION_ENUM	one value from a list of allowed values	int	unsigned a string with a comma-separated list of allowed values, and a default value as a number, starting from 0.
HA_TOPTION_BOOL	a boolean	bool	a default value
HA_TOPTION_SYSVAR	defined by the system variable	defined by the system variable	system variable name

*Do not use*

enum

*for your  
 HA\_OPTION\_ENUM  
 C structure members, the size of the  
 enum  
 depends on the compiler, and even on the compilation options, and the plugin API uses only types with known storage sizes.*

In all macros the first two parameters are name of the attribute as should be used in SQL in the

```
CREATE TABLE
statement, and the name of the corresponding member of the
ha_table_option_struct
structure.
```

The

HA\_TOPTION\_SYSVAR

stands aside a bit. It does not specify the attribute type or the default value, instead it binds the attribute to a system variable. The attribute type and the range of allowed values will be the same as of the corresponding system variable. The attribute **default value** will be the **current value** of its system variable. And unlike other attribute types that are only stored in the

```
.frm
file if explicitly set in the
```

```
CREATE TABLE
statement, the
```

HA\_TOPTION\_SYSVAR

attributes are always stored. If the system variable value is changed, it will not affect existing tables. Note that for this very reason, if a table was created in the old version of a storage engine, and a new version has introduced a

```
HA_TOPTION_SYSVAR
attribute, the attribute value in the old tables will be the default value of the system variable, not its current value.
```

The array ends with a

```
HA_TOPTION_END
macro.
```

Field and index (key) attributes are declared similarly using

```
HA_FOPTION_*
and
HA_IOPTION_*
macros.
```

When in a

```
CREATE TABLE
statement, the
::create()
handler method is called, the table attributes are available in the
table_arg->s->option_struct
, field attributes - in the
option_struct
member of the individual fields (objects of the
Field
class), index attributes - in the
option_struct
member of the individual keys (objects of the
KEY
class).
```

Additionally, they are available in most other handler methods: the attributes are stored in the

```
.frm
file and on every open MySQL makes them available to the engine by filling the corresponding
option_struct
members of the table, fields, and keys.
```

The

ALTER TABLE

needs a special support from the engine. MySQL compares old and new table definitions to decide whether it needs to rebuild the table or not. As the semantics of the engine declared attributes is unknown, MySQL cannot make this decision by analyzing attribute values - this is delegated to the engine. The

```
HA_CREATE_INFO
structure has three new members:
```

```
ha_table_option_struct *option_struct;           /////< structure with parsed table options
ha_field_option_struct **fields_option_struct;  /////< array of field option structures
ha_index_option_struct **indexes_option_struct; /////< array of index option structures
```

The engine (in the

```
::check_if_incompatible_data()
```

method) is responsible for comparing new values of the attributes from the HA\_CREATE\_INFO structure with the old values from the table and returning COMPATIBLE\_DATA\_NO if they were changed in such a way that requires the table to be rebuilt.

The example of declaring the attributes and comparing the values for the

```
ALTER TABLE  
can be found in the EXAMPLE engine.
```

## SQL

The engine declared attributes can be specified per field, index, or table in the

```
CREATE TABLE  
or  
ALTER TABLE  
. The syntax is the conventional:
```

```
CREATE TABLE ... (  
    field ... [attribute=value [attribute=value ...]],  
    ...  
    index ... [attribute=value [attribute=value ...]],  
    ...  
) ... [attribute=value [attribute=value ...]]
```

All values must be specified as literals, not expressions. The value of a boolean option may be specified as one of YES, NO, ON, OFF, 1, or 0. A string value may be specified either quoted or not, as an identifier (if it is a valid identifier, of course). Compare with the old behavior:

```
CREATE TABLE ... ENGINE=FEDERATED CONNECTION='mysql://root@127.0.0.1';
```

where the value of the ENGINE attribute is specified not quoted, while the value of the CONNECTION is quoted.

When an attribute is set, it will be stored with the table definition and shown in the

```
SHOW
```

```
CREATE
```

```
TABLE
```

```
;
```

. To remove an attribute from a table definition use

```
ALTER
```

```
TABLE
```

to set its value to a

```
DEFAULT
```

The values of unknown attributes or attributes with the illegal values cause an error by default. But with [ALTER TABLE](#) one can change the storage engine and some previously valid attributes may become unknown — to the new engine. They are not removed automatically, though, because the table might be altered back to the first engine, and these attributes will be valid again. Still [SHOW CREATE TABLE](#) will comment these unknown attributes out in the output, otherwise they would make a generated [CREATE TABLE](#) statement invalid.

With the

```
IGNORE_BAD_TABLE_OPTIONS
```

[sql mode](#) this behavior changes. Unknown attributes do not cause an error, they only result in a warning. And [SHOW CREATE TABLE](#) will not comment them out. This mode is implicitly enabled in the replication slave thread.

## See Also

- [Writing Plugins for MariaDB](#)
- [Storage Engines](#)
- [Storage Engine Development](#)

### 4.3.22.3 Table Discovery

In MariaDB it is not always necessary to run an explicit

```
CREATE TABLE
```

statement for a table to appear. Sometimes a table may already exist in the storage engine, but the server does not know about it, because there is no

```
.frm
```

file for this table. This can happen for various reasons; for example, for a cluster engine the table might have been created in the cluster by another MariaDB server node. Or for the engine that supports table shipping a table file might have been simply copied into the MariaDB data directory. But no matter what the reason is, there is a mechanism for an engine to tell the server that the table exists. This mechanism is called **table discovery** and if an engine wants the server to discover its tables, the engine should support the table discovery API.

## Contents

1. [Automatic Discovery](#)
  1. [handlerton::tablefile\\_extensions](#)
  2. [handlerton::discover\\_table\\_names\(\)](#)
  3. [handlerton::discover\\_table\\_existence\(\)](#)
  4. [handlerton::discover\\_table\(\)](#)
  5. [TABLE\\_SHARE::init\\_from\\_binary\\_frm\\_in](#)
  6. [TABLE\\_SHARE::init\\_from\\_sql\\_statemen](#)
  7. [TABLE\\_SHARE::read\\_frm\\_image\(\)](#)
  8. [TABLE\\_SHARE::free\\_frm\\_image\(\)](#)
  9. [HA\\_ERR\\_TABLE\\_DEF\\_CHANGED](#)
  10. [TABLE\\_SHARE::tabledef\\_version](#)
2. [Assisted discovery](#)
  1. [handlerton::discover\\_table\\_structure\(\)](#)
  3. [The role of .frm files](#)

There are two different kinds of table discovery — a fully automatic discovery and a user-assisted one. In the former, the engine can automatically discover the table whenever an SQL statement needs it. In MariaDB, the [Archive](#) and [Sequence](#) engines support this kind of discovery. For example, one can copy a

```
t1.ARZ
```

file into the database directory and immediately start using it — the corresponding

```
.frm
```

file will be created automatically. Or one can select from say, the

```
seq_1_to_10
```

table without any explicit

```
CREATE TABLE
```

```
statement.
```

In the latter, user-assisted, discovery the engine does not have enough information to discover the table all on its own. But it can discover the table structure if the user provides certain additional information. In this case, an explicit

```
CREATE TABLE
```

statement is still necessary, but it should contain no table structure — only the table name and the table attributes. In MariaDB, the [FederatedX](#) storage engine supports this. When creating a table, one only needs to specify the

```
CONNECTION
```

attribute and the table structure — fields and indexes — will be provided automatically by the engine.

## Automatic Discovery

As far as automatic table discovery is concerned, the tables, from the server point of view, may appear, disappear, or change structure anytime. Thus the server needs to be able to ask whether a given table exists and what its structure is. It needs to be notified when a table structure changes outside of the server. And it needs to be able to get a list of all (unknown to the server) tables, for statements like

```
SHOW TABLES
```

. The server does all that by invoking specific methods of the

```
handlerton
```

```
:
```

```
const char **tablefile_extensions;
int (*discover_table_names)(handlerton *hton, LEX_STRING *db, MY_DIR *dir,
                           discovered_list *result);
int (*discover_table_existence)(handlerton *hton, const char *db,
                               const char *table_name);
int (*discover_table)(handlerton *hton, THD* thd, TABLE_SHARE *share);
```

## handlerton::tablefile\_extensions

Engines that store tables in separate files (one table might occupy many files with different extensions, but having the same base file name) should store the list of possible extensions in the **tablefile\_extensions** member of the

```
handlerton
(earlier this list was returned by the
handler::bas_ext()
method). This will significantly simplify the discovery implementation for these engines, as you will see below.
```

## handlerton::discover\_table\_names()

When a user asks for a list of tables in a specific database — for example, by using

```
SHOW TABLES
or by selecting from
INFORMATION_SCHEMA.TABLES
— the server invokes discover_table_names() method of the
handlerton
. For convenience this method, besides the database name in question, gets the list of all files in this database directory, so that the engine can
look for table files without doing any filesystem i/o. All discovered tables should be added to the
result
collector object. It is defined as
```

```
class discovered_list
{
public:
    bool add_table(const char *pname, size_t tlen);
    bool add_file(const char *fname);
};
```

and the engine should call

```
result->add_table()
or
result->add_file()
for every discovered table (use
add_file()
if the name to add is in the MariaDB file name encoding, and
add_table()
if it's a true table name, as shown in
SHOW TABLES
).
```

If the engine is file-based, that is, it has non-empty list in the

```
tablefile_extensions
, this method is optional. For any file-based engine that does not implement
discover_table_names()
, MariaDB will automatically discover the list of all tables of this engine, by looking for files with the extension
tablefile_extensions[0]
```

## handlerton::discover\_table\_existence()

In some rare cases MariaDB needs to know whether a given table exists, but does not particularly care about this table structure (for example, when executing a

```
DROP TABLE
statement). In these cases, the server uses the discover_table_existence() method to find out whether a table with the given name exists in the
engine.
```

This method is optional. For the engine that does not implement it, MariaDB will look for files with the

```
tablefile_extensions[0]
, if possible. But if the engine is not file-based, MariaDB will use the
discover_table()
method to perform a full table discovery. While this will allow determining correctly whether a table exists, a full discovery is usually slower than
the simple existence check. In other words, engines that are not file-based might want to support
discover_table_existence()
method as a useful optimization.
```

## handlerton::discover\_table()

This is the main method of table discovery, the heart of it. The server invokes it when it wants to use the table. The **discover\_table()** method gets the

```
TABLE_SHARE
structure, which is not completely initialized — only the table and the database name (and a path to the table file) are filled in. It should initialize
```

this

```
TABLE_SHARE  
with the desired table structure.
```

MariaDB provides convenient and easy to use helpers that allow the engine to initialize the

```
TABLE_SHARE  
with minimal efforts. They are the  
TABLE_SHARE  
methods  
init_from_binary_frm_image()  
and  
init_from_sql_statement_string()  
.
```

## TABLE\_SHARE::init\_from\_binary\_frm\_image()

This method is used by engines that use "frm shipping" — such as [Archive](#) or NDB Cluster in MySQL. An frm shipping engine reads the frm file for a given table, exactly as it was generated by the server, and stores it internally. Later it can discover the table structure by using this very frm image. In this sense, a separate frm file in the database directory becomes redundant, because a copy of it is stored in the engine.

## TABLE\_SHARE::init\_from\_sql\_statement\_string()

This method allows initializing the

```
TABLE_SHARE  
using a conventional SQL  
CREATE TABLE  
syntax.
```

## TABLE\_SHARE::read\_frm\_image()

Engines that use frm shipping need to get the frm image corresponding to a particular table (typically in the

```
handler::create()  
method). They do it via the read_frm_image() method. It returns an allocated buffer with the binary frm image, that the engine can use the way it needs.
```

## TABLE\_SHARE::free\_frm\_image()

The frm image that was returned by

```
read_frm_image()  
must be freed with the free_frm_image().
```

## HA\_ERR\_TABLE\_DEF\_CHANGED

One of the consequences of automatic discovery is that the table definition might change when the server doesn't expect it to. Between two

```
SELECT  
queries, for example. If this happens, if the engine detects that the server is using an outdated version of the table definition, it should return a  
HA_ERR_TABLE_DEF_CHANGED handler error. Depending on when in the query processing this error has happened, MariaDB will either re-discover  
the table and execute the query with the correct table structure, or abort the query and return an error message to the user.
```

## TABLE\_SHARE::tabledef\_version

The previous paragraph doesn't cover one important question — how can the engine know that the server uses an outdated table definition? The answer is — by checking the **tabledef\_version**, the table definition version. Every table gets a unique

```
tabledef_version  
value. Normally it is generated automatically when a table is created. When a table is discovered the engine can force it to have a specific  
tabledef_version  
value (simply by setting it in the  
TABLE_SHARE  
before calling the  
init_from_binary_frm_image()  
or  
init_from_sql_statement_string()  
methods).
```

Now the engine can compare the table definition version that the server is using (from any handler method it can be accessed as

```
this->table->s->tabledef_version  
) with the version of the actual table definition. If they differ — it is  
HA_ERR_TABLE_DEF_CHANGED  
.
```

Assisted discovery is a lot simpler from the server point of view, a lot more controlled. The table cannot appear or disappear at will, one still needs explicit DDL statements to manipulate it. There is only one new handlerton method that the server uses to discover the table structure when a user has issued an explicit

```
CREATE TABLE  
statement without declaring any columns or indexes.
```

```
int (*discover_table_structure)(handlerton *hton, THD* thd,  
                                TABLE_SHARE *share, HA_CREATE_INFO *info);
```

The assisted discovery API is pretty much independent from the automatic discovery API. An engine can implement either of them or both (or none); there is no requirement to support automatic discovery if only assisted discovery is needed.

### handlerton::discover\_table\_structure()

Much like the

```
discover_table()  
method, the discover_table_structure() handlerton method gets a partially initialized  
TABLE_SHARE  
with the table name, database name, and a path to table files filled in, but without a table structure. Unlike  
discover_table()  
, here the  
TABLE_SHARE  
has all the engine-defined table attributes in the the  
TABLE_SHARE::option_struct  
structure. Based on the values of these attributes the  
discover_table_structure()  
method should initialize the  
TABLE_SHARE  
with the desired set of fields and keys. It can use  
TABLE_SHARE  
helper methods  
init_from_binary_frm_image()  
and  
init_from_sql_statement_string()  
for that.
```

## The role of

### .frm files

Before table discovery was introduced, MariaDB used

```
.frm  
files to store the table definition. But now the engine can store the table definition (if the engine supports automatic discovery, of course), and  
.frm  
files become redundant. Still, the server can use  
.frm  
files for such an engine — but they are no longer the only source of the table definition. Now  
.frm  
files are merely a cache of the table definition, while the original authoritative table definition is stored in the engine. Like any cache, its purpose  
is to reduce discovery attempts for a table. The engine decides whether it makes sense to cache table definition in the  
.frm  
file or not (see the second argument for the  
TABLE_SHARE::init_from_binary_frm_image()  
). For example, the Archive engine uses  
.frm  
cache, while the Sequence engine does not. In other words, MariaDB creates  
.frm  
files for Archive tables, but not for Sequence tables.
```

The cache is completely transparent for a user; MariaDB makes sure that it always stores the actual table definition and invalidates the

```
.frm  
file automatically when it becomes out of date. This can happen, for example, if a user copies a new Archive table into the datadir and forgets to  
delete the  
.frm  
file of the old table with the same name.
```

## 4.3.23 Converting Tables from MyISAM to InnoDB

## Contents

1. [The task](#)
2. [INDEX Issues](#)
3. [Non-INDEX Issues](#)
4. [See Also](#)

## The task

You have decided to change one or more tables from [MyISAM](#) to [InnoDB](#). That should be as simple as

```
ALTER TABLE foo ENGINE=InnoDB
```

- . But you have heard that there might be some subtle issues.

This describes possible issues that may arise and what to do about them.

*Recommendation.* One way to assist in searching for issues in is to do (at least in \*nix)

```
mysqldump --no-data --all-databases >schemas  
egrep 'CREATE|PRIMARY' schemas    # Focusing on PRIMARY KEYS  
egrep 'CREATE|FULLTEXT' schemas   # Looking for FULLTEXT indexes  
egrep 'CREATE|KEY' schemas       # Looking for various combinations of indexes
```

Understanding how the indexes work will help you better understand what might run faster or slower in InnoDB.

## INDEX Issues

(*Most of these Recommendations and some of these Facts have exceptions.*)

*Fact.* Every InnoDB table has a PRIMARY KEY. If you do not provide one, then the first non-NULL UNIQUE key is used. If that can't be done, then a 6-byte, hidden, integer is provided.

*Recommendation.* Look for tables without a PRIMARY KEY. Explicitly specify a PRIMARY KEY, even if it's an artificial AUTO\_INCREMENT. This is not an absolute requirement, but it is a stronger admonishment for InnoDB than for MyISAM. Some day you may need to walk through the table; without an explicit PK, you can't do it.

*Fact.* The fields of the PRIMARY KEY are included in each Secondary key.

- Check for redundant indexes with this in mind.

```
PRIMARY KEY(id),  
INDEX(b), -- effectively the same as INDEX(b, id)  
INDEX(b, id) -- effectively the same as INDEX(b)
```

(Keep one of the INDEXes, not both)

- Note subtle things like

```
PRIMARY KEY(id),  
UNIQUE(b), -- keep for uniqueness constraint  
INDEX(b, id) -- DROP this one
```

- Also, since the PK and the data coexist:

```
PRIMARY KEY(id),  
INDEX(id, b) -- DROP this one; it adds almost nothing
```

*Contrast.* This feature of MyISAM is not available in InnoDB; the value of 'id' will start over at 1 for each different value of 'abc':

```
id INT UNSIGNED NOT NULL AUTO_INCREMENT,  
PRIMARY KEY (abc, id)
```

A way to simulate the MyISAM 'feature' might be something like: What you want is this, but it won't work because it is referencing the table twice:

```
INSERT INTO foo  
(other, id, ...)  
VALUES  
(123, (SELECT MAX(id)+1 FROM foo WHERE other = 123), ...);
```

Instead, you need some variant on this. (You may already have a BEGIN...COMMIT.)

```

BEGIN;
SELECT @id := MAX(id)+1 FROM foo WHERE other = 123 FOR UPDATE;
INSERT INTO foo
    (other, id, ...)
VALUES
    (123, @id, ...);
COMMIT;

```

Having a transaction is mandatory to prevent another thread from grabbing the same id.

*Recommendation.* Look for such PRIMARY KEYS. If you find such, ponder how to change the design. There is no straightforward workaround. However, the following may be ok. (Be sure that the datatype for id is big enough since it won't start over.):

```

id INT UNSIGNED NOT NULL AUTO_INCREMENT,
PRIMARY KEY (abc, id),
UNIQUE(id)

```

*Recommendation.* Keep the PRIMARY KEY short. If you have Secondary keys, remember that they include the fields of the PK. A long PK would make the Secondary keys bulky. Well, maybe not — if there is a lot of overlap in fields. Example:

```

PRIMARY KEY(a,b,c), INDEX(c,b,a)
— no extra bulk.

```

*Recommendation.* Check AUTO\_INCREMENT sizes.

- BIGINT is almost never needed. It wastes at least 4 bytes per row (versus INT).
- Always use UNSIGNED and NOT NULL.
- MEDIUMINT UNSIGNED (16M max) might suffice instead of INT
- Be sure to be pessimistic — it is painful to ALTER.

*Contrast.* "Vertical Partitioning". This is where you artificially split a table to move bulky columns (eg, a BLOB) into another, parallel, table. It is beneficial in MyISAM to avoid stepping over the blob when you don't need to read it. InnoDB stores BLOB and TEXT differently — 767 bytes are in the record, the rest is in some other block. So, it may (or may not) be worth putting the tables back together. Caution: An InnoDB row is limited to 8KB, and the 767 counts against that.

*Fact.* FULLTEXT (prior to MariaDB 10.0.5) and SPATIAL indexes are not available in InnoDB. Note that MyISAM and InnoDB [FULLTEXT indexes](#) use different [stopword](#) lists and different system variables.

*Recommendation.* Search for such indexes. Keep such tables in MyISAM. Better yet, do Vertical Partitioning (see above) to split out the minimum number of columns from InnoDB.

*Fact.* The maximum length of an INDEX is different between the Engines. (This change is not likely to hit you, but watch out.) MyISAM allows 1000 bytes; InnoDB allows 767 bytes, just big enough for a

```

VARCHAR(255) CHARACTER SET utf8.

ERROR 1071 (42000): Specified key was too long; max key length is 767 bytes

```

*Fact.* The PRIMARY KEY is included in the data. Hence, SHOW TABLE STATUS will show and

```

Index_length
of 0 bytes (or 16KB) for a table with no secondary indexes. Otherwise,
Index_length
is the total size for the secondary keys.

```

*Fact.* The PRIMARY KEY is included in the data. Hence, exact match by PK may be a little faster with InnoDB. And, "range" scans by PK are likely to be faster.

*Fact.* A lookup by Secondary Key traverses the secondary key's BTree, grabs the PRIMARY KEY, then traverses the PK's BTree. Hence, secondary key lookups are a little more cumbersome in InnoDB.

*Contrast.* The fields of the PRIMARY KEY are included in each Secondary key. This may lead to "Using index" (in the EXPLAIN plan) for InnoDB for cases where it did not happen in MyISAM. (This is a slight performance boost, and counteracts the double-lookup otherwise needed.) However, when "Using index" would be useful on the PRIMARY KEY, MyISAM would do an "index scan", yet InnoDB effectively has to do a "table scan".

*Same as MyISAM.* Almost always

```

INDEX(a) -- DROP this one because the other one handles it.
INDEX(a,b)

```

*Contrast.* The data is stored in PK order. This means that "recent" records are 'clustered' together at the end. This may give you better 'locality of reference' than in MyISAM.

*Same as MyISAM.* The optimizer almost never uses two indexes in a single SELECT. (5.1 will occasionally do "index merge".) SELECT in subqueries and UNIONs can independently pick indexes.

*Subtle issue.* When you DELETE a row, the AUTO\_INCREMENT id will be burned. Ditto for REPLACE, which is a DELETE plus an INSERT.

*Very subtle issue.* Replication occurs on COMMIT. If you have multiple threads using transactions, the AUTO\_INCREMENTs can arrive at a slave out of

order. One transaction BEGINs, grabs an id. Then another transaction grabs an id but COMMITS before the first finishes.

Same as MyISAM. "Prefix" indexing is usually bad in both InnoDB and MyISAM. Example:

```
INDEX(foo(30))
```

## Non-INDEX Issues

Disk space for InnoDB is likely to be 2-3 times as much as for MyISAM.

MyISAM and InnoDB use RAM radically differently. If you change all your tables, you should make significant adjustments:

- `key_buffer_size` — small but non-zero; say, 10M;
- `innodb_buffer_pool_size` — 70% of available RAM

InnoDB has essentially no need for CHECK, OPTIMIZE, or ANALYZE. Remove them from your maintenance scripts. (No real harm if you keep them.)

Backup scripts may need checking. A MyISAM table can be backed up by copying three files. With InnoDB this is only possible if `innodb_file_per_table` is set to 1. Before MariaDB 10.0, capturing a table or database for copying from production to a development environment was not possible. Change to `mysqldump`. Since MariaDB 10.0 a hot copy can be created - see [Backup and restore overview](#).

Before MariaDB 5.5, the DATA DIRECTORY table option was not supported for InnoDB. Since MariaDB 5.5 it is supported, but only in CREATE TABLE. INDEX DIRECTORY has no effect, since InnoDB does not use separate files for indexes. To better balance the workload through several disks, the paths of some InnoDB log files can also be changed.

Understand autocommit and BEGIN/COMMIT.

- (default) autocommit = 1: In the absence of any BEGIN or COMMIT statements, every statement is a transaction by itself. This is close to the MyISAM behavior, but is not really the best.
- autocommit = 0: COMMIT will close a transaction and start another one. To me, this is kludgy.
- (recommended) BEGIN...COMMIT gives you control over what sequence of operation(s) are to be considered a transaction and "atomic". Include the ROLLBACK statement if you need to undo stuff back to the BEGIN.

Perl's DBIx::DWIW and Java's JDBC have API calls to do BEGIN and COMMIT. These are probably better than 'executing' BEGIN and COMMIT.

Test for errors everywhere! Because InnoDB uses row-level locking, it can stumble into deadlocks that you are not expecting. The engine will automatically ROLLBACK to the BEGIN. The normal recovery is to redo, beginning at the BEGIN. Note that this is a strong reason to have BEGINs.

LOCK/UNLOCK TABLES — remove them. Replace them (sort of) with BEGIN ... COMMIT. (LOCK will work if `innodb_table_locks` is set to 1, but it is less efficient, and may have subtle issues.)

In 5.1, ALTER ONLINE TABLE can speed up some operations significantly. (Normally ALTER TABLE copies the table over and rebuilds the indexes.)

The "limits" on virtually everything are different between MyISAM and InnoDB. Unless you have huge tables, wide rows, lots of indexes, etc, you are unlikely to stumble into a different limit.

Mixture of MyISAM and InnoDB? This is OK. But there are caveats.

- RAM settings should be adjusted to accordingly.
- JOINing tables of different Engines works.
- A transaction that affects tables of both types can ROLLBACK InnoDB changes, but will leave MyISAM changes intact.
- Replication: MyISAM statements are replicated when finished; InnoDB statements are held until the COMMIT.

FIXED (vs DYNAMIC) is meaningless in InnoDB.

PARTITION — You can partition MyISAM and InnoDB tables. Remember the screwball rule: You must either

- have no UNIQUE (or PRIMARY) keys, or
- have the value you are "partitioning on" in every UNIQUE key.

The former is not advised for InnoDB. The latter is messy if you want an AUTO\_INCREMENT.

PRIMARY KEY in PARTITION — Since every key must include the field on which you are PARTITIONing, how can AUTO\_INCREMENT work? Well, there seems to be a convenient special case:

- This works: PRIMARY KEY(autoinc, partition\_key)
- This does not work for InnoDB: PRIMARY KEY(partition\_key, autoinc)

That is, an AUTO\_INCREMENT will correctly increment, and be unique across all PARTITINOs, when it is the first field of the PRIMARY KEY, but not otherwise.

## See Also

Rick James graciously allowed us to use this article in the Knowledge Base.

[Rick James' site](#) has other useful tips, how-tos, optimizations, and debugging tips.

Original source: <http://mysql.rjweb.org/doc.php/myisam2innodb>

## 4.3.24 Machine Learning with MindsDB

## Contents

- 1. Overview
- 2. Installation
- 3. Usage

## Overview

MindsDB is a third-party application that interfaces with MariaDB Server to provide Machine Learning capabilities through SQL. The interface is done via the [Connect Storage Engine](#).

## Installation

To get a functional MariaDB - MindsDB installation, one needs to install the following components:

- **MindsDB** : follow the instructions in the project's [official documentation](#).
- **Connect Storage Engine** must be enabled for the integration to work. See [installing the connect storage engine](#).

MindsDB connects to MariaDB Server via a regular user to setup a dedicated database called

```
mindsdb
. Which user will be used is specified within MindsDB's configuration file.
```

For example, if MindsDB is installed locally, one can create a user called

```
mindsdb@localhost
. MindsDB only authenticates via the mysql_native_password plugin, hence one must set a password for the user:
```

```
CREATE USER mindsdb@localhost;
SET PASSWORD FOR mindsdb@localhost=PASSWORD("password");
```

The user must be granted the global `FILE` privilege and all privileges on the

```
mindsdb
database.
```

```
GRANT FILE ON *.* TO mindsdb@localhost;
GRANT ALL ON mindsdb.* TO mindsdb@localhost;
```

Assuming MindsDB is in the python path one can start up MindsDB with the following parameters:

```
python -m mindsdb --config=$CONFIG_PATH --api=http,mysql
```

Make sure

```
$CONFIG_PATH
points to the appropriate MindsDB configuration file.
```

## Usage

Always consult the project's [official documentation](#) for up-to-date usage scenarios as MindsDB is an actively developed project.

For a step-by-step example, you can consult the following [blog post](#).

If the connection between MindsDB and MariaDB is successful, you should see the

```
mindsdb
database present and two tables within it:
commands
and
predictors
```

MindsDB, as an AutoML framework does all the work when it comes to training the AI model. What is necessary is to pass it the initial data, which MindsDB retrieves via a SELECT statement. This can be done by inserting into the

```
predictors
table.
```

```
INSERT INTO `predictors`
(`name`, `predict`, `select_data_query`)
VALUES ('bikes_model', 'count', 'SELECT * FROM test.bike_data');
```

The values inserted into predictors act as a command instructing MindsDB to:

1. Train a model called 'bikes\_model'
2. From the input data, learn to predict the 'count' column.
3. The input data is generated via the select statement 'SELECT \* FROM test.bike\_data'. The

`select_data_query`  
should be a valid select that MindsDB can run against MariaDB.

## 4.4 Plugins

MariaDB supports the use of plugins, software components that may be added to the core software without having to rebuild the MariaDB server from source code. Therefore, plugins can be loaded at start-up, or loaded and unloaded while the server is running without interruption. Plugins are commonly used for adding desired storage engines, additional security requirements, and logging special information about the server.



### Plugin Overview

*Basics of listing, installing and uninstalling plugins.*



### Information on Plugins

*Information on installed and disabled plugins on a MariaDB Server.*



### Plugin SQL Statements

*List of SQL statements related to plugins.*



### Installing, Using, and Creating Plugins

*Documentation on how to install and enable plugins, as well as create new ones.*



### MariaDB Audit Plugin

*Logging user activity with the MariaDB Audit Plugin.*



### Authentication Plugins

*Authentication plugins allow various authentication methods to be used, and new ones developed.*



### Password Validation Plugins

*Ensuring that user passwords meet certain minimal security requirements.*



### Key Management and Encryption Plugins

*MariaDB uses plugins to handle key management and encryption of data.*



### MariaDB Replication & Cluster Plugins

*Plugins related to MariaDB replication and other replication cluster systems.*



### Storage Engines

*Various storage engines available for MariaDB.*



### Other Plugins

*Information on installing and using other plugins.*

There are [5 related questions](#).

## 4.4.1 Information on Plugins

### 4.4.1.1 List of Plugins

#### Contents

1. [MariaDB Plugin Maturity](#)
2. [See Also](#)

## MariaDB Plugin Maturity

The following table lists the various plugins included in MariaDB ordered by their maturity. Note that maturity will differ across MariaDB versions - see below for an easy way to get a complete list of plugins and their maturity in your version of MariaDB:

Plugin	Version	Maturity	From
Archive	3.0	Stable	
Aria	1.5	Stable	
Audit Plugin	1.4	Stable	<a href="#">MariaDB 10.1.11</a>
aws_key_management	1.0	Stable	<a href="#">MariaDB 10.2.6</a> , <a href="#">MariaDB 10.1.24</a>
binlog	1.0	Stable	
Blackhole	1.0	Stable	

Connect	1.7	Stable	<a href="#">MariaDB 10.4.12</a> , <a href="#">MariaDB 10.3.22</a> , <a href="#">MariaDB 10.2.31</a> , <a href="#">MariaDB 10.1.44</a>
CLIENT_STATISTICS	2.0	Stable	<a href="#">MariaDB 10.1.13</a>
cracklib_password_check	1.0	Stable	<a href="#">MariaDB 10.1.18</a>
CSV	1.0	Stable	
DISKS	1.1	Stable	<a href="#">MariaDB 10.4.7</a> , <a href="#">MariaDB 10.3.17</a> , <a href="#">MariaDB 10.2.26</a> , <a href="#">MariaDB 10.1.41</a>
ed25519	1.1	Stable	<a href="#">MariaDB 10.4.0</a>
FederatedX [1]	2.1	Stable	
Feedback	1.1	Stable	
file_key_management	1.0	Stable	<a href="#">MariaDB 10.1.18</a>
gssapi	1.0	Stable	<a href="#">MariaDB 10.1.15</a>
INDEX_STATISTICS	2.0	Stable	<a href="#">MariaDB 10.1.13</a>
InnoDB	5.7 10.*	Stable	<a href="#">MariaDB 10.2</a>
LOCALES	1.0	Stable	<a href="#">MariaDB 10.1.13</a>
Memory	1.0	Stable	
METADATA_LOCK_INFO	0.1	Stable	<a href="#">MariaDB 10.1.13</a>
MRG_MyISAM	1.0	Stable	
Mroonga	7.7	Stable	<a href="#">MariaDB 10.2.11</a> , <a href="#">MariaDB 10.1.29</a>
MyISAM	1.0	Stable	
MyRocks	1.0	Stable	<a href="#">MariaDB 10.3.7</a> , <a href="#">MariaDB 10.2.16</a>
mysql_native_password	1.0	Stable	
mysql_old_password	1.0	Stable	
named_pipe	1.0	Stable	<a href="#">MariaDB 10.1.11</a>
pam	1.0	Stable	
partition	1.0	Stable	
Performance_Schema	0.1	Stable	
QUERY_CACHE_INFO	1.1	Stable	<a href="#">MariaDB 10.1.13</a>
query_response_time	1.0	Stable	<a href="#">MariaDB 10.1.13</a>
semisync	1.0	Stable	<a href="#">MariaDB 10.1.13</a>
Sequence	1.0	Stable	
SERVER_AUDIT	1.4	Stable	<a href="#">MariaDB 10.1.11</a>
simple_password_check	1.0	Stable	<a href="#">MariaDB 10.1.18</a>
Spider	3.3	Stable	<= <a href="#">MariaDB 10.4</a> , <a href="#">MariaDB 10.5.7</a>
SQL_ERROR_LOG	1.0	Stable	<a href="#">MariaDB 10.1.13</a>
TABLE_STATISTICS	2.0	Stable	<a href="#">MariaDB 10.1.18</a>
USER_STATISTICS	2.0	Stable	<a href="#">MariaDB 10.1.18</a>
user_variables	1.0	Stable	<a href="#">MariaDB 10.3.13</a>
TokuDB	4.0	Stable	
unix_socket	1.0	Stable	
wsrep	1.0	Stable	
WSREP_INFO	1.0	Stable	<a href="#">MariaDB 10.1.18</a>
Plugin	Version	Maturity	From
Federated [2]	1.0	Gamma	
INET6	1.0	Gamma	<a href="#">MariaDB 10.5.9</a>
OQGraph	3.0	Gamma	

S3	1.0	Gamma	MariaDB 10.5.7
Sphinx	2.0	Gamma	
<b>Plugin Version Maturity From</b>			
Columnstore	1.0	Beta	MariaDB 10.5.4
handlersocket	1.0	Beta	
<b>Plugin Version Maturity From</b>			
mysql_json	0.1	Alpha	MariaDB 10.5.7
password_reuse_check	1.0	Alpha	MariaDB 10.7.0
<b>Plugin Version Maturity From</b>			
Cassandra	0.1	Experimental	
debug_key_management	1.0	Experimental	MariaDB 10.1.3
example_key_management	1.0	Experimental	MariaDB 10.1.3

Execute the following on your MariaDB server to get a complete list of plugins and their maturity for your version of MariaDB:

```
SELECT plugin_name, plugin_version, plugin_maturity
FROM information_schema.plugins
ORDER BY plugin_name;
```

## See Also

- [Plugin Overview](#)
- [INSTALL PLUGIN](#)
- [INFORMATION\\_SCHEMA.PLUGINS Table](#)
- [mysql\\_plugin](#)
- [SHOW PLUGINS](#)
- [INSTALL SONAME](#)
- [UNINSTALL PLUGIN](#)
- [UNINSTALL SONAME](#)

## 4.4.1.2 Information Schema PLUGINS Table

## 4.4.1.3 Information Schema ALL\_PLUGINS Table

## 4.4.2 Plugin SQL Statements

## 4.4.3 Installing, Using, and Creating Plugins

### 4.4.3.1 Specifying Which Plugins to Build

By default all plugins are enabled and built as dynamic

```
.so
(or
.dll
) modules. If a plugin does not support dynamic builds, it is not built at all.
```

#### MariaDB 5.5 - 10.0

To specify that a specific plugin should be enabled and built statically into the server executable, use the

```
-DWITH_xxx=1
cmake option (where
xxx
```

is the plugin name). Of course, static linking only works if the plugin itself supports it — some plugins can only be built as dynamic modules.

A plugin will automatically be skipped by cmake if it cannot be built; for example, if some required libraries are missing or if you want to statically link a plugin that only supports dynamic linking.

If you want to avoid building some specific plugin, specify the

```
-DWITHOUT_xxx=1
cmake option.
```

MariaDB starting with 10.1

Use

```
PLUGIN_xxx  
cmake variables. They can be set on the command line with  
-DPLUGIN_xxx=  
  
value
```

or in the cmake gui. Supported values are

Value	Effect
<b>NO</b>	the plugin will be not compiled at all
<b>STATIC</b>	the plugin will be compiled statically, if supported. Otherwise it will be not compiled.
<b>DYNAMIC</b>	the plugin will be compiled dynamically, if supported. Otherwise it will be not compiled. This is the default behavior.
<b>AUTO</b>	the plugin will be compiled statically, if supported. Otherwise it will be compiled dynamically.
<b>YES</b>	same as <b>AUTO</b> , but if plugin prerequisites (for example, specific libraries) are missing, it will not be skipped, it will abort cmake with an error.

Note that unlike autotools, cmake tries to configure and build incrementally. You can modify one configuration option and cmake will only rebuild the part of the tree affected by it. For example, when you do

```
cmake -DWITH_EMBEDDED_SERVER=1  
in the already-built tree, it will make libmysqld to be built, but no other configuration options will be changed or reset to their default values.
```

In particular this means that if you have run, for example

MariaDB 5.5 - 10.0

```
cmake -DWITHOUT_OQGRAPH=1
```

MariaDB starting with 10.1

```
cmake -DPLUGIN_OQGRAPH=NO
```

and later you want to restore the default behavior (with OQGraph being built) in the same build tree, you would need to run

MariaDB 5.5 - 10.0

```
cmake -UWITHOUT_OQGRAPH
```

MariaDB starting with 10.1

```
cmake -DPLUGIN_OQGRAPH=DYNAMIC
```

Alternatively, you might simply delete the

CMakeCache.txt  
file — this is the file where cmake stores current build configuration — and rebuild everything from scratch.

#### 4.4.3.2 Writing Plugins for MariaDB

##### Contents

1. [About](#)
2. [Authentication Plugins](#)
3. [Storage Engine Plugins](#)
4. [Information Schema Plugins](#)
5. [Encryption Plugins](#)
6. [Plugin Declaration Structure](#)
  1. [Example Plugin Declaration](#)

# About

Generally speaking, writing plugins for MariaDB is very similar to [writing plugins for MySQL](#).

## Authentication Plugins

See [Pluggable Authentication](#).

## Storage Engine Plugins

Storage engines can extend

```
CREATE TABLE  
syntax with optional index, field, and table attribute clauses. See Extending CREATE TABLE for more information.
```

See [Storage Engine Development](#).

## Information Schema Plugins

Information Schema plugins can have their own [FLUSH](#) and [SHOW](#) statements. See [FLUSH and SHOW for Information Schema plugins](#).

## Encryption Plugins

[Encryption plugins](#) in MariaDB are used for the [data at rest encryption](#) feature. They are responsible for both key management and for the actual encryption and decryption of data.

## Plugin Declaration Structure

The MariaDB plugin declaration differs from the MySQL plugin declaration in the following ways:

1. it has no useless 'reserved' field (the very last field in the MySQL plugin declaration)
2. it has a 'maturity' declaration
3. it has a field for a text representation of the version field

MariaDB can load plugins that only have the MySQL plugin declaration but both

```
PLUGIN_MATURITY  
and  
PLUGIN_AUTH_VERSION  
will show up as 'Unknown' in the INFORMATION\_SCHEMA.PLUGINS table.
```

For compiled-in (not dynamically loaded) plugins, the presence of the MariaDB plugin declaration is mandatory.

## Example Plugin Declaration

The MariaDB plugin declaration looks like this:

```
/* MariaDB plugin declaration */  
maria_declare_plugin(example)  
{  
    MYSQL_STORAGE_ENGINE_PLUGIN, /* the plugin type (see include/mysql/plugin.h) */  
    &example_storage_engine_info, /* pointer to type-specific plugin descriptor */  
    "EXAMPLEDB", /* plugin name */  
    "John Smith", /* plugin author */  
    "Example of plugin interface", /* the plugin description */  
    PLUGIN_LICENSE_GPL, /* the plugin license (see include/mysql/plugin.h) */  
    example_init_func, /* Pointer to plugin initialization function */  
    example_deinit_func, /* Pointer to plugin deinitialization function */  
    0x0001 /* Numeric version 0xAABB means AA.BB version */,  
    example_status_variables, /* Status variables */  
    example_system_variables, /* System variables */  
    "0.1 example", /* String version representation */  
    MariaDB_PLUGIN_MATURITY_EXPERIMENTAL /* Maturity (see include/mysql/plugin.h) */  
}  
maria_declare_plugin_end;
```

## 4.4.4 MariaDB Audit Plugin

## Contents

1. [Additional documentation](#)
2. [Tutorials](#)
3. [Web Log Articles](#)
4. [Sub-Documents](#)

MariaDB and MySQL are used in a broad range of environments, but if you needed to record user access to be in compliance with auditing regulations for your organization, you would previously have had to use other database solutions. To meet this need, though, MariaDB has developed the MariaDB Audit Plugin. Although the MariaDB Audit Plugin has some unique features available only for MariaDB, it can be used also with MySQL.

Basically, the purpose of the MariaDB Audit Plugin is to log the server's activity. For each client session, it records who connected to the server (i.e., user name and host), what queries were executed, and which tables were accessed and server variables that were changed. This information is stored in a rotating log file or it may be sent to the local `syslog`.

The MariaDB Audit Plugin works with MariaDB, MySQL (as of, version 5.5.34 and 10.0.7) and Percona Server. MariaDB started including by default the Audit Plugin from versions 10.0.10 and 5.5.37, and it can be installed in any version from [MariaDB 5.5.20](#).

### Additional documentation

Below are links to additional documentation on the MariaDB Audit Plugin. They explain in detail how to install, configure and use the Audit Plugin.

- [Installation](#)
- [Configuration](#)
- [Log Settings](#)
- [Log Location & Rotation](#)
- [Log Format](#)
- [Status Variables](#)
- [System Variables](#)
- [Release Notes](#)

### Tutorials

Below are links to some tutorials on MariaDB's site and other sites. They may help you to get more out of the MariaDB Audit Plugin.

- [Introducing the MariaDB Audit Plugin](#)  
by Anatoliy Dimitrov, September 2, 2014
- [Activating MariaDB Audit Log](#) by Jaykishan Mutkawoa, May 30, 2016
- [Installing MariaDB Audit Plugin on Amazon RDS](#)  
Amazon RDS supports using the MariaDB Audit Plugin on MySQL and MariaDB database instances.

### Web Log Articles

Below are links to web log articles on the MariaDB Audit Plugin. You may find them useful in understanding better how to use the Audit Plugin. Since some of these articles are older, they won't include changes and improvements in newer versions. You can rely on the documentation pages listed above for the most current information.

- [Activating Auditing for MariaDB in 5 Minutes](#)  
by Ralf Gebhardt, September 29, 2013
- [Query and Password Filtering with the MariaDB Audit Plugin](#)  
by Ralf Gebhardt, May 4, 2015
- [Set Up a Remote Log File using rsyslog](#)  
by Ralf Gebhardt, December 16, 2013
- [MySQL Auditing with MariaDB Auditing Plugin](#) by Peter Zeitsev, February 15, 2016

### Sub-Documents



#### **MariaDB Audit Plugin - Installation**

*Installing the MariaDB Audit Plugin.*



#### **MariaDB Audit Plugin - Configuration**

*Audit Plugin global variables within MariaDB*



#### **MariaDB Audit Plugin - Log Settings**

*Log audit events to a file or syslog.*



#### **MariaDB Audit Plugin - Location and Rotation of Logs**

*Logs can be written to a separate file or to the system logs*



#### **MariaDB Audit Plugin - Log Format**

*The audit log is a set of records written as a list of fields to a file in plain-text format*



## MariaDB Audit Plugin - Versions

*Releases of the MariaDB Audit Plugin, and in which versions of MariaDB each...*



## MariaDB Audit Plugin Options and System Variables

*Description of Server\_Audit plugin options and system variables.*



## MariaDB Audit Plugin - Status Variables

*Server Audit plugin status variables*



## Release Notes - MariaDB Audit Plugin

*MariaDB Audit Plugin release notes*

There are [7 related questions](#).

### 4.4.4.1 MariaDB Audit Plugin - Installation

The

`server_audit`

plugin logs the server's activity. For each client session, it records who connected to the server (i.e., user name and host), what queries were executed, and which tables were accessed and server variables that were changed. This information is stored in a rotating log file or it may be sent to the local syslogd.

#### Contents

1. [Locating the Plugin](#)
2. [Installing the Plugin](#)
3. [Uninstalling the Plugin](#)
4. [Prohibiting Uninstallation](#)

### Locating the Plugin

The

`server_audit`

plugin's shared library is included in MariaDB packages as the `server_audit.so` or `server_audit.dll` shared library on systems where it can be built.

The plugin must be located in the plugin directory, the directory containing all plugin libraries for MariaDB. The path to this directory is configured by the

`plugin_dir`

system variable. To see the value of this variable and thereby determine the file path of the plugin library, execute the following SQL statement:

```
SHOW GLOBAL VARIABLES LIKE 'plugin_dir';  
  
+-----+  
| Variable_name | Value          |  
+-----+  
| plugin_dir    | /usr/lib64/mysql/plugin/ |  
+-----+
```

Check the directory returned at the filesystem level to make sure that you have a copy of the plugin library,

`server_audit.so`

or

`server_audit.dll`

, depending on your system. It's included in recent installations of MariaDB. If you do not have it, you should upgrade MariaDB.

### Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

`INSTALL SONAME`

or

## INSTALL PLUGIN

. For example:

```
INSTALL SONAME 'server_audit';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = server_audit
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'server_audit';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Prohibiting Uninstallation

The

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

statements may be used to uninstall plugins. For the  
server\_audit

plugin, you might want to disable this capability. To prevent the plugin from being uninstalled, you could set the the

```
server_audit
```

option to

```
FORCE_PLUS_PERMANENT
```

in a relevant server [option group](#) in an [option file](#) after the plugin is loaded once:

```
[mariadb]
...
plugin_load_add = server_audit
server_audit=FORCE_PLUS_PERMANENT
```

Once you've added the option to the server's option file and restarted the server, the plugin can't be uninstalled. If someone tries to uninstall the audit plugin, then an error message will be returned. Below is an example of the error message:

```
UNINSTALL PLUGIN server_audit;

ERROR 1702 (HY000):
Plugin 'server_audit' is force_plus_permanent and can not be unloaded
```

For more information on

[FORCE\\_PLUS\\_PERMANENT](#)

and other option values for the

[server\\_audit](#)

option, see [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

## 4.4.4.2 MariaDB Audit Plugin - Configuration

After the audit plugin has been installed and loaded, there will be some new global variables within MariaDB. These can be used to configure many components, limits, and methods related to auditing the server. You may set these variables related to the logs, such as their location, size limits, rotation parameters, and method of logging information. You may also set what information is logged, such connects, disconnects, and failed attempts to connect. You can also have the audit plugin log queries, read and write access to tables. So as not to overload your logs, the audit plugin can be configured based on lists of users. You can include or exclude the activities of specific users in the logs.

To see a list of [audit plugin-related variables](#) on the server and their values, execute the follow while connected to the server:

```
SHOW GLOBAL VARIABLES LIKE 'server_audit%';

+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| server_audit_events | CONNECT,QUERY,TABLE |
| server_audit_excl_users |          |
| server_audit_file_path | server_audit.log |
| server_audit_file_rotate_now | OFF |
| server_audit_file_rotate_size | 1000000 |
| server_audit_file_rotations | 9 |
| server_audit_incl_users |          |
| server_audit_logging | ON |
| server_audit_mode | 0 |
| server_audit_output_type | file |
| server_audit_query_log_limit | 1024 |
| server_audit_syslog_facility | LOG_USER |
| server_audit_syslog_ident | mysql-server_auditing |
| server_audit_syslog_info |          |
| server_audit_syslog_priority | LOG_INFO |
+-----+-----+
```

The values of these variables can be changed by an administrator with the

[SUPER](#)

privilege, using the

[SET](#)

statement. Below is an example of how to disable audit logging:

```
SET GLOBAL server_audit_logging=OFF;
```

Although it is possible to change all of the variables shown above, some of them may be reset when the server restarts. Therefore, you may want set them in the configuration file (e.g.,

[/etc/my.cnf.d/server.cnf](#)

) to ensure the values are the same after a restart:

```
[server]
...
server_audit_logging=OFF
...
```

For the reason given in the paragraph above, you would not generally set variables related to the auditing plugin using the

SET

statement. However, you might do so to test settings before making them more permanent. Since one cannot always restart the server, you would use the

SET

statement to change immediately the variables and then include the same settings in the configuration file so that the variables are set again as you prefer when the server is restarted.

## Configuring Logs and Setting Other Variables

Of all of the server variables you can set, you may want to set initially the [server\\_audit\\_events](#) variable to tell the Audit Plugin which events to log. The [Log Settings documentation page](#) describes in detail the choices you have and provides examples of log entries related to them.

You can see a detailed list of system variables related to the MariaDB Audit Plugin on the [System Variables documentation page](#). Status variables related to the Audit Plugin are listed and explained on the [Status Variables documentation page](#).

### 4.4.4.3 MariaDB Audit Plugin - Log Settings

Events that are logged by the MariaDB Audit Plugin are grouped generally into different types: connect, query, and table events. To log based on these types of events, set the variable, [server\\_audit\\_events](#) to

CONNECT

,

QUERY

, or

TABLE

. To have the Audit Plugin log more than one type of event, put them in a comma-separated list like so:

```
SET GLOBAL server_audit_events = 'CONNECT,QUERY,TABLE';
```

## Contents

1. [Logging Connect Events](#)
2. [Logging Query Events](#)
  1. [Queries Not Included in Subordinate Query Event Types](#)
  3. [Logging Table Events](#)
  4. [Logging User Activities](#)
  5. [Excluding or Including Users](#)

You can put the equivalent of this in the configuration file like so:

```
[mysqld]
...
server_audit_events=connect,query
```

By default, logging is set to

OFF

. To enable it, set the [server\\_audit\\_logging](#) variable to

ON

. Note that if the [query cache](#) is enabled, and a query is returned from the query cache, no

TABLE

records will appear in the log since the server didn't open or access any tables and instead relied on the cached results. So you may want to disable query caching.

There are actually a few types of events that may be logged, not just the three common ones mentioned above. A full list of related system variables is detailed on the [Server\\_Audit System Variables](#) page, and status variables on the [Server\\_Audit Status Variables](#) page of this documentation. Some of the major ones are highlighted below:

Type	Description
CONNECT	Connects, disconnects and failed connects — including the error code
QUERY	Queries executed and their results in plain text, including failed queries due to syntax or permission errors
TABLE	Tables affected by query execution

---

Similar to

```
QUERY
, but filters only DDL-type queries (
CREATE
,
ALTER
,
DROP
,
RENAME
and
TRUNCATE
). There are some exceptions however.
RENAME USER
is not logged, while
CREATE/DROP [PROCEDURE / FUNCTION / USER]
are only logged from MariaDB 10.2.38 , MariaDB 10.3.29 , MariaDB 10.4.22 , MariaDB 10.5.13 and
MariaDB 10.6.5 . In earlier versions they are not logged. See MDEV-23457 .
```

---

Similar to

```
QUERY
, but filters only DML-type queries (
DO
,
CALL
,
LOAD DATA/XML
,
DELETE
,
INSERT
,
SELECT
,
UPDATE
,
HANDLER
and
REPLACE
statements)
```

---

Similar to

```
QUERY_DML
, but doesn't log SELECT queries. (since version 1.4.4) (
DO
,
CALL
,
LOAD DATA/XML
,
DELETE
,
INSERT
,
UPDATE
,
HANDLER
and
REPLACE
statements)
```

---

---

Similar to	<b>QUERY</b> , but filters only DCL-type queries ( <b>CREATE USER</b> , <b>DROP USER</b> , <b>RENAME USER</b> , <b>GRANT</b> , <b>REVOKE</b> and <b>SET PASSWORD</b> statements)
QUERY_DCL	

Since there are other types of queries besides DDL and DML, using the

```
QUERY_DDL
and
QUERY_DML
options together is not equivalent to using
QUERY
. Starting in version 1.3.0 of the Audit Plugin, there is the
QUERY_DCL
option for logging DCL types of queries (e.g.,
GRANT
and
REVOKE
```

statements). In the same version, the [server\\_audit\\_query\\_log\\_limit](#) variable was added to be able to set the length of a log record. Previously, a log entry would be truncated due to long query strings.

## Logging Connect Events

If the Audit Plugin has been configured to log connect events, it will log connects, disconnects, and failed connects. For a failed connection, the log includes the error code.

It's possible to define a list of users for which events can be excluded or included for tracing their database activities. This list will be ignored, though, for the loggings of connect events. This is because auditing standards distinguish between technical and physical users. Connects need to be logged for all types of users; access to objects need to be logged only for physical users.

## Logging Query Events

If

```
QUERY
,
QUERY_DDL
,
QUERY_DML
,
QUERY_DML_NO_SELECT
, and/or
QUERY_DCL
```

event types are enabled, then the corresponding types of queries that are executed will be logged for defined users. The queries will be logged exactly as they are executed, in plain text. This is a security vulnerability: anyone who has access to the log files will be able to read the queries. So make sure that only trusted users have access to the log files and that the files are in a protected location. An alternative is to use

```
TABLE
event type instead of the query-related event types.
```

Queries are also logged if they cannot be executed, if they're unsuccessful. For example, a query will be logged because of a syntax error or because the user doesn't have the privileges necessary to access an object. These queries can be parsed by the error code that's provided in the log.

You may find failed queries to be more interesting: They can reveal problems with applications (e.g., an SQL statement in an application that doesn't match the current schema). They can also reveal if a malicious user is guessing at the names of tables and columns to try to get access to data.

Below is an example in which a user attempts to execute an

```
UPDATE
statement on a table for which he does not have permission:
```

```
UPDATE employees
SET salary = salary * 1.2
WHERE emp_id = 18236;

ERROR 1142 (42000):
UPDATE command denied to user 'bob'@'localhost' for table 'employees'
```

Looking in the Audit Plugin log (  
server\_audit.log  
) for this entry, you can see the following entry:

```
20170817 11:07:18,ip-172-30-0-38,bob,localhost,15,46,QUERY,company,
'UPDATE employees SET salary = salary * 1.2 WHERE emp_id = 18236',1142
```

This log entry would be on one line, but it's reformatted here for better rendering. Looking at this log entry, you can see the date and time of the query, followed by the server host, the user and host for the account. Next is the connection and query identification numbers (i.e.,

15  
and  
46  
. After the log event type (i.e.,  
QUERY  
, the database name (i.e.,  
company  
, the query, and the error number is recorded.

Notice that the last value in the log entry is

1142  
. That's the error number for the query. To find failed queries, you would look for two elements: the notation indicating that it's a  
QUERY  
entry, and the last value for the entry. If the query is successful, the value will be  
0  
.

## Queries Not Included in Subordinate Query Event Types

Note that the

QUERY  
event type will log queries that are not included in any of the subordinate  
QUERY\_\*  
event types, such as:

- CREATE FUNCTION
- DROP FUNCTION
- CREATE PROCEDURE
- DROP PROCEDURE
- SET
- CHANGE MASTER TO
- FLUSH
- KILL
- CHECK
- OPTIMIZE
- LOCK
- UNLOCK
- ANALYZE
- INSTALL PLUGIN
- UNINSTALL PLUGIN
- INSTALL SONAME
- UNINSTALL SONAME
- EXPLAIN

## Logging Table Events

MariaDB has the ability to record table events in the logs — this is not a feature of MySQL. This feature is the only way to log which tables have been accessed through a view, a stored procedure, a stored function, or a trigger. Without this feature, a log entry for a query shows only the view, stored procedure or function used, not the underlying tables. Of course, you could create a custom application to parse each query executed to find the SQL statements used and the tables accessed, but that would be a drain on system resources. Table event logging is much simpler: it adds a line to the log for each table accessed, without any parsing. It includes notes as to whether it was a read or a write.

If you want to monitor user access to specific databases or tables (e.g.,

mysql.user  
, you can search the log for them. Then if you want to see a query which accessed a certain table, the audit log entry will include the query

identification number. You can use it to search the same log for the query entry. This can be useful when searching a log containing tens of thousands of entries.

Because of the

TABLE

option, you may disable query logging and still know who accessed which tables. You might want to disable

QUERY

event logging to prevent sensitive data from being logged. Since *table* event logging will log who accessed which table, you can still watch for malicious activities with the log. This is often enough to fulfill auditing requirements.

Below is an example with both

TABLE

and

QUERY

events logging. For this scenario, suppose there is a **VIEW** in which columns are selected from a few tables in a company

database. The underlying tables are related to sensitive employee information, in particular salaries. Although we may have taken precautions to ensure that only certain user accounts have access to those tables, we will monitor the Audit Plugin logs for anyone who queries them — directly or indirectly through a view.

```
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,READ,company,employees,
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,READ,company,employees_salaries,
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,READ,company,ref_job_titles,
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,READ,company,org_departments,
20170817 16:04:33,ip-172-30-0-38,root,localhost,29,913,QUERY,company,
'SELECT * FROM employee_pay WHERE title LIKE \'%Executive%' OR title LIKE '%Manager%',0
```

Although the user executed only one **SELECT** statement, there are multiple entries to the log: one for each table accessed and one entry for the query on the view, (i.e.,

employee\_pay

). We know primarily this is all for one query because they all have the same connection and query identification numbers (i.e.,

29

and

913

).

## Logging User Activities

The Audit Plugin will log the database activities of all users, or only the users that you specify. A database activity is defined as a *query* event or a *table* event. *Connect* events are logged for all users.

You may specify users to include in the log with the

server\_audit\_incl\_users

variable or exclude users with the

server\_audit\_excl\_users

variable. This can be useful if you would like to log entries, but are not interested in entries from trusted applications and would like to exclude them from the logs.

You would typically use either the

server\_audit\_incl\_users

variable or the

server\_audit\_excl\_users

variable. You may, though, use both variables. If a username is inadvertently listed in both variables, database activities for that user will be logged because

server\_audit\_incl\_users

takes priority.

Although MariaDB considers a user as the combination of the username and hostname, the Audit Plugin logs only based on the username. MariaDB uses both the username and hostname so as to grant privileges relevant to the location of the user. Privileges are not relevant though for tracing the access to database objects. The host name is still recorded in the log, but logging is not determined based on that information.

The following example shows how to add a new username to the

server\_audit\_incl\_users

variable without removing previous usernames:

```
SET GLOBAL server_audit_incl_users = CONCAT(@@global.server_audit_incl_users, ',Maria');
```

Remember to add also any new users to be included in the logs to the same variable in MariaDB configuration file. Otherwise, when the server restarts it will discard the setting.

## Excluding or Including Users

By default events from all users are logged, but certain users can be excluded from logging by using the `server_audit_excl_users` variable. For example, to exclude users `valerianus` and `rocky` from having their events logged:

```
server_audit_excl_users=valerianus,rocky
```

This option is primarily used to exclude the activities of trusted applications.

Alternatively, `server_audit_incl_users` can be used to specifically include users. Both variables can be used, but if a user appears on both lists, `server_audit_incl_users` has a higher priority, and their activities will be logged.

Note that

`CONNECT`

events are always logged for all users, regardless of these two settings. Logging is also based on username only, not the username and hostname combination that MariaDB uses to determine privileges.

## 4.4.4.4 MariaDB Audit Plugin - Location and Rotation of Logs

### Contents

1. Separate log files
2. System logs

Logs can be written to a separate file or to the system logs. If you prefer to have the logging separated from other system information, the value of the variable, `server_audit_output_type` should be set to

```
file  
. Incidentally,  
file  
is the only option on Windows systems.
```

You can force a rotation by enabling the `server_audit_file_rotate_now` variable like so:

```
SET GLOBAL server_audit_file_rotate_now = ON;
```

### Separate log files

In addition to setting `server_audit_output_type`, you will have to provide the file path and name of the audit file. This is set in the variable, `server_audit_file_path`. You can set the file size limit of the log file with the variable, `server_audit_file_rotate_size`.

So, if rotation is on and the log file has reached the size limit you set, a copy is created with a consecutive number as extension, the original file will be truncated to be used for the auditing again. To limit the number of log files created, set the variable, `server_audit_file_rotations`. You can force log file rotation by setting the variable, `server_audit_file_rotate_now` to a value of

`ON`

. When the number of files permitted is reached, the oldest file will be overwritten. Below is an example of how the variables described above might be set in a server's configuration files:

```
[mysqld]  
...  
server_audit_file_rotate_now=ON  
server_audit_file_rotate_size=1000000  
server_audit_file_rotations=5  
...
```

### System logs

For security reasons, it's better sometimes to use the system logs instead of a local file owned by the

```
mysql  
user. To do this, the value of server_audit_output_type needs to be set to  
syslog  
. Advanced configurations, such as using a remote  
syslogd  
service, are part of the  
syslogd  
configuration.
```

The variables, `server_audit_syslog_ident` and `server_audit_syslog_info` can be used to identify a system log entry made by the audit plugin. If a remote syslogd service is used for several MariaDB servers, these same variables are also used to identify the MariaDB server.

Below is an example of a system log entry taken from a server which had `server_audit_syslog_ident` set to the default value of

```
mysql -server_auditing  
, and server_audit_syslog_info set to
```

```
<prod1>
```

```
Aug 7 17:19:58localhostmysql- server_auditing:  
<prod1> localhost.localdomain,root,localhost,1,7,  
QUERY, mysql, 'SELECT * FROM user',0
```

Although the default values for `server_audit_syslog_facility` and `server_audit_syslog_priority` should be sufficient in most cases, they can be changed based on the definition in

```
syslog.h  
for the functions  
openlog()  
and  
syslog()
```

## 4.4.4.5 MariaDB Audit Plugin - Log Format

The audit plugin logs user access to MariaDB and its objects. The audit trail (i.e., audit log) is a set of records, written as a list of fields to a file in a plain-text format. The fields in the log are separated by commas. The format used for the plugin's own log file is slightly different from the format used if it logs to the system log because it has its own standard format. The general format for the logging to the plugin's own file is defined like the following:

```
[timestamp],[serverhost],[username],[host],[connectionid],  
[queryid],[operation],[database],[object],[retcode]
```

If the `server_audit_output_type` variable is set to

```
syslog  
instead of the default,  
file  
, the audit log file format will be as follows:
```

```
[timestamp][syslog_host][syslog_ident]:[syslog_info][serverhost],[username],[host],  
[connectionid],[queryid],[operation],[database],[object],[retcode]
```

Item logged	Description
timestamp	Time at which the event occurred. If syslog is used, the format is defined by syslogs
syslog_host	Host from which the syslog entry was received.
syslog_ident	For identifying a system log entry, including the MariaDB server.
syslog_info	For providing information for identifying a system log entry.
serverhost	The MariaDB server host name.
username	Connected user.
host	Host from which the user connected.
connectionid	Connection ID number for the related operation.
queryid	Query ID number, which can be used for finding the relational table events and related queries. For TABLE events, multiple lines will be added.
operation	Recorded action type: CONNECT, QUERY, READ, WRITE, CREATE, ALTER, RENAME, DROP.
database	Active database (as set by <code>USE</code> ).
object	Executed query for QUERY events, or the table name in the case of TABLE events.
retcode	Return code of the logged operation.

Various events will result in different audit records. Some events will not return a value for some fields (e.g., when the active database is not set when connecting to the server).

Below is a generic example of the output for connect events, with placeholders representing data. These are events in which a user connected, disconnected, or tried unsuccessfully to connect to the server.

```
[timestamp],[serverhost],[username],[host],[connectionid],0,CONNECT,[database],,0  
[timestamp],[serverhost],[username],[host],[connectionid],0,DISCONNECT,,,0  
[timestamp],[serverhost],[username],[host],[connectionid],0,FAILED_CONNECT,,,[retcode]
```

Here is the one audit record generated for each query event:

```
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],QUERY,[database],[object], [retcode]
```

Below are generic examples of records that are entered in the audit log for each type of table event:

```
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],CREATE,[database],[object],  
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],READ,[database],[object],  
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],WRITE,[database],[object],  
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],ALTER,[database],[object],  
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],RENAME,[database],  
[object_old]||[database_new].[object_new],  
[timestamp],[serverhost],[username],[host],[connectionid],[queryid],DROP,[database],[object],
```

Starting in version 1.2.0, passwords are hidden in the log for certain types of queries. They are replaced with asterisks for

```
GRANT  
,  
CREATE USER  
,  
CREATE MASTER  
,  
CREATE SERVER  
, and  
ALTER SERVER  
statements. Passwords, however, are not replaced for the  
PASSWORD()  
and  
OLD_PASSWORD()  
functions when they are used inside other SQL statements (i.e., SET PASSWORD  
).
```

#### 4.4.4.6 MariaDB Audit Plugin - Versions

Below is a list of the releases of the MariaDB Audit Plugin, the most recent version first, and in which versions of MariaDB each plugin version was included.

Version	Introduced
1.4.13	<a href="#">MariaDB 10.2.38</a> , <a href="#">MariaDB 10.3.29</a> , <a href="#">MariaDB 10.4.19</a> , <a href="#">MariaDB 10.5.10</a>
1.4.10	<a href="#">MariaDB 10.2.35</a> , <a href="#">MariaDB 10.3.26</a> , <a href="#">MariaDB 10.5.7</a>
1.4.7	<a href="#">MariaDB 10.1.41</a> , <a href="#">MariaDB 10.2.26</a> , <a href="#">MariaDB 10.3.17</a> , <a href="#">MariaDB 10.4.7</a>
1.4.5	<a href="#">MariaDB 10.2.24</a> , <a href="#">MariaDB 10.3.15</a> , <a href="#">MariaDB 10.4.5</a>
1.4.4	<a href="#">MariaDB 5.5.61</a> , <a href="#">MariaDB 10.0.36</a> , <a href="#">MariaDB 10.1.34</a> , <a href="#">MariaDB 10.2.15</a> , <a href="#">MariaDB 10.3.7</a> , <a href="#">MariaDB 10.4.0</a>
1.4.0	<a href="#">MariaDB 5.5.48</a> , <a href="#">MariaDB 10.0.24</a> , <a href="#">MariaDB 10.1.11</a>
1.3.0	<a href="#">MariaDB 5.5.43</a> , <a href="#">MariaDB 10.0.18</a> , <a href="#">MariaDB 10.1.5</a>
1.2.0	<a href="#">MariaDB 5.5.42</a> , <a href="#">MariaDB 10.0.17</a> , <a href="#">MariaDB 10.1.4</a>
1.1.7	<a href="#">MariaDB 5.5.38</a> , <a href="#">MariaDB 10.0.11</a> , <a href="#">MariaDB 10.1.0</a>
1.1.6	<a href="#">MariaDB 5.5.37</a> , <a href="#">MariaDB 10.0.10</a>
1.1.5	<a href="#">MariaDB 10.0.09</a>
1.1.4	<a href="#">MariaDB 5.5.36</a>
1.1.3	<a href="#">MariaDB 5.5.34</a> , <a href="#">MariaDB 10.0.7</a>

#### 4.4.4.7 MariaDB Audit Plugin Options and System Variables

## Contents

1. System Variables
  1. `server_audit_events`
  2. `server_audit_excl_users`
  3. `server_audit_file_path`
  4. `server_audit_file_rotate_now`
  5. `server_audit_file_rotate_size`
  6. `server_audit_file_rotations`
  7. `server_audit_incl_users`
  8. `server_audit_loc_info`
  9. `server_audit_logging`
  10. `server_audit_mode`
  11. `server_audit_output_type`
  12. `server_audit_query_log_limit`
  13. `server_audit_syslog_facility`
  14. `server_audit_syslog_ident`
  15. `server_audit_syslog_info`
  16. `server_audit_syslog_priority`
2. Options
  1. `server_audit`

There are several options and system variables related to the [MariaDB Audit Plugin](#), once it has been [installed](#). System variables can be displayed using the `SHOW VARIABLES` statement like so:

```
SHOW GLOBAL VARIABLES LIKE '%server_audit%';

+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| server_audit_events | CONNECT,QUERY,TABLE |
| server_audit_excl_users |          |
| server_audit_file_path | server_audit.log |
| server_audit_file_rotate_now | OFF |
| server_audit_file_rotate_size | 1000000 |
| server_audit_file_rotations | 9 |
| server_audit_incl_users |          |
| server_audit_logging | ON |
| server_audit_mode | 0 |
| server_audit_output_type | file |
| server_audit_query_log_limit | 1024 |
| server_audit_syslog_facility | LOG_USER |
| server_audit_syslog_ident | mysql-server_auditing |
| server_audit_syslog_info |          |
| server_audit_syslog_priority | LOG_INFO |
+-----+-----+
```

To change the value of one of these variables, you can use the

`SET`

statement, or set them at the command-line when starting MariaDB. It's recommended that you set them in the MariaDB configuration for the server like so:

```
[mariadb]
...
server_audit_excl_users='bob,ted'
...
```

## System Variables

Below is a list of all system variables related to the Audit Plugin. See [Server System Variables](#) for a complete list of system variables and instructions on setting them. See also the [full list of MariaDB options, system and status variables](#).

`server_audit_events`

- **Description:** If set, then this restricts audit logging to certain event types. If not set, then every event type is logged to the audit log. For example:  
`SET GLOBAL server_audit_events='connect, query'`
- **Commandline:**  
`--server-audit-events=value`
- **Scope:** Global
- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:** Empty string

- **Valid Values:**

- - CONNECT
  - ,
  - QUERY
  - ,
  - TABLE

(MariaDB Audit Plugin < 1.2.0)
- - CONNECT
  - ,
  - QUERY
  - ,
  - TABLE
  - ,
  - QUERY\_DDL
  - ,
  - QUERY\_DML

(MariaDB Audit Plugin >= 1.2.0)
- - CONNECT
  - ,
  - QUERY
  - ,
  - TABLE
  - ,
  - QUERY\_DDL
  - ,
  - QUERY\_DML
  - ,
  - QUERY\_DCL

(MariaDB Audit Plugin >=1.3.0)
- - CONNECT
  - ,
  - QUERY
  - ,
  - TABLE
  - ,
  - QUERY\_DDL
  - ,
  - QUERY\_DML
  - ,
  - QUERY\_DCL
  - ,
  - QUERY\_DML\_NO\_SELECT

(MariaDB Audit Plugin >= 1.4.4)

◦ See [MariaDB Audit Plugin - Versions](#) to determine which MariaDB releases contain each MariaDB Audit Plugin versions.

---

### server\_audit\_excl\_users

- **Description:** If not empty, it contains the list of users whose activity will NOT be logged. For example:

```
SET GLOBAL server_audit_excl_users='user_foo, user_bar'
```

. CONNECT records aren't affected by this variable - they are always logged. The user is still logged if it's specified in [server\\_audit\\_incl\\_users](#).

- **Commandline:**

```
--server-audit-excl-users=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:** Empty string

- **Size limit:** 1024 characters

---

## server\_audit\_file\_path

- **Description:** When `server_audit_output_type=file`, sets the path and the filename to the log file. If the specified path exists as a directory, then the log will be created inside that directory with the name 'server\_audit.log'. Otherwise the value is treated as a filename. The default value is 'server\_audit.log', which means this file will be created in the database directory.

- **Commandline:**

```
--server-audit-file-path=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:**

server\_audit.log

---

## server\_audit\_file\_rotate\_now

- **Description:** When `server_audit_output_type=file`, the user can force the log file rotation by setting this variable to ON or 1.

- **Commandline:**

```
--server-audit-rotate-now[={0|1}]
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

OFF

---

## server\_audit\_file\_rotate\_size

- **Description:** When `server_audit_output_type=file`, it limits the size of the log file to the given amount of bytes. Reaching that limit turns on the rotation - the current log file is renamed as 'file\_path.1'. The empty log file is created as 'file\_path' to log into it. The default value is 1000000.

- **Commandline:**

```
--server-audit-rotate-size=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1000000

- **Range:**

100  
to  
9223372036854775807

---

## server\_audit\_file\_rotations

- **Description:** When `server_audit_output_type=file`', this specifies the number of rotations to save. If set to 0 then the log never rotates. The default value is 9.

- **Commandline:**

```
--server-audit-rotations=#
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric

- **Default Value:**

9

- **Range:**

0  
to  
999

---

### server\_audit\_incl\_users

- **Description:** If not empty, it contains a comma-delimited list of users whose activity will be logged. For example:

```
SET GLOBAL server_audit_incl_users='user_foo, user_bar'  
. CONNECT records aren't affected by this variable - they are always logged. This setting has higher priority than  
server_audit_excl_users . So if the same user is specified both in incl_ and excl_ lists, they will still be logged.
```

- **Commandline:**

--server-audit-incl-users=value

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
string

- **Default Value:** Empty string
- **Size limit:** 1024 characters

---

### server\_audit\_loc\_info

- **Description:** Used by plugin internals. It has no useful meaning to users.

- In earlier versions, users see it as a read-only variable.
- In later versions, it is hidden from the user.

- **Commandline:** N/A
- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
string

- **Default Value:** Empty string
- **Introduced:** MariaDB 10.1.12 , MariaDB 10.0.24 , MariaDB 5.5.48
- **Hidden:** MariaDB 10.1.18 , MariaDB 10.0.28 , MariaDB 5.5.53

---

### server\_audit\_logging

- **Description:** Enables/disables the logging. Expected values are ON/OFF. For example:

```
SET GLOBAL server_audit_logging=on
```

If the server\_audit\_output\_type is FILE, this will actually create/open the logfile so the server\_audit\_file\_path should be properly specified beforehand. Same about the SYSLOG-related parameters. The logging is turned off by default.

- **Commandline:**

--server-audit-logging[={0|1}]

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**

---

## server\_audit\_mode

- **Description:** This variable doesn't have any distinctive meaning for a user. Its value mostly reflects the server version with which the plugin was started and is intended to be used by developers for testing.

- **Commandline:**

```
--server-audit-mode[=#]
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
numeric
```

- **Default Value:**

```
0
```

- **Range:**

```
0
```

```
to
```

```
1
```

---

## server\_audit\_output\_type

- **Description:** Specifies the desired output type. Can be SYSLOG, FILE or null as no output. For example:

```
SET GLOBAL server_audit_output_type=file
```

file: log records will be saved into the rotating log file. The name of the file set by [server\\_audit\\_file\\_path](#) variable. syslog: log records will be sent to the local syslogd daemon with the standard <syslog.h> API. The default value is 'file'.

- **Commandline:**

```
--server-audit-output-type=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
enum
```

- **Default Value:**

```
file
```

- **Valid Values:**

```
SYSLOG
```

```
,
```

```
FILE
```

---

## server\_audit\_query\_log\_limit

- **Description:** Limit on the length of the query string in a record.

- **Commandline:**

```
--server-audit-query-log-limit=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

```
numeric
```

- **Default Value:**

```
1024
```

- **Range:**

0  
to  
2147483647

---

### server\_audit\_syslog\_facility

- **Description:** SYSLOG-mode variable. It defines the 'facility' of the records that will be sent to the syslog. Later the log can be filtered by this parameter.

- **Commandline:**

```
--server-audit-syslog-facility=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

enum

- **Default Value:**

LOG\_USER

- **Valid Values:**

```
LOG_USER
,
LOG_MAIL
,
LOG_DAEMON
,
LOG_AUTH
,
LOG_SYSLOG
,
LOG_LPR
,
LOG_NEWS
,
LOG_UUCP
,
LOG_CRON
,
LOG_AUTHPRIV
,
LOG_FTP
, and
LOG_LOCAL0
-
LOG_LOCAL7
```

---

### server\_audit\_syslog\_ident

- **Description:** SYSLOG-mode variable. String value for the 'ident' part of each syslog record. Default value is 'mysql-server\_auditing'. New value becomes effective only after restarting the logging.

- **Commandline:**

```
--server-audit-syslog-ident=value
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

string

- **Default Value:**

mysql-server\_auditing

---

## server\_audit\_syslog\_info

- **Description:** SYSLOG-mode variable. The 'info' string to be added to the syslog records. Can be changed any time.
  - **Commandline:**  
--server-audit-syslog-info=value
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
string
  - **Default Value:** Empty string
- 

## server\_audit\_syslog\_priority

- **Description:** SYSLOG-mode variable. Defines the priority of the log records for the syslogd.
  - **Commandline:**  
--server-audit-syslog-priority=value
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
enum
  - **Default Value:**  
LOG\_INFO
  - **Valid Values:**  
LOG\_EMERG  
,  
LOG\_ALERT  
,  
LOG\_CRIT  
,  
LOG\_ERR  
,  
LOG\_WARNING  
,  
LOG\_NOTICE  
,  
LOG\_INFO  
,  
LOG\_DEBUG
- 

## Options

### server\_audit

- **Description:** Controls how the server should treat the plugin when the server starts up.
    - Valid values are:
      - OFF
        - Disables the plugin without removing it from the `mysql.plugins` table.
      - ON
        - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
-

FORCE

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

FORCE\_PLUS\_PERMANENT

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

UNINSTALL SONAME

or

UNINSTALL PLUGIN

while the server is running.

- See [MariaDB Audit Plugin - Installation: Prohibiting Uninstallation](#) for more information on one use case.
- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

--server-audit=val

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF

,

ON

,

FORCE

,

FORCE\_PLUS\_PERMANENT

## 4.4.4.8 MariaDB Audit Plugin - Status Variables

### Contents

1. Status Variables
  1. [Server\\_audit\\_active](#)
  2. [Server\\_audit\\_current\\_log](#)
  3. [Server\\_audit\\_last\\_error](#)
  4. [Server\\_audit\\_writes\\_failed](#)

There are a few status variables related to the [MariaDB Audit Plugin](#), once it has been [installed](#). These variables can be displayed using the [SHOW STATUS](#) statement like so:

```
SHOW STATUS LIKE 'server_audit%';

+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| Server_audit_active | ON
| Server_audit_current_log | server_audit.log
| Server_audit_last_error | 
| Server_audit_writes_failed | 0
+-----+-----+
```

### Status Variables

Below is a list of all status variables related to the Audit Plugin. These cannot be set: These are not to be confused with system variables, which can be set. See [Server Status Variables](#) for a complete list of status variables that can be viewed with the [SHOW STATUS](#) statement. See also the [Full list of MariaDB options, system and status variables](#).

#### Server\_audit\_active

- **Description:** If the auditing is actually working. It gets the ON value when the logging is successfully started. Then it can get the OFF value if the logging was stopped or log records can't be properly stored due to file or syslog errors.

- **Data Type:**

boolean

---

#### Server\_audit\_current\_log

- **Description:** The name of the logfile or the SYSLOG parameters that are in current use.

- **Data Type:**

string

---

#### Server\_audit\_last\_error

- **Description:** If something went wrong with the logging here you can see the message.

- **Data Type:**

string

---

#### Server\_audit\_writes\_failed

- **Description:** The number of log records since last logging-start that weren't properly stored because of errors of any kind. The global value can be flushed by

[FLUSH STATUS](#)

- **Data Type:**

numeric

- **Default Value:**

0

---

## 4.4.5 Authentication Plugins

When a user attempts to log in, the authentication plugin controls how MariaDB Server determines whether the connection is from a legitimate user.

When creating or altering a user account with the [GRANT](#), [CREATE USER](#) or [ALTER USER](#) statements, you can specify the authentication plugin you want the user account to use by providing the `IDENTIFIED VIA` clause. By default, when you create a user account without specifying an authentication plugin, MariaDB uses the [mysql\\_native\\_password](#) plugin.

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, there are some notable changes, such as:

- You can specify multiple authentication plugins for each user account.
- The `root@localhost` user created by [mysql\\_install\\_db](#) is created with the ability to use two authentication plugins. First, it is configured to try to use the `unix_socket` authentication plugin. This allows the `root@localhost` user to login without a password via the local Unix socket file defined by the `socket` system variable, as long as the login is attempted from a process owned by the operating system `root` user account. Second, if authentication fails with the `unix_socket` authentication plugin, then it is configured to try to use the [mysql\\_native\\_password](#) authentication plugin. However, an invalid password is initially set, so in order to authenticate this way, a password must be set with [SET PASSWORD](#).



### Pluggable Authentication Overview

*The authentication of users is delegated to plugins.*



### Authentication Plugin - mysql\_native\_password

*Uses the password hashing algorithm introduced in MySQL 4.1.*



### Authentication Plugin - mysql\_old\_password

*The mysql\_old\_password authentication plugin uses the pre-MySQL 4.1 password hashing algorithm.*



### Authentication Plugin - ed25519

Uses the Elliptic Curve Digital Signature Algorithm to securely store users' passwords.



### Authentication Plugin - GSSAPI

The gssapi authentication plugin uses the GSSAPI interface to authenticate with Kerberos or NTLM.



### Authentication with Pluggable Authentication Modules (PAM)

Uses the Pluggable Authentication Module (PAM) framework to authenticate MariaDB users.



### Authentication Plugin - Unix Socket

Uses the user name that owns the process connected to MariaDB's unix socket file.



### Authentication Plugin - Named Pipe

Uses the user name that owns the process connected to MariaDB's named pipe on Windows.



### Authentication Plugin - SHA-256

MySQL supports the sha256\_password and caching\_sha2\_password authentication plugins.

There are 1 related questions .

## 4.4.5.1 Pluggable Authentication Overview

### Contents

- 1. Supported Authentication Plugins
  - 1. Supported Server Authentication Plugins
  - 2. Supported Client Authentication Plugins
- 2. Options Related to Authentication Plugins
  - 1. Server Options Related to Authentication Plugins
  - 2. Client Options Related to Authentication Plugins
  - 3. Installation Options Related to Authentication Plugins
- 3. Extended SQL Syntax
- 4. Authentication Plugins Installed by Default
  - 1. Server Authentication Plugins Installed by Default
  - 2. Client Authentication Plugins Installed by Default
- 5. Default Authentication Plugin
  - 1. Default Server Authentication Plugin
  - 2. Default Client Authentication Plugin
    - 1. Setting the Default Client Authentication Plugin
- 6. Authentication Plugins
  - 1. Server Authentication Plugins
    - 1. mysql\_native\_password
    - 2. mysql\_old\_password
    - 3. ed25519
    - 4. gssapi
    - 5. pam
    - 6. unix\_socket
    - 7. named\_pipe
- 7. Authentication Plugin API
  - 1. Dialog Client Authentication Plugin - Client Library Extension
- 8. See Also

When a user attempts to log in, the authentication plugin controls how MariaDB Server determines whether the connection is from a legitimate user.

When creating or altering a user account with the GRANT , CREATE USER or ALTER USER statements, you can specify the authentication plugin you want the user account to use by providing the

IDENTIFIED VIA

clause. By default, when you create a user account without specifying an authentication plugin, MariaDB uses the mysql\_native\_password plugin.

In MariaDB 10.4 and later, there are some notable changes, such as:

- You can specify multiple authentication plugins for each user account.

- The

```
root@localhost
```

user created by `mysql_install_db` is created with the ability to use two authentication plugins. First, it is configured to try to use the `unix_socket` authentication plugin. This allows the the

```
root@localhost
```

user to login without a password via the local Unix socket file defined by the `socket` system variable, as long as the login is attempted from a process owned by the operating system

```
root
```

user account. Second, if authentication fails with the `unix_socket` authentication plugin, then it is configured to try to use the `mysql_native_password` authentication plugin. However, an invalid password is initially set, so in order to authenticate this way, a password must be set with `SET PASSWORD`.

## Supported Authentication Plugins

The authentication process is a conversation between the server and a client. MariaDB implements both server-side and client-side authentication plugins.

### Supported Server Authentication Plugins

MariaDB provides seven server-side authentication plugins:

- `mysql_native_password`
- `mysql_old_password`
- `ed25519`
- `gssapi`
- `pam` (Unix only)
- `unix_socket` (Unix only)
- `named_pipe` (Windows only)

### Supported Client Authentication Plugins

MariaDB provides eight client-side authentication plugins:

- `mysql_native_password`
- `mysql_old_password`
- `client_ed25519`
- `auth_gssapi_client`
- `dialog`
- `mysql_clear_password`
- `sha256_password`
- `caching_sha256_password`

## Options Related to Authentication Plugins

### Server Options Related to Authentication Plugins

MariaDB supports the following server options related to authentication plugins:

Server Option	Description
<code>old_passwords={1   0}</code>	If set to 1 ( 0 is default), MariaDB reverts to using the <code>mysql_old_password</code> authentication plugin by default for newly created users and passwords, instead of the <code>mysql_native_password</code> authentication plugin.
<code>plugin_dir=path</code>	Path to the <code>plugin</code> directory. For security reasons, either make sure this directory can only be read by the server, or set <code>secure_file_priv</code> .

<code>plugin_maturity=level</code>	The lowest acceptable <a href="#">plugin</a> maturity. MariaDB will not load plugins less mature than the specified level.
<code>secure_auth</code>	Connections will be blocked if they use the the <a href="#">mysql_old_password</a> authentication plugin. The server will also fail to start if the privilege tables are in the old, pre-MySQL 4.1 format.

## Client Options Related to Authentication Plugins

Most [clients and utilities](#) support some command line arguments related to client authentication plugins:

Client Option	Description
<code>--connect-expired-password</code>	Notify the server that this client is prepared to handle <a href="#">expired password sandbox mode</a> even if <code>--batch</code> was specified. From <a href="#">MariaDB 10.4.3</a> .
<code>--default-auth=name</code>	Default authentication client-side plugin to use.
<code>--plugin-dir=path</code>	Directory for client-side plugins.
<code>--secure-auth</code>	Refuse to connect to the server if the server uses the <a href="#">mysql_old_password</a> authentication plugin. This mode is off by default, which is a difference in behavior compared to MySQL 5.6 and later, where it is on by default.

Developers who are using [MariaDB Connector/C](#) can implement similar functionality in their application by setting the following options with the `mysql_optionsv` function:

- `MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS`
- `MYSQL_PLUGIN_DIR`
- `MYSQL_DEFAULT_AUTH`
- `MYSQL_SECURE_AUTH`

For example:

```
mysql_optionsv(mysql, MYSQL_OPT_CAN_HANDLE_EXPIRED_PASSWORDS, 1);
mysql_optionsv(mysql, MYSQL_DEFAULT_AUTH, "name");
mysql_optionsv(mysql, MYSQL_PLUGIN_DIR, "path");
mysql_optionsv(mysql, MYSQL_SECURE_AUTH, 1);
```

## Installation Options Related to Authentication Plugins

[mysql\\_install\\_db](#) supports the following installation options related to authentication plugins:

Installation Option	Description
---------------------	-------------

--auth-root- authentication-method={normal   socket}  user=USER --auth-root-socket-	If set to normal , it creates a root@localhost account that authenticates with the <a href="#">mysql_native_password</a> authentication plugin and that has no initial password set, which can be insecure. If set to socket , it creates a root@localhost account that authenticates with the <a href="#">unix_socket</a> authentication plugin. Set to normal by default. Available since <a href="#">MariaDB 10.1</a> .  Used with --auth-root-authentication-method=socket . It specifies the name of the second account to create with <a href="#">SUPER</a> privileges in addition to root , as well as of the system account allowed to access it. Defaults to the value of --user .
-------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Extended SQL Syntax

MariaDB has extended the SQL standard [GRANT](#), [CREATE USER](#), and [ALTER USER](#) statements, so that they support specifying different authentication plugins for specific users. An authentication plugin can be specified with these statements by providing the

IDENTIFIED VIA  
clause.

For example, the [GRANT](#) syntax is:

```
GRANT <privileges> ON <level> TO <user>  
IDENTIFIED VIA <plugin> [ USING <string> ]
```

And the [CREATE USER](#) syntax is:

```
CREATE USER <user>  
IDENTIFIED VIA <plugin> [ USING <string> ]
```

And the [ALTER USER](#) syntax is:

```
ALTER USER <user>  
IDENTIFIED VIA <plugin> [ USING <string> ]
```

The optional

USING  
clause allows users to provide an authentication string to a plugin. The authentication string's format and meaning is completely defined by the plugin.

For example, for the [mysql\\_native\\_password](#) authentication plugin, the authentication string should be a password hash:

```
CREATE USER mysqltest_up1  
IDENTIFIED VIA mysql_native_password USING '*E8D46CE25265E545D225A8A6F1BAF642FEBEE5CB';
```

Since [mysql\\_native\\_password](#) is the default authentication plugin, the above is just another way of saying the following:

```
CREATE USER mysqltest_up1  
IDENTIFIED BY PASSWORD '*E8D46CE25265E545D225A8A6F1BAF642FEBEE5CB';
```

In contrast, for the [pam](#) authentication plugin, the authentication string should refer to a [PAM service name](#):

```
CREATE USER mysqltest_up1  
IDENTIFIED VIA pam USING 'mariadb';
```

MariaDB starting with 10.4

In [MariaDB 10.4](#) and later, a user account can be associated with multiple authentication plugins.

For example, to configure the

root@localhost  
user account to try the [unix\\_socket](#) authentication plugin, followed by the [mysql\\_native\\_password](#) authentication plugin as a backup, you

could execute the following:

```
CREATE USER root@localhost
IDENTIFIED VIA unix_socket
OR mysql_native_password USING PASSWORD("verysecret");
```

See [Authentication from MariaDB 10.4](#) for more information.

## Authentication Plugins Installed by Default

### Server Authentication Plugins Installed by Default

Not all server-side authentication plugins are installed by default. If a specific server-side authentication plugin is not installed by default, then you can find the installation procedure on the documentation page for the specific authentication plugin.

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, the following server-side authentication plugins are installed by default:

- The `mysql_native_password` and `mysql_old_password` authentication plugins are installed by default in all builds.
- The `unix_socket` authentication plugin is installed by default in all builds on Unix and Linux.
- The `named_pipe` authentication plugin is installed by default in all builds on Windows.

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and below, the following server-side authentication plugins are installed by default:

- The `mysql_native_password` and `mysql_old_password` authentication plugins are installed by default in all builds.
- The `unix_socket` authentication plugin is installed by default in **new installations** that use the `.deb` packages provided by Debian's default repositories in Debian 9 and later and Ubuntu's default repositories in Ubuntu 15.10 and later. See [Differences in MariaDB in Debian \(and Ubuntu\)](#) for more information.
- The `named_pipe` authentication plugin is installed by default in all builds on Windows.

## Client Authentication Plugins Installed by Default

Client-side authentication plugins do not need to be *installed* in the same way that server-side authentication plugins do. If the client uses either the `libmysqlclient` or [MariaDB Connector/C](#) library, then the library automatically loads client-side authentication plugins from the library's plugin directory whenever they are needed.

Most [clients and utilities](#) support the

```
--plugin-dir
```

command line argument that can be used to set the path to the library's plugin directory:

Client Option	Description
<code>--plugin-dir=path</code>	Directory for client-side plugins.

Developers who are using [MariaDB Connector/C](#) can implement similar functionality in their application by setting the `MYSQL_PLUGIN_DIR` option with the `mysql_optionsv` function.

For example:

```
mysql_optionsv(mysql, MYSQL_PLUGIN_DIR, "path");
```

If your client encounters errors similar to the following, then you may need to set the path to the library's plugin directory:

```
ERROR 2059 (HY000): Authentication plugin 'dialog' cannot be loaded: /usr/lib/mysql/plugin/dialog.so: cannot open shared object
```

If the client does **not** use either the

`libmysqlclient` or [MariaDB Connector/C](#) library, then you will have to determine which authentication plugins are supported by the specific client library used by the client.

If the client uses either the `libmysqlclient` or [MariaDB Connector/C](#) library, but the client is not bundled with either library's *optional* client authentication plugins, then you can only use the conventional authentication plugins (like `mysql_native_password` and `mysql_old_password`) and the non-conventional authentication plugins that don't require special client-side authentication plugins (like `unix_socket` and `named_pipe`).

## Default Authentication Plugin

### Default Server Authentication Plugin

The `mysql_native_password` authentication plugin is currently the default authentication plugin in all versions of MariaDB if the `old_passwords` system variable is set to

`0`, which is the default.

On a system with the `old_passwords` system variable set to

`0`, this means that if you create a user account with either the `GRANT` or `CREATE USER` statements, and if you do not specify an authentication plugin with the `IDENTIFIED VIA` clause, then MariaDB will use the

`mysql_native_password`

authentication plugin for the user account.

For example, this user account will use the `mysql_native_password` authentication plugin:

```
CREATE USER username@hostname;
```

And so will this user account:

```
CREATE USER username@hostname IDENTIFIED BY 'notagoodpassword';
```

The `mysql_old_password` authentication plugin becomes the default authentication plugin in all versions of MariaDB if the `old_passwords` system variable is explicitly set to

`1`.

However, the `mysql_old_password` authentication plugin is not considered secure, so it is recommended to avoid using this authentication plugin. To help prevent undesired use of the `mysql_old_password` authentication plugin, the server supports the `secure_auth` system variable that can be used to configured the server to refuse connections that try to use the `mysql_old_password` authentication plugin:

Server Option	Description
<code>old_passwords={1   0}</code>	If set to <code>1</code> <code>(</code> <code>0</code> is default), MariaDB reverts to using the <code>mysql_old_password</code> authentication plugin by default for newly created users and passwords, instead of the <code>mysql_native_password</code> authentication plugin.
<code>secure_auth</code>	Connections will be blocked if they use the the <code>mysql_old_password</code> authentication plugin. The server will also fail to start if the privilege tables are in the old, pre-MySQL 4.1 format.

Most [clients and utilities](#) also support the

`--secure-auth`

command line argument that can also be used to configure the client to refuse to connect to servers that use the `mysql_old_password` authentication plugin:

Client Option	Description
<code>--secure-auth</code>	Refuse to connect to the server if the server uses the <code>mysql_old_password</code> authentication plugin. This mode is off by default, which is a difference in behavior compared to MySQL 5.6 and later, where it is on by default.

Developers who are using [MariaDB Connector/C](#) can implement similar functionality in their application by setting the `MYSQL_SECURE_AUTH` option with the [mysql\\_optionsv](#) function.

For example:

```
mysql_optionsv(mysql, MYSQL_SECURE_AUTH, 1);
```

## Default Client Authentication Plugin

The default client-side authentication plugin depends on a few factors.

If a client doesn't explicitly set the default client-side authentication plugin, then the client will determine which authentication plugin to use by checking the length of the scramble in the server's handshake packet.

If the server's handshake packet contains a 9-byte scramble, then the client will default to the [mysql\\_old\\_password](#) authentication plugin.

If the server's handshake packet contains a 20-byte scramble, then the client will default to the [mysql\\_native\\_password](#) authentication plugin.

### Setting the Default Client Authentication Plugin

Most [clients and utilities](#) support the

```
--default-auth  
command line argument that can be used to set the default client-side authentication plugin:
```

Client Option	Description
<code>--default-auth=name</code>	Default authentication client-side plugin to use.

Developers who are using [MariaDB Connector/C](#) can implement similar functionality in their application by setting the `MYSQL_DEFAULT_AUTH` option with the [mysql\\_optionsv](#) function.

For example:

```
mysql_optionsv(mysql, MYSQL_DEFAULT_AUTH, "name");
```

If you know that your user account is configured to require a client-side authentication plugin that isn't [mysql\\_old\\_password](#) or [mysql\\_native\\_password](#), then it can help speed up your connection process to explicitly set the default client-side authentication plugin.

According to the [client-server protocol](#), the server first sends the handshake packet to the client, then the client replies with a packet containing the user name of the user account that is requesting access. The server handshake packet initially tells the client to use the default server authentication plugin, and the client reply initially tells the server that it will use the default client authentication plugin.

However, the server-side and client-side authentication plugins mentioned in these initial packets may not be the correct ones for this specific user account. The server only knows what authentication plugin to use for this specific user account after reading the user name from the client reply packet and finding the appropriate row for the user account in either the [mysql.user](#) table or the [mysql.global\\_priv](#) table, depending on the MariaDB version.

If the server finds that either the server-side or client-side default authentication plugin does not match the actual authentication plugin that should be used for the given user account, then the server restarts the authentication on either the server side or the client side.

This means that, if you know what client authentication plugin your user account requires, then you can avoid an unnecessary authentication restart and you can save two packets and two round-trips between the client and server by configuring your client to use the correct authentication plugin by default.

## Authentication Plugins

### Server Authentication Plugins

#### `mysql_native_password`

The [mysql\\_native\\_password](#) authentication plugin uses the password hashing algorithm introduced in MySQL 4.1, which is also used by the [PASSWORD\(\)](#) function when

```
old_passwords=0
```

is set. This hashing algorithm is based on

```
SHA-1
```

## `mysql_old_password`

The `mysql_old_password` authentication plugin uses the pre-MySQL 4.1 password hashing algorithm, which is also used by the `OLD_PASSWORD()` function and by the `PASSWORD()` function when `old_passwords=1` is set.

## `ed25519`

The `ed25519` authentication plugin uses [Elliptic Curve Digital Signature Algorithm](#) to securely store users' passwords and to authenticate users. The `ed25519` algorithm is the same one that is [used by OpenSSH](#). It is based on the elliptic curve and code created by [Daniel J. Bernstein](#).

From a user's perspective, the `ed25519` authentication plugin still provides conventional password-based authentication.

## `gssapi`

The `gssapi` authentication plugin allows the user to authenticate with services that use the [Generic Security Services Application Program Interface \(GSSAPI\)](#). Windows has a slightly different but very similar API called [Security Support Provider Interface \(SSPI\)](#).

On Windows, this authentication plugin supports [Kerberos](#) and [NTLM](#) authentication. Windows authentication is supported regardless of whether a [domain](#) is used in the environment.

On Unix systems, the most dominant GSSAPI service is [Kerberos](#). However, it is less commonly used on Unix systems than it is on Windows. Regardless, this authentication plugin also supports Kerberos authentication on Unix.

The `gssapi` authentication plugin is most often used for authenticating with [Microsoft Active Directory](#).

## `pam`

The `pam` authentication plugin allows MariaDB to offload user authentication to the system's [Pluggable Authentication Module \(PAM\)](#) framework. PAM is an authentication framework used by Linux, FreeBSD, Solaris, and other Unix-like operating systems.

## `unix_socket`

The `unix_socket` authentication plugin allows the user to use operating system credentials when connecting to MariaDB via the local Unix socket file. This Unix socket file is defined by the `socket` system variable.

The `unix_socket` authentication plugin works by calling the `getsockopt` system call with the `SO_PEERCRED` socket option, which allows it to retrieve the `uid` of the process that is connected to the socket. It is then able to get the user name associated with that `uid`. Once it has the user name, it will authenticate the connecting user as the MariaDB account that has the same user name.

For example:

```
$ mysql -uroot
MariaDB []> CREATE USER serg IDENTIFIED VIA unix_socket;
MariaDB []> CREATE USER monty IDENTIFIED VIA unix_socket;
MariaDB []> quit
Bye
$ whoami
serg
$ mysql --user=serg
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.2.0-MariaDB-alpha-debug Source distribution
MariaDB []> quit
Bye
$ mysql --user=monty
ERROR 1045 (28000): Access denied for user 'monty'@'localhost' (using password: NO)
```

In this example, a user

serg

is already logged into the operating system and has full shell access. He has already authenticated with the operating system and his MariaDB account is configured to use the [unix\\_socket](#) authentication plugin, so he does not need to authenticate again for the database. MariaDB accepts his operating system credentials and allows him to connect. However, any attempt to connect to the database as another operating system user will be denied.

### named\_pipe

The [named\\_pipe](#) authentication plugin allows the user to use operating system credentials when connecting to MariaDB via named pipe on Windows. Named pipe connections are enabled by the [named\\_pipe](#) system variable.

The [named\\_pipe](#) authentication plugin works by using [named pipe impersonation](#) and calling

GetUserName()

to retrieve the user name of the process that is connected to the named pipe. Once it has the user name, it authenticates the connecting user as the MariaDB account that has the same user name.

For example:

```
CREATE USER wlad IDENTIFIED VIA named_pipe;
CREATE USER monty IDENTIFIED VIA named_pipe;
quit

C:\>echo %USERNAME%
wlad

C:\> mysql --user=wlad --protocol=PIPE
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.1.12-MariaDB-debug Source distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> quit
Bye

C:\> mysql --user=monty --protocol=PIPE
ERROR 1698 (28000): Access denied for user 'monty'@'localhost'
```

## Authentication Plugin API

The authentication plugin API is extensively documented in the [source code](#) in the following files:

- [mysql/plugin\\_auth.h](#)  
(server part)
- [mysql/client\\_plugin.h](#)  
(client part)
- [mysql/plugin\\_auth\\_common.h](#)  
(common parts)

The MariaDB [source code](#) also contains some authentication plugins that are intended explicitly to be examples for developers. They are located in [plugin/auth\\_examples](#)

The definitions of two example authentication plugins called

two\_questions  
and  
three\_attempts  
can be seen in  
[plugin/auth\\_examples/dialog\\_examples.c](#).  
. These authentication plugins demonstrate how to communicate with the user using the [dialog](#) client authentication plugin.

The

two\_questions  
authentication plugin asks the user for a password and a confirmation ("Are you sure?").

The

three\_attempts  
authentication plugin gives the user three attempts to enter a correct password.

The password for both of these plugins should be specified in the plain text in the

USING  
clause:

```
CREATE USER insecure IDENTIFIED VIA two_questions USING 'notverysecret';
```

## Dialog Client Authentication Plugin - Client Library Extension

The [dialog](#) client authentication plugin, strictly speaking, is not part of the client-server or authentication plugin API. But it can be loaded into any client application that uses the

`libmysqlclient`  
or [MariaDB Connector/C](#) libraries. This authentication plugin provides a way for the application to customize the UI of the dialog function.

In order to use the [dialog](#) client authentication plugin to communicate with the user in a customized way, the application will need to implement a function with the following signature:

```
extern "C" char *mysql_authentication_dialog_ask(  
    MYSQL *mysql, int type, const char *prompt, char *buf, int buf_len)
```

The function takes the following arguments:

- The connection handle.
- A question "type", which has one of the following values:
  - 1
    - Normal question
  - 2
    - Password (no echo)
- A prompt.
- A buffer.
- The length of the buffer.

The function returns a pointer to a string of characters, as entered by the user. It may be stored in

`buf`  
or allocated with  
`malloc()`

Using this function a GUI application can pop up a dialog window, a network application can send the question over the network, as required. If no

`mysql_authentication_dialog_ask`  
function is provided by the application, the [dialog](#) client authentication plugin falls back to `fputs()` and `fgets()`.

Providing this callback is particularly important on Windows, because Windows GUI applications have no associated console and the default dialog function will not be able to reach the user. An example of Windows GUI client that does it correctly is [HeidiSQL](#).

## See Also

- [GRANT](#)
- [CREATE USER](#)
- [ALTER USER](#)
- [Authentication from MariaDB 10.4](#)
- [Who are you? The history of MySQL and MariaDB authentication protocols from 1997 to 2017](#)
- [MySQL 5.6 Reference Manual: Pluggable Authentication](#)
- [MySQL 5.6 Reference Manual: Writing Authentication Plugins](#)

## 4.4.5.2 Authentication Plugin - mysql\_native\_password

The

`mysql_native_password`

authentication plugin is the default authentication plugin that will be used for an account created when no authentication plugin is explicitly mentioned and `old_passwords=0` is set. It uses the password hashing algorithm introduced in MySQL 4.1, which is also used by the `PASSWORD()` function when `old_passwords=0` is set. This hashing algorithm is based on `SHA-1`.

It is not recommended to use the

`mysql_native_password`

authentication plugin for new installations that require **high password security**. If someone is able to both listen to the connection protocol and get a copy of the `mysql.user` table, then the person would be able to use this information to connect to the MariaDB server. The [ed25519](#) authentication plugin is a more modern authentication plugin that provides simple password authentication using a more secure algorithm.

## Contents

1. [Installing the Plugin](#)
2. [Creating Users](#)
3. [Changing User Passwords](#)
4. [Client Authentication Plugins](#)
  1. [mysql\\_native\\_password](#)
5. [Support in Client Libraries](#)
6. [Known Old Issues \(Only Relevant for Old Installations\)](#)
  1. [Mismatches Between Password and authentication\\_string Columns](#)
7. [See Also](#)

## Installing the Plugin

The

```
mysql_native_password
```

authentication plugin is statically linked into the server, so no installation is necessary.

## Creating Users

The easiest way to create a user account with the

```
mysql_native_password
```

authentication plugin is to make sure that `old_passwords=0` is set, and then create a user account via `CREATE USER` that does not specify an authentication plugin, but does specify a password via the `IDENTIFIED BY` clause. For example:

```
SET old_passwords=0;
CREATE USER username@hostname IDENTIFIED BY 'mariadb';
```

If `SQL_MODE` does not have

```
NO_AUTO_CREATE_USER
```

set, then you can also create the user account via `GRANT`. For example:

```
SET old_passwords=0;
GRANT SELECT ON db.* TO username@hostname IDENTIFIED BY 'mariadb';
```

You can also create the user account by providing a password hash via the `IDENTIFIED BY PASSWORD` clause, and MariaDB will validate whether the password hash is one that is compatible with

```
mysql_native_password
```

. For example:

```
SET old_passwords=0;

SELECT PASSWORD('mariadb');
+-----+
| PASSWORD('mariadb')           |
+-----+
| *54958E764CE10E50764C2EECB71D01F08549980 |
+-----+

CREATE USER username@hostname
  IDENTIFIED BY PASSWORD '*54958E764CE10E50764C2EECB71D01F08549980';
```

Similar to all other `authentication plugins`, you could also specify the name of the plugin in the `IDENTIFIED VIA` clause while providing the password hash as the

```
USING
```

clause. For example:

```
CREATE USER username@hostname
  IDENTIFIED VIA mysql_native_password USING '*54958E764CE10E50764C2EECB71D01F08549980';
```

## Changing User Passwords

You can change a user account's password with the `SET PASSWORD` statement while providing the plain-text password as an argument to the `PASSWORD()` function. For example:

```
SET PASSWORD = PASSWORD('new_secret')
```

You can also change the user account's password with the `ALTER USER` statement. You would have to make sure that `old_passwords=0` is set, and then you would have to specify a password via the `IDENTIFIED BY` clause. For example:

```
SET old_passwords=0;
ALTER USER username@hostname IDENTIFIED BY 'new_secret';
```

## Client Authentication Plugins

For clients that use the

`libmysqlclient`  
or [MariaDB Connector/C](#) libraries, MariaDB provides one client authentication plugin that is compatible with the `mysql_native_password` authentication plugin:

- `mysql_native_password`

When connecting with a [client or utility](#) to a server as a user account that authenticates with the

`mysql_native_password` authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the `--plugin-dir` option. For example:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

However, the

`mysql_native_password` client authentication plugin is generally statically linked into client libraries like `libmysqlclient` or [MariaDB Connector/C](#), so this is not usually necessary.

### `mysql_native_password`

The

`mysql_native_password` client authentication plugin hashes the password before sending it to the server.

## Support in Client Libraries

The

`mysql_native_password` authentication plugin is one of the conventional authentication plugins, so all client libraries should support it.

## Known Old Issues (Only Relevant for Old Installations)

### Mismatches Between Password and `authentication_string` Columns

For compatibility reasons, the

`mysql_native_password` authentication plugin tries to read the password hash from both the `Password` and `authentication_string` columns in the `mysql.user` table. This has caused issues in the past if one of the columns had a different value than the other.

Starting with [MariaDB 10.2.19](#) and [MariaDB 10.3.11](#), `CREATE USER`, `ALTER USER`, `GRANT`, and `SET PASSWORD` will set both columns whenever an account's password is changed.

See [MDEV-16774](#) for more information.

## See Also

- [ed25519](#) secure connection plugin
- [History of MySQL and MariaDB authentication protocols](#)

## 4.4.5.3 Authentication Plugin - mysql\_old\_password

The

```
mysql_old_password
```

authentication plugin is the default authentication plugin that will be used for an account created when no authentication plugin is explicitly mentioned and

```
old_passwords=1
```

is set. It uses the pre-MySQL 4.1 password hashing algorithm, which is also used by the

```
OLD_PASSWORD()
```

function and by the

```
PASSWORD()
```

function when

```
old_passwords=1
```

is set.

It is not recommended to use the

```
mysql_old_password
```

authentication plugin for new installations. The password hashing algorithm is no longer as secure as it used to be, and the plugin is primarily provided for backward-compatibility. The

```
ed25519
```

authentication plugin is a more modern authentication plugin that provides simple password authentication.

### Contents

1. [Installing the Plugin](#)
2. [Creating Users](#)
3. [Changing User Passwords](#)
4. [Client Authentication Plugins](#)
  1. [mysql\\_old\\_password](#)
5. [Support in Client Libraries](#)

## Installing the Plugin

The

```
mysql_old_password
```

authentication plugin is statically linked into the server, so no installation is necessary.

## Creating Users

The easiest way to create a user account with the

```
mysql_old_password
```

authentication plugin is to make sure that

```
old_passwords=1
```

is set, and then create a user account via

```
CREATE USER
```

that does not specify an authentication plugin, but does specify a password via the

```
IDENTIFIED BY
```

clause. For example:

```
SET old_passwords=1;
CREATE USER username@hostname IDENTIFIED BY 'mariadb';
```

If

SQL\_MODE

does not have  
NO\_AUTO\_CREATE\_USER  
set, then you can also create the user via

GRANT

. For example:

```
SET old_passwords=1;
GRANT SELECT ON db.* TO username@hostname IDENTIFIED BY 'mariadb';
```

You can also create the user account by providing a password hash via the

IDENTIFIED BY PASSWORD

clause, and MariaDB will validate whether the password hash is one that is compatible with  
mysql\_old\_password

. For example:

```
SET old_passwords=1;
Query OK, 0 rows affected (0.000 sec)

SELECT PASSWORD('mariadb');
+-----+
| PASSWORD('mariadb') |
+-----+
| 021bec665bf663f1 |
+-----+
1 row in set (0.000 sec)

CREATE USER username@hostname IDENTIFIED BY PASSWORD '021bec665bf663f1';
Query OK, 0 rows affected (0.000 sec)
```

Similar to all other [authentication plugins](#), you could also specify the name of the plugin in the

IDENTIFIED VIA

clause while providing the password hash as the  
USING  
clause. For example:

```
CREATE USER username@hostname IDENTIFIED VIA mysql_old_password USING '021bec665bf663f1';
Query OK, 0 rows affected (0.000 sec)
```

## Changing User Passwords

You can change a user account's password with the

SET PASSWORD

statement while providing the plain-text password as an argument to the

PASSWORD()

function. For example:

```
SET PASSWORD = PASSWORD('new_secret')
```

You can also change the user account's password with the

ALTER USER

statement. You would have to make sure that

old\_passwords=1

is set, and then you would have to specify a password via the

`IDENTIFIED BY`

clause. For example:

```
SET old_passwords=1;
ALTER USER username@hostname IDENTIFIED BY 'new_secret';
```

## Client Authentication Plugins

For clients that use the

`libmysqlclient`  
or [MariaDB Connector/C](#) libraries, MariaDB provides one client authentication plugin that is compatible with the `mysql_old_password` authentication plugin:

- `mysql_old_password`

When connecting with a [client or utility](#) to a server as a user account that authenticates with the

`mysql_old_password` authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the `--plugin-dir` option. For example:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

However, the

`mysql_old_password` client authentication plugin is generally statically linked into client libraries like `libmysqlclient` or [MariaDB Connector/C](#), so this is not usually necessary.

### `mysql_old_password`

The

`mysql_old_password` client authentication plugin hashes the password before sending it to the server.

## Support in Client Libraries

The

`mysql_old_password` authentication plugin is one of the conventional authentication plugins, so all client libraries should support it.

### 4.4.5.4 Authentication Plugin - ed25519

MySQL has used SHA-1 based authentication since version 4.1. Since [MariaDB 5.2](#) this authentication plugin has been called

`mysql_native_password`

. Over the years as computers became faster, new attacks on SHA-1 were being developed. Nowadays SHA-1 is no longer considered as secure as it was in 2001. That's why the

`ed25519` authentication plugin was created.

The

`ed25519` authentication plugin uses [Elliptic Curve Digital Signature Algorithm \(ECDSA\)](#) to securely store users' passwords and to authenticate users. The `ed25519` algorithm is the same one that is [used by OpenSSH](#). It is based on the elliptic curve and code created by [Daniel J. Bernstein](#).

From a user's perspective, the

`ed25519` authentication plugin still provides conventional password-based authentication.

## Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Creating Users](#)
4. [Changing User Passwords](#)
5. [Client Authentication Plugins](#)
  1. [client\\_ed25519](#)
6. [Support in Client Libraries](#)
  1. [Using the Plugin with MariaDB Connector/C](#)
  2. [Using the Plugin with MariaDB Connector/ODBC](#)
  3. [Using the Plugin with MariaDB Connector/J](#)
  4. [Using the Plugin with MariaDB Connector/Node.js](#)
  5. [Using the Plugin with MySqlConnector for .NET](#)
7. [Versions](#)
8. [Options](#)
  1. [ed25519](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default as

`auth_ed25519.so`

or

`auth_ed25519.dll`

depending on the operating system, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

`INSTALL SONAME`

or

`INSTALL PLUGIN`

. For example:

```
INSTALL SONAME 'auth_ed25519';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

`--plugin-load`

or the

`--plugin-load-add`

options. This can be specified as a command-line argument to

`mysqld`

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = auth_ed25519
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

`UNINSTALL SONAME`

or

## UNINSTALL PLUGIN

. For example:

```
UNINSTALL SONAME 'auth_ed25519';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Creating Users

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, you can create a user account by executing the

```
CREATE USER
```

statement and providing the

```
IDENTIFIED VIA
```

clause followed by the the name of the plugin, which is  
ed25519  
, and providing the the  
USING  
clause followed by the

```
PASSWORD()
```

function with the plain-text password as an argument. For example:

```
CREATE USER username@hostname IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

If

```
SQL_MODE
```

does not have  
NO\_AUTO\_CREATE\_USER  
set, then you can also create the user account via

```
GRANT
```

. For example:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA ed25519 USING PASSWORD('secret');
```

MariaDB until [10.3](#)

In [MariaDB 10.3](#) and before, the

```
PASSWORD()
```

function and

```
SET PASSWORD
```

statement did not work with the  
ed25519  
authentication plugin. Instead, you would have to use the [UDF](#) that comes with the authentication plugin to calculate the password hash. For

example:

```
CREATE FUNCTION ed25519_password RETURNS STRING SONAME "auth_ed25519.so";
```

Now you can calculate a password hash by executing:

```
SELECT ed25519_password("secret");
+-----+
| SELECT ed25519_password("secret")           |
+-----+
| ZIgUREUg5PVgQ6LskhXm0+eZLS0nC8be6HPjYWR4YJY |
+-----+
```

Now you can use it to create the user account using the new password hash.

To create a user account via

[CREATE USER](#)

, specify the name of the plugin in the

[IDENTIFIED VIA](#)

clause while providing the password hash as the  
[USING](#)  
clause. For example:

```
CREATE USER username@hostname IDENTIFIED VIA ed25519 USING 'ZIgUREUg5PVgQ6LskhXm0+eZLS0nC8be6HPjYWR4YJY';
```

If

[SQL\\_MODE](#)

does not have

[NO\\_AUTO\\_CREATE\\_USER](#)

set, then you can also create the user account via

[GRANT](#)

. For example:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA ed25519 USING 'ZIgUREUg5PVgQ6LskhXm0+eZLS0nC8be6HPjYWR4YJY';
```

Note that users require a password in order to be able to connect. It is possible to create a user without specifying a password, but they will be unable to connect.

## Changing User Passwords

MariaDB starting with [10.4](#)

In [MariaDB 10.4](#) and later, you can change a user account's password by executing the

[SET PASSWORD](#)

statement followed by the

[PASSWORD\(\)](#)

function and providing the plain-text password as an argument. For example:

```
SET PASSWORD = PASSWORD('new_secret');
```

You can also change the user account's password with the

[ALTER USER](#)

statement. You would have to specify the name of the plugin in the

[IDENTIFIED VIA](#)

clause while providing the plain-text password as an argument to the

`PASSWORD()`

function in the

`USING`

clause. For example:

```
ALTER USER username@hostname IDENTIFIED VIA ed25519 USING PASSWORD('new_secret');
```

MariaDB until 10.3

In MariaDB 10.3 and before, the

`PASSWORD()`

function and

`SET PASSWORD`

statement did not work with the

`ed25519`

authentication plugin. Instead, you would have to use the [UDF](#) that comes with the authentication plugin to calculate the password hash. For example:

```
CREATE FUNCTION ed25519_password RETURNS STRING SONAME "auth_ed25519.so";
```

Now you can calculate a password hash by executing:

```
SELECT ed25519_password("secret");
+-----+
| SELECT ed25519_password("secret");      |
+-----+
| ZIgUREUg5PVgQ6LskhXm0+eZLS0nC8be6HPjYWR4YJY |
+-----+
```

Now you can change the user account's password using the new password hash.

You can change the user account's password with the

`ALTER USER`

statement. You would have to specify the name of the plugin in the

`IDENTIFIED VIA`

clause while providing the password hash as the

`USING`

clause. For example:

```
ALTER USER username@hostname IDENTIFIED VIA ed25519 USING 'ZIgUREUg5PVgQ6LskhXm0+eZLS0nC8be6HPjYWR4YJY';
```

## Client Authentication Plugins

For clients that use the

`libmysqlclient`

or [MariaDB Connector/C](#) libraries, MariaDB provides one client authentication plugin that is compatible with the

`ed25519`

authentication plugin:

- `client_ed25519`

When connecting with a [client or utility](#) to a server as a user account that authenticates with the

`ed25519`

authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the  
`--plugin-dir`

option. For example:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

client\_ed25519

The

client\_ed25519

client authentication plugin hashes and signs the password using the [Elliptic Curve Digital Signature Algorithm \(ECDSA\)](#) before sending it to the server.

## Support in Client Libraries

### Using the Plugin with MariaDB Connector/C

MariaDB Connector/C supports

ed25519

authentication using the [client authentication plugins](#) mentioned in the previous section since MariaDB Connector/C 3.1.0.

### Using the Plugin with MariaDB Connector/ODBC

MariaDB Connector/ODBC supports

ed25519

authentication using the [client authentication plugins](#) mentioned in the previous section since MariaDB Connector/ODBC 3.1.2.

### Using the Plugin with MariaDB Connector/J

MariaDB Connector/J supports

ed25519

authentication since MariaDB Connector/J 2.2.1.

### Using the Plugin with MariaDB Connector/Node.js

MariaDB Connector/Node.js supports

ed25519

authentication since MariaDB Connector/Node.js 2.1.0.

### Using the Plugin with MySqlConnector for .NET

MySqlConnector for ADO.NET supports

ed25519

authentication since MySqlConnector 0.56.0.

The connector implemented support for this authentication plugin in a separate [NuGet](#) package called

[MySqlConnector.Authentication.Ed25519](#)

. After the package is installed, your application must call

`Ed25519AuthenticationPlugin.Install`

to enable it.

## Versions

Version	Status	Introduced
1.1	Stable	MariaDB 10.4.0
1.0	Stable	MariaDB 10.3.8 , MariaDB 10.2.17 , MariaDB 10.1.35
1.0	Beta	MariaDB 10.2.5 , MariaDB 10.1.22

## Options

## ed25519

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF

- Disables the plugin without removing it from the

- `mysql.plugins`

- table.

- ON

- Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

- FORCE

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

- FORCE\_PLUS\_PERMANENT

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

- `UNINSTALL SONAME`

- or

- `UNINSTALL PLUGIN`

- while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

- `--ed25519=value`

- **Data Type:**

- enumerated

- **Default Value:**

- ON

- **Valid Values:**

- OFF

- ,

- ON

- ,

- FORCE

- ,

- FORCE\_PLUS\_PERMANENT

---

## 4.4.5.5 Authentication Plugin - GSSAPI

The

`gssapi`

authentication plugin allows the user to authenticate with services that use the [Generic Security Services Application Program Interface \(GSSAPI\)](#). Windows has a slightly different but very similar API called [Security Support Provider Interface \(SSPI\)](#). The GSSAPI is a standardized API described in [RFC2743](#) and [RFC2744](#). The client and server negotiate using a standardized protocol described in [RFC7546](#).

On Windows, this authentication plugin supports [Kerberos](#) and [NTLM](#) authentication. Windows authentication is supported regardless of whether a [domain](#) is used in the environment.

On Unix systems, the most dominant GSSAPI service is [Kerberos](#). However, it is less commonly used on Unix systems than it is on Windows. Regardless, this authentication plugin also supports Kerberos authentication on Unix.

The

`gssapi`

authentication plugin is most often used for authenticating with [Microsoft Active Directory](#).

This article gives instructions on configuring the

**gssapi**  
authentication plugin for MariaDB for passwordless login.

## Contents

1. [Installing the Plugin's Package](#)
  1. [Installing on Linux](#)
    1. [Installing with a Package Manager](#)
      1. [Installing with yum/dnf](#)
      2. [Installing with apt-get](#)
      3. [Installing with zypper](#)
    2. [Installing on Windows](#)
  2. [Installing the Plugin](#)
  3. [Uninstalling the Plugin](#)
  4. [Configuring the Plugin](#)
    1. [Creating a Kkeytab File on Unix](#)
      1. [Creating a Kkeytab File with Microsoft Active Directory](#)
      2. [Creating a Kkeytab File with MIT Kerberos](#)
    2. [Configuring the Path to the Kkeytab File on Unix](#)
    3. [Configuring the Service Principal Name](#)
  5. [Creating Users](#)
    1. [Creating users identified via group membership or SID \(Windows-specific\)](#)
  6. [Client Authentication Plugins](#)
    1. [auth\\_gssapi\\_client](#)
  7. [Support in Client Libraries](#)
    1. [Using the Plugin with MariaDB Connector/C](#)
    2. [Using the Plugin with MariaDB Connector/ODBC](#)
    3. [Using the Plugin with MariaDB Connector/J](#)
    4. [Using the Plugin with MariaDB Connector/Node.js](#)
    5. [Using the Plugin with MySqlConnector for .NET](#)
      1. [.NET specific problems/workarounds](#)
  8. [Versions](#)
  9. [System Variables](#)
    1. [gssapi\\_keytab\\_path](#)
    2. [gssapi\\_principal\\_name](#)
    3. [gssapi\\_mech\\_name](#)
  10. [Options](#)
    1. [gssapi](#)

## Installing the Plugin's Package

The

**gssapi**  
authentication plugin's shared library is included in MariaDB packages as the  
`auth_gssapi.so`  
or  
`auth_gssapi.dll`  
shared library on systems where it can be built. The plugin was first included in [MariaDB 10.1.11](#).

## Installing on Linux

The

**gssapi**  
authentication plugin is included in [binary tarballs](#) on Linux.

### Installing with a Package Manager

The

**gssapi**  
authentication plugin can also be installed via a package manager on Linux. In order to do so, your system needs to be configured to install from

one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the [MariaDB Package Repository setup script](#).

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the [MariaDB Repository Configuration Tool](#).

Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant [RPM package](#) from MariaDB's repository using

`yum`

or

`dnf`

. Starting with RHEL 8 and Fedora 22,

`yum`

has been replaced by

`dnf`

, which is the next major version of

`yum`

. However,

`yum`

commands still work on many systems that use

`dnf`

. For example:

```
sudo yum install MariaDB-gssapi-server
```

Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant [DEB package](#) from MariaDB's repository using

`apt-get`

. For example:

```
sudo apt-get install mariadb-plugin-gssapi-server
```

Installing with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant [RPM package](#) from MariaDB's repository using

`zypper`

. For example:

```
sudo zypper install MariaDB-gssapi-server
```

## Installing on Windows

The

`gssapi`

authentication plugin is included in [MSI](#) and [ZIP](#) packages on Windows.

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

`INSTALL SONAME`

or

## INSTALL PLUGIN

. For example:

```
INSTALL SONAME 'auth_gssapi';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = auth_gssapi
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'auth_gssapi';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Configuring the Plugin

If the MariaDB server is running on Unix, then some additional configuration steps will need to be implemented in order to use the plugin.

If the MariaDB server is running on Windows, then no special configuration steps will need to be implemented in order to use the plugin, as long as the following is true:

- The Windows server is joined to a domain.
- The MariaDB server process is running as either a [NetworkService Account](#) or a [Domain User Account](#).

## Creating a Keytab File on Unix

If the MariaDB server is running on Unix, then the KDC server will need to create a keytab file for the MariaDB server. The keytab file contains the service principal name, which is the identity that the MariaDB server will use to communicate with the KDC server. The keytab will need to be transferred to the MariaDB server, and the

```
mysqld
server process will need read access to this keytab file.
```

How this keytab file is generated depends on whether the KDC server is [Microsoft Active Directory KDC](#) or [MIT Kerberos KDC](#).

## Creating a Keytab File with Microsoft Active Directory

If you are using [Microsoft Active Directory KDC](#), then you may need to create a keytab using the

`ktpass.exe`

utility on a Windows host. The service principal will need to be mapped to an existing domain user. To do so, follow the steps listed below.

Be sure to replace the following items in the step below:

- Replace  
     `${HOST}`  
    with the fully qualified DNS name for the MariaDB server host.
- Replace  
     `${DOMAIN}`  
    with the Active Directory domain.
- Replace  
     `${AD_USER}`  
    with the existing domain user.
- Replace  
     `${PASSWORD}`  
    with the password for the service principal.

To create the service principal, execute the following:

```
ktpass.exe /princ mariadb/${HOST}@${DOMAIN} /mapuser ${AD_USER} /pass ${PASSWORD} /out mariadb.keytab /crypto all /ptype KRB5_NT
```

## Creating a Keytab File with MIT Kerberos

If you are using [MIT Kerberos KDC](#), then you can create a [keytab](#) file using the

`kadmin`

utility. To do so, follow the steps listed below.

In the following steps, be sure to replace

`${HOST}`  
    with the fully qualified DNS name for the MariaDB server host.

First, create the service principal using the

`kadmin`

utility. For example:

```
kadmin -q "addprinc -randkey mariadb/${HOST}"
```

Then, export the newly created user to the keytab file using the

`kadmin`

utility. For example:

```
kadmin -q "ktadd -k /path/to/mariadb.keytab mariadb/${HOST}"
```

More details can be found at the following links:

- [MIT Kerberos Documentation: Database administration](#)
- [MIT Kerberos Documentation: Application servers](#)

## Configuring the Path to the Keytab File on Unix

If the MariaDB server is running on Unix, then the path to the keytab file that was previously created can be set by configuring the

`gssapi_keytab_path`

system variable. This can be specified as a command-line argument to

`mysqld`

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
gssapi_keytab_path=/path/to/mariadb.keytab
```

## Configuring the Service Principal Name

The service principal name can be set by configuring the

`gssapi_principal_name`

system variable. This can be specified as a command-line argument to

`mysqld`

or it can be specified in a relevant server option group in an [option file](#). For example:

```
[mariadb]
...
gssapi_principal_name=service_principal_name/host.domain.com@REALM
```

If a service principal name is not provided, then the plugin will try to use

`mariadb/host.domain.com@REALM`  
by default.

If the MariaDB server is running on Unix, then the plugin needs a service principal name in order to function.

If the MariaDB server is running on Windows, then the plugin does not usually need a service principal in order to function. However, if you want to use one anyway, then one can be created with the

`setspn`

utility.

Different KDC implementations may use different canonical forms to identify principals. See [RFC2744: Section 3.10](#) to learn what the standard says about principal names.

More details can be found at the following links:

- [Active Directory Domain Services: Service Principal Names](#)
- [MIT Kerberos Documentation: Realm configuration decisions](#)
- [MIT Kerberos Documentation: Principal names and DNS](#)

## Creating Users

To create a user account via

`CREATE USER`

, specify the name of the plugin in the

`IDENTIFIED VIA`

clause. For example:

```
CREATE USER username@hostname IDENTIFIED VIA gssapi;
```

If

`SQL_MODE`

does not have  
`NO_AUTO_CREATE_USER`  
set, then you can also create the user account via

`GRANT`

. For example:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA gssapi;
```

You can also specify the user's [realm](#) for MariaDB with the  
USING  
clause. For example:

```
CREATE USER username@hostname IDENTIFIED VIA gssapi USING 'username@EXAMPLE.COM';
```

The format of the realm depends on the specific authentication mechanism that is used. For example, the format would need to be  
machine\\username  
for Windows users authenticating with NTLM.

If the realm is not provided in the user account's definition, then the realm is **not** used for comparison. Therefore, 'usr1@EXAMPLE.COM', 'usr1@EXAMPLE.CO.UK' and 'mymachine\usr1' would all identify as the following user account:

```
CREATE USER usr1@hostname IDENTIFIED VIA gssapi;
```

## Creating users identified via group membership or SID (Windows-specific)

Since 10.6.0, on Windows only, it is possible to login using a AD or local group-membership. This is achieved by using GROUP prefix in IDENTIFIED ... AS

```
CREATE USER root IDENTIFIED VIA gssapi AS 'GROUP:Administrators'  
CREATE USER root IDENTIFIED VIA gssapi AS 'GROUP:BUILTIN\\Administrators'
```

Effect of the above definition is that every user that identifies as member of group Administrators can login using user name root, passwordless.

User can also login using own or group [SID](#)

```
CREATE USER root IDENTIFIED VIA gssapi AS 'SID:S-1-5-32-544'
```

Using SIDs will perform slightly faster than using name (since it will spare translation between SID and name which is otherwise done), also SIDs immune against user or group renaming.

## Client Authentication Plugins

For clients that use the

libmysqlclient  
or [MariaDB Connector/C](#) libraries, MariaDB provides one client authentication plugin that is compatible with the  
gssapi  
authentication plugin:

- auth\_gssapi\_client

When connecting with a [client or utility](#) to a server as a user account that authenticates with the

gssapi  
authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the  
--plugin-dir  
option. For example:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

### auth\_gssapi\_client

The

auth\_gssapi\_client

client authentication plugin receives the principal name from the server, and then uses either the

[gss\\_init\\_sec\\_context](#)

function (on Unix) or the

[InitializeSecurityContext](#)

function (on Windows) to establish a security context on the client.

# Support in Client Libraries

## Using the Plugin with MariaDB Connector/C

MariaDB Connector/C supports

gssapi

authentication using the [client authentication plugins](#) mentioned in the previous section since MariaDB Connector/C 3.0.1.

## Using the Plugin with MariaDB Connector/ODBC

MariaDB Connector/ODBC supports

gssapi

authentication using the [client authentication plugins](#) mentioned in the previous section since MariaDB Connector/ODBC 3.0.0.

## Using the Plugin with MariaDB Connector/J

MariaDB Connector/J supports

gssapi

authentication since MariaDB Connector/J 1.4.0. Current documentation can be found [here](#).

## Using the Plugin with MariaDB Connector/Node.js

MariaDB Connector/Node.js does not yet support

gssapi

authentication. See [CONJS-72](#) for more information.

## Using the Plugin with MySqlConnector for .NET

MySqlConnector for ADO.NET supports

gssapi

authentication since MySqlConnector 0.47.0.

The support is transparent. Normally, the connector only needs to be provided the correct user name, and no other parameters are required.

However, this connector also supports the

[ServerSPN](#)

connection string parameter, which can be used for mutual authentication.

### .NET specific problems/workarounds

When connecting from Unix client to Windows server with ADO.NET, in an Active Directory domain environment, be aware that .NET Core on Unix does not support principal names in UPN(User Principal Name) form, which is default on Windows (e.g machine\$@domain.com). Thus, upon encountering an authentication exception with "server not found in Kerberos database", use one of workarounds below

- Force host-based SPN on server side.
  - For example, this can be done by setting the

[gssapi\\_principal\\_name](#)

system variable to  
HOST/machine  
in a server [option group](#) in an [option file](#).

- Pass host-based SPN on client side.
  - For example, this can be done by setting the connector's

[ServerSPN](#)

connection string parameter to  
HOST/machine

## Versions

Version	Status	Introduced
1.0	Stable	<a href="#">MariaDB 10.1.15</a>

## System Variables

### gssapi\_keytab\_path

- **Description:** Defines the path to the server's keytab file.
    - This system variable is only meaningful on [Unix](#) .
    - See [Creating a Keytab File on Unix](#) and [Configuring the Path to the Keytab File on Unix](#) for more information.
  - **Commandline:**

```
--gssapi-keytab-path
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
string
  - **Default Value:** "
  - **Introduced:** MariaDB 10.1.11
- 

### gssapi\_principal\_name

- **Description:** Name of the service principal.
    - See [Configuring the Service Principal Name](#) for more information.
  - **Commandline:**

```
--gssapi-principal-name
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
string
  - **Default Value:** "
  - **Introduced:** MariaDB 10.1.11
- 

### gssapi\_mech\_name

- **Description:** Name of the SSPI package used by server. Can be either 'Kerberos' or 'Negotiate'. Set it to 'Kerberos', to prevent less secure NTLM in domain environments, but leave it as default (Negotiate) to allow non-domain environments (e.g if server does not run in a domain environment).
    - This system variable is only meaningful on [Windows](#) .
  - **Commandline:**

```
--gssapi-mech-name
```
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
enumerated
  - **Default Value:**  
Negotiate
  - **Valid Values:**  
Kerberos  
,Negotiate
  - **Introduced:** MariaDB 10.1.11
-

## Options

### gssapi

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF

- Disables the plugin without removing it from the

- `mysql.plugins`

- table.

- ON

- Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

- FORCE

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

- FORCE\_PLUS\_PERMANENT

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

- `UNINSTALL SONAME`

- or

- `UNINSTALL PLUGIN`

- while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

- `--gssapi=value`

- **Data Type:**

- enumerated

- **Default Value:**

- ON

- **Valid Values:**

- OFF

- ,

- ON

- ,

- FORCE

- ,

- FORCE\_PLUS\_PERMANENT

- **Introduced:** MariaDB 10.1.11

## 4.4.5.6 Authentication with Pluggable Authentication Modules (PAM)



### Authentication Plugin - PAM

Uses the Pluggable Authentication Module (PAM) framework to authenticate MariaDB users.



### User and Group Mapping with PAM

Configure PAM to map a given PAM user or group to a different MariaDB user.



### Configuring PAM Authentication and User Mapping with Unix Authentication

Walkthrough configuration of PAM authentication and user mapping with Unix authentication.



## Configuring PAM Authentication and User Mapping with LDAP Authentication

*Configuring PAM authentication and user mapping with LDAP authentication.*

### 4.4.5.7 Authentication Plugin - Unix Socket

MariaDB starting with 10.4.3

In MariaDB 10.4.3 and later, the

```
unix_socket  
authentication plugin is installed by default, and it is used by the  
'root'@'localhost'  
user account by default. See Authentication from MariaDB 10.4 for more information.
```

The

```
unix_socket
```

authentication plugin allows the user to use operating system credentials when connecting to MariaDB via the local Unix socket file. This Unix socket file is defined by the [socket](#) system variable.

The

```
unix_socket
```

```
authentication plugin works by calling the getsockopt system call with the  
SO_PEERCRED  
socket option, which allows it to retrieve the  
uid  
of the process that is connected to the socket. It is then able to get the user name associated with that  
uid
```

. Once it has the user name, it will authenticate the connecting user as the MariaDB account that has the same user name.

The

```
unix_socket
```

authentication plugin is not suited to multiple Unix users accessing a single MariaDB user account.

## Contents

1. [Security](#)
  1. [Strengths](#)
  2. [Weaknesses](#)
2. [Disabling the Plugin](#)
3. [Installing the Plugin](#)
4. [Uninstalling the Plugin](#)
5. [Creating Users](#)
6. [Switching to Password-based Authentication](#)
7. [Client Authentication Plugins](#)
8. [Support in Client Libraries](#)
9. [Example](#)
10. [Versions](#)
11. [Options](#)
  1. [unix\\_socket](#)
12. [See Also](#)

## Security

A

```
unix_socket
```

authentication plugin is a passwordless security mechanism. Its security is in the strength of the access to the Unix user rather than the complexity and the secrecy of the password. As the security is different from passwords, the strengths and weaknesses need to be considered, and these aren't the same in every installation.

## Strengths

- Access is limited to the Unix user so, for example, a  

```
www-data  
user cannot access  
root  
with the  
unix_socket  
authentication plugin.
```
- There is no password to brute force.
- There is no password that can be accidentally exposed by user accident, poor security on backups, or poor security on passwords in configuration files.

- Default Unix user security is usually strong on preventing remote access and password brute force attempts.

## Weaknesses

The strength of a

`unix_socket`

authentication plugin is effectively the strength of the security of the Unix users on the system. The Unix user default installation in most cases is sufficiently secure, however, business requirements or unskilled management may expose risks. The following is a non-exhaustive list of potential Unix user security issues that may arise.

- Common access areas without screen locks, where an unauthorized user accesses the logged in Unix user of an authorized user.
- Extensive sudo access grants that provide users with access to execute commands of a different Unix user.
- Scripts writable by Unix users other than the Unix user that are executed (cron or directly) by the unix user.
- Web pages that are susceptible to command injection, where the Unix user running the web page has elevated privileges in the database that weren't intended to be used.
- Poor Unix user password practices including weak user passwords, password exposure and password reuse accompanied by an access vulnerability/mechanism of an unauthorized user to exploit this weakness.
- Weak remote access mechanisms and network file system privileges.
- Poor user security behavior including running untrusted scripts and software.

In some of these scenarios a database password may prevent these security exploits, however it will remove all the strengths of the

`unix_socket`

authentication plugin previously mentioned.

## Disabling the Plugin

MariaDB starting with [10.4.3](#)

In [MariaDB 10.4.3](#) and later, the

`unix_socket`

authentication plugin is installed by default, so **if you do not want it to be available by default on those versions, then you will need to disable it**.

The

`unix_socket`

authentication plugin is also installed by default in **new installations** that use the

`.deb`

packages provided by Debian's default repositories in Debian 9 and later and Ubuntu's default repositories in Ubuntu 15.10 and later, so **if you do not want it to be available by default on those systems when those packages are used, then you will need to disable it**. See [Differences in MariaDB in Debian \(and Ubuntu\)](#) for more information.

The

`unix_socket`

authentication plugin can be disabled by starting the server with the

`unix_socket`

option set to

`OFF`

. This can be specified as a command-line argument to

`mysqld`

or it can be specified in a relevant server `option group` in an `option file`. For example:

```
[mariadb]
...
unix_socket=OFF
```

As an alternative, the

`unix_socket`

option can also be set to

`OFF`

by pairing the option with the

```
disable  
option prefix . For example:
```

```
[mariadb]  
...  
disable_unix_socket
```

## Installing the Plugin

MariaDB starting with [10.4.3](#)

In [MariaDB 10.4.3](#) and later, the `unix_socket` authentication plugin is installed by default, so **this step can be skipped on those versions**.

The

```
unix_socket  
authentication plugin is also installed by default in new installations that use the  
.deb
```

packages provided by Debian's default repositories in Debian 9 and later and Ubuntu's default repositories in Ubuntu 15.10 and later, so **this step can be skipped on those systems when those packages are used**. See [Differences in MariaDB in Debian \(and Ubuntu\)](#) for more information.

In other systems, although the plugin's shared library is distributed with MariaDB by default as

```
auth_socket.so  
, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.
```

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'auth_socket';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]  
...  
plugin_load_add = auth_socket
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

## UNINSTALL PLUGIN

. For example:

```
UNINSTALL SONAME 'auth_socket';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Creating Users

To create a user account via

```
CREATE USER
```

, specify the name of the plugin in the

```
IDENTIFIED VIA
```

clause. For example:

```
CREATE USER username@hostname IDENTIFIED VIA unix_socket;
```

If

```
SQL_MODE
```

does not have

```
NO_AUTO_CREATE_USER
```

set, then you can also create the user account via

```
GRANT
```

. For example:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA unix_socket;
```

## Switching to Password-based Authentication

Sometimes Unix socket authentication does not meet your needs, so it can be desirable to switch a user account back to password-based authentication. This can easily be done by telling MariaDB to use another [authentication plugin](#) for the account by executing the

```
ALTER USER
```

statement. The specific authentication plugin is specified with the

```
IDENTIFIED VIA
```

clause. For example, if you wanted to switch to the

```
mysql_native_password
```

authentication plugin, then you could execute:

```
ALTER USER root@localhost IDENTIFIED VIA mysql_native_password;
SET PASSWORD = PASSWORD('foo');
```

Note that if your operating system has scripts that require password-less access to MariaDB, then this may break those scripts. You may be able to fix that by setting a password in the

```
[client]
option group in your /root/.my.cnf option file . For example:
```

```
[client]
password=foo
```

## Client Authentication Plugins

The

```
unix_socket
authentication plugin does not require any specific client authentication plugins. It should work with all clients.
```

## Support in Client Libraries

The

```
unix_socket
authentication plugin does not require any special support in client libraries. It should work with all client libraries.
```

## Example

```
$ mysql -uroot
MariaDB []> CREATE USER serg IDENTIFIED VIA unix_socket;
MariaDB []> CREATE USER monty IDENTIFIED VIA unix_socket;
MariaDB []> quit
Bye
$ whoami
serg
$ mysql --user=serg
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.2.0-MariaDB-alpha-debug Source distribution
MariaDB []> quit
Bye
$ mysql --user=monty
ERROR 1045 (28000): Access denied for user 'monty'@'localhost' (using password: NO)
```

In this example, a user

```
serg
is already logged into the operating system and has full shell access. He has already authenticated with the operating system and his MariaDB account is configured to use the
unix_socket
authentication plugin, so he does not need to authenticate again for the database. MariaDB accepts his operating system credentials and allows him to connect. However, any attempt to connect to the database as another operating system user will be denied.
```

## Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.0.11
1.0	Beta	MariaDB 5.2.0

## Options

### unix\_socket

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- 

- OFF  
- Disables the plugin without removing it from the

- [mysql.plugin](#)

table.

ON

- Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

FORCE

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

FORCE\_PLUS\_PERMANENT

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

[UNINSTALL SONAME](#)

or

[UNINSTALL PLUGIN](#)

while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

`--unix-socket=value`

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF

,

ON

,

FORCE

,

FORCE\_PLUS\_PERMANENT

## See Also

- [Differences in MariaDB in Debian \(and Ubuntu\)](#)
- [Authentication from MariaDB 10.4](#)
- [Authentication from MariaDB 10.4 video tutorial](#)

## 4.4.5.8 Authentication Plugin - Named Pipe

The

`named_pipe`

authentication plugin allows the user to use operating system credentials when connecting to MariaDB via named pipe on Windows. Named pipe connections are enabled by the

[named\\_pipe](#)

system variable.

The

`named_pipe`

authentication plugin works by using [named pipe impersonation](#) and calling `GetUserName()`

to retrieve the user name of the process that is connected to the named pipe. Once it has the user name, it authenticates the connecting user as the MariaDB account that has the same user name.

## Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Creating Users](#)
4. [Client Authentication Plugins](#)
5. [Support in Client Libraries](#)
6. [Example](#)
7. [Versions](#)
8. [Options](#)
  1. [named\\_pipe](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'auth_named_pipe';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = auth_named_pipe
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'auth_named_pipe';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Creating Users

To create a user account via

[CREATE\\_USER](#)

, specify the name of the plugin in the

[IDENTIFIED VIA](#)

clause. For example:

```
CREATE USER username@hostname IDENTIFIED VIA named_pipe;
```

If

[SQL\\_MODE](#)

does not have

[NO\\_AUTO\\_CREATE\\_USER](#)

set, then you can also create the user account via

[GRANT](#)

. For example:

```
GRANT SELECT ON db.* TO username@hostname IDENTIFIED VIA named_pipe;
```

## Client Authentication Plugins

The

[named\\_pipe](#)

authentication plugin does not require any specific client authentication plugins. It should work with all clients.

## Support in Client Libraries

The

[named\\_pipe](#)

authentication plugin does not require any special support in client libraries. It should work with all client libraries.

## Example

```
CREATE USER wlad IDENTIFIED VIA named_pipe;
CREATE USER monty IDENTIFIED VIA named_pipe;
quit

C:\>echo %USERNAME%
wlad

C:\> mysql --user=wlad --protocol=PIPE
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.1.12-MariaDB-debug Source distribution

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> quit
Bye

C:\> mysql --user=monty --protocol=PIPE
ERROR 1698 (28000): Access denied for user 'monty'@'localhost'
```

In this example, a user

wlad

is already logged into the system. Because he has identified himself to the operating system, he does not need to do it again for the database — MariaDB trusts the operating system credentials. However, he cannot connect to the database as another user.

## Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.1.11

## Options

### `named_pipe`

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF

- Disables the plugin without removing it from the

- `mysql.plugins`

- table.

- ON

- Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

- FORCE

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

- FORCE\_PLUS\_PERMANENT

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

- `UNINSTALL SONAME`

- or

- `UNINSTALL PLUGIN`

- while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.
  - There may be ambiguity between this option and the

- `named_pipe`

- system variable. See [MDEV-19625](#) about that.

- **Commandline:**

- `--named-pipe=value`

- **Data Type:**

- enumerated

- **Default Value:**

- ON

- **Valid Values:**

- OFF

- ,

- ON

- ,

- FORCE

- ,

- FORCE\_PLUS\_PERMANENT

- **Introduced:** [MariaDB 10.1.11](#)

## 4.4.5.9 Authentication Plugin - SHA-256

### Contents

1. [Support in MariaDB Server](#)
2. [Client Authentication Plugins](#)
  1. [sha256\\_password](#)
  2. [caching\\_sha2\\_password](#)
3. [Support in Client Libraries](#)
  1. [Using the Plugin with MariaDB Connector/C](#)
  2. [Using the Plugin with MariaDB Connector/ODBC](#)
  3. [Using the Plugin with MariaDB Connector/J](#)
  4. [Using the Plugin with MariaDB Connector/Node.js](#)
4. [See Also](#)

MySQL 5.6 added support for the

`sha256_password`

authentication plugin, and MySQL 8.0 also added support for the

`caching_sha2_password`

authentication plugin.

The

`caching_sha2_password`

plugin is now the default authentication plugin in MySQL 8.0.4 and above, based on the value of the

`default_authentication_plugin`

system variable.

## Support in MariaDB Server

MariaDB Server does not currently support either the

`sha256_password`

or the

`caching_sha2_password`

authentication plugins. See [MDEV-9804](#) for more information.

MariaDB Server does not support either of these authentication plugins. This is mainly because:

- To use the protocol, one has to distribute the server's public key to all MariaDB users, which can be cumbersome and impractical.
- The server gets the password in clear text which can cause problems if the user is convinced to connect to a malicious server.

## Client Authentication Plugins

For clients that use the [MariaDB Connector/C](#) library, MariaDB provides two client authentication plugins that are compatible with MySQL's SHA-256 authentication plugins:

- `sha256_password`
- `caching_sha2_password`

When connecting with a [client or utility](#) to a server as a user account that authenticates with the

`sha256_password`

or

caching\_sha256\_password  
authentication plugin, you may need to tell the client where to find the relevant client authentication plugin by specifying the  
--plugin-dir  
option. For example:

```
mysql --plugin-dir=/usr/local/mysql/lib64/mysql/plugin --user=alice
```

For clients that use MariaDB's  
`libmysqlclient` library instead of [MariaDB Connector/C](#), these client authentication plugins are not supported.

## sha256\_password

The

`sha256_password`  
client authentication plugin is compatible with MySQL's

[sha256\\_password](#)

authentication plugin, which was added in MySQL 5.6.

## caching\_sha256\_password

The

`caching_sha256_password`  
client authentication plugin is compatible with MySQL's

[caching\\_sha2\\_password](#)

authentication plugin, which was added in MySQL 8.0.

The

`caching_sha2_password`  
plugin is now the default authentication plugin in MySQL 8.0.4 and above, based on the value of the

[default\\_authentication\\_plugin](#)

system variable.

# Support in Client Libraries

## Using the Plugin with MariaDB Connector/C

[MariaDB Connector/C](#) supports

`sha256_password`  
and  
`caching_sha2_password`  
authentication using the [client authentication plugins](#) mentioned in the previous section.

It has supported the

`sha256_password`  
client authentication plugin since MariaDB Connector/C 3.0.2. See [CONC-229](#) for more information.

It has supported the

`caching_sha256_password`  
client authentication plugin since MariaDB Connector/C 3.0.8 and MariaDB Connector/C 3.1.0. See [CONC-312](#) for more information.

## Using the Plugin with MariaDB Connector/ODBC

[MariaDB Connector/ODBC](#) supports

`sha256_password`  
and  
`caching_sha2_password`  
authentication using the [client authentication plugins](#) mentioned in the previous section.

It has supported  
sha256\_password  
and  
caching\_sha2\_password  
authentication since MariaDB Connector/ODBC 3.1.4. See [ODBC-241](#) for more information.

## Using the Plugin with MariaDB Connector/J

[MariaDB Connector/J](#) supports  
sha256\_password  
and  
caching\_sha2\_password  
authentication since MariaDB Connector/J 2.5.0. See [CONJ-327](#) and [CONJ-663](#) for more information.

## Using the Plugin with MariaDB Connector/Node.js

[MariaDB Connector/Node.js](#) supports  
sha256\_password  
and  
caching\_sha2\_password  
authentication since MariaDB Connector/Node.js 2.5.0. See [CONJS-76](#) and [CONJS-77](#) for more information.

## See Also

- [MDEV-9804](#) contains the plans to use if we ever decide to support these protocols.
- [History of MySQL and MariaDB authentication protocols](#)

## 4.4.6 Password Validation Plugins

### 4.4.6.1 Simple Password Check Plugin

simple\_password\_check

is a [password validation](#) plugin. It can check whether a password contains at least a certain number of characters of a specific type. When first installed, a password is required to be at least eight characters, and requires at least one digit, one uppercase character, one lowercase character, and one character that is neither a digit nor a letter.

Note that passwords can be directly set as a hash, bypassing the password validation, if the [strict\\_password\\_validation](#) variable is

OFF  
(it is  
ON  
by default).

#### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Example](#)
4. [Known Issues](#)
  1. [Issues with PAM Authentication Plugin](#)
5. [Versions](#)
6. [System Variables](#)
  1. [simple\\_password\\_check\\_digits](#)
  2. [simple\\_password\\_check\\_letters\\_same\\_case](#)
  3. [simple\\_password\\_check\\_minimal\\_length](#)
  4. [simple\\_password\\_check\\_other\\_characters](#)
7. [Options](#)
  1. [simple\\_password\\_check](#)
8. [See Also](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

`INSTALL SONAME`

or

## INSTALL PLUGIN

. For example:

```
INSTALL SONAME 'simple_password_check';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = simple_password_check
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'simple_password_check';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Example

When creating a new password, if the criteria are not met, the following error is returned:

```
SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('abc');
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

## Known Issues

### Issues with PAM Authentication Plugin

Prior to [MariaDB 10.4.0](#), all [password validation plugins](#) are incompatible with the

```
pam
```

authentication plugin. See [Authentication Plugin - PAM: Conflicts with Password Validation](#) for more information.

# Versions

Version	Status	Introduced
1.0	Stable	<a href="#">MariaDB 10.1.18</a>
1.0	Gamma	<a href="#">MariaDB 10.1.13</a>
1.0	Beta	<a href="#">MariaDB 10.1.11</a>
1.0	Alpha	<a href="#">MariaDB 10.1.2</a>

## System Variables

### simple\_password\_check\_digits

- **Description:** A password must contain at least this many digits.

- **Commandline:**

```
--simple-password-check-digits=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0

to

1000

### simple\_password\_check\_letters\_same\_case

- **Description:** A password must contain at least this many upper-case and this many lower-case letters.

- **Commandline:**

```
--simple-password-check-letters-same-case=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0

to

1000

### simple\_password\_check\_minimal\_length

- **Description:** A password must contain at least this many characters.

- **Commandline:**

```
--simple-password-check-minimal-length=#
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

8

- **Range:**

0

to

1000

---

simple\_password\_check\_other\_characters

- **Description:** A password must contain at least this many characters that are neither digits nor letters.

- **Commandline:**

--simple-password-check-other-characters=#

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

1

- **Range:**

0

to

1000

---

## Options

simple\_password\_check

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF

- Disables the plugin without removing it from the

- `mysql.plugins`

- table.

- ON

- Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

- FORCE

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

- FORCE\_PLUS\_PERMANENT

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

- `UNINSTALL SONAME`

- or

- `UNINSTALL PLUGIN`

- while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

```
--simple-password-check=value
```

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF

,

ON

,

FORCE

,

FORCE\_PLUS\_PERMANENT

## See Also

- [Password Validation](#)
- [cracklib\\_password\\_check plugin](#) - use the Cracklib password-strength checking library

## 4.4.6.2 Cracklib Password Check Plugin

### Contents

1. [Installing the Plugin's Package](#)
  1. [Installing on Linux](#)
    1. [Installing with a Package Manager](#)
      1. [Installing with yum/dnf](#)
      2. [Installing with apt-get](#)
      3. [Installing with zypper](#)
  2. [Installing the Plugin](#)
  3. [Uninstalling the Plugin](#)
  4. [Viewing CrackLib Errors](#)
  5. [Example](#)
  6. [Known Issues](#)
    1. [Issues with PAM Authentication Plugin](#)
    2. [SELinux](#)
  7. [Versions](#)
  8. [System Variables](#)
    1. [cracklib\\_password\\_check\\_dictionary](#)
  9. [Options](#)
    1. [cracklib\\_password\\_check](#)
  10. [See Also](#)

### cracklib\_password\_check

is a [password validation](#) plugin. It uses the [CrackLib](#) library to check the strength of new passwords. CrackLib is installed by default in many Linux distributions, since the system's [Pluggable Authentication Module \(PAM\)](#) authentication framework is usually configured to check the strength of new passwords with the

### pam\_cracklib

PAM module.

Note that passwords can be directly set as a hash, bypassing the password validation, if the [strict\\_password\\_validation](#) variable is

OFF

(it is

ON

by default).

The plugin requires at least cracklib 2.9.0, so it is not available on Debian/Ubuntu builds before Debian 8 Jessie/Ubuntu 14.04 Trusty, RedHat Enterprise Linux / CentOS 6.

## Installing the Plugin's Package

The

### cracklib\_password\_check

plugin's shared library is included in MariaDB packages as the `cracklib_password_check.so` or `cracklib_password_check.dll` shared library on systems where it can be built.

## Installing on Linux

The

`cracklib_password_check` plugin is included in `systemd` [binary tarballs](#) on Linux, but not in the older generic and `glibc_214` tarballs.

### Installing with a Package Manager

The

`cracklib_password_check` plugin can also be installed via a package manager on Linux. In order to do so, your system needs to be configured to install from one of the MariaDB repositories.

You can configure your package manager to install it from MariaDB Corporation's MariaDB Package Repository by using the [MariaDB Package Repository setup script](#).

You can also configure your package manager to install it from MariaDB Foundation's MariaDB Repository by using the [MariaDB Repository Configuration Tool](#).

#### Installing with yum/dnf

On RHEL, CentOS, Fedora, and other similar Linux distributions, it is highly recommended to install the relevant [RPM package](#) from MariaDB's repository using

`yum`

or

`dnf`

. Starting with RHEL 8 and Fedora 22,

`yum`

has been replaced by

`dnf`

, which is the next major version of

`yum`

. However,

`yum`

commands still work on many systems that use

`dnf`

. For example:

```
sudo yum install MariaDB-cracklib-password-check
```

#### Installing with apt-get

On Debian, Ubuntu, and other similar Linux distributions, it is highly recommended to install the relevant [DEB package](#) from MariaDB's repository using

`apt-get`

. For example:

```
sudo apt-get install mariadb-plugin-cracklib-password-check
```

#### Installing with zypper

On SLES, OpenSUSE, and other similar Linux distributions, it is highly recommended to install the relevant [RPM package](#) from MariaDB's repository using

`zypper`

. For example:

```
sudo zypper install MariaDB-cracklib-password-check
```

## Installing the Plugin

Once the shared library is in place, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'cracklib_password_check';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = cracklib_password_check
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'cracklib_password_check';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Viewing CrackLib Errors

If password validation fails, then the original CrackLib error message can be viewed by executing

```
SHOW WARNINGS
```

## Example

When creating a new password, if the criteria are not met, the following error is returned:

```
SET PASSWORD FOR 'bob'@'%loc.gov' = PASSWORD('abc');
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

## Known Issues

### Issues with PAM Authentication Plugin

Prior to [MariaDB 10.4.0](#), all [password validation plugins](#) are incompatible with the

pam

authentication plugin. See [Authentication Plugin - PAM: Conflicts with Password Validation](#) for more information.

### SELinux

When using the standard [SELinux](#) policy with the mode set to

```
enforcing
,
mysqld
does not have access to
/usr/share/cracklib
, and you may see the following error when attempting to use the
cracklib_password_check
plugin:
```

```
CREATE USER `user`@`hostname` IDENTIFIED BY 's0mePwd123.';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

```
SHOW WARNINGS;
+-----+-----+
| Level | Code | Message
+-----+-----+
| Warning | 1819 | cracklib: error loading dictionary
| Error   | 1819 | Your password does not satisfy the current policy requirements |
| Error   | 1396 | Operation CREATE USER failed for 'user'@'hostname'           |
+-----+-----+
```

And the SELinux

```
audit.log
will contain errors like the following:
```

```
type=AVC msg=audit(1548371977.821:66): avc: denied { read } for pid=3537 comm="mysqld" name="pw_dict.pwd" dev="xvda2" ino=564
type=SYSCALL msg=audit(1548371977.821:66): arch=c000003e syscall=2 success=no exit=-13 a0=7fdd2a674580 a1=0 a2=1b6 a3=1b items=0
```

This can be fixed by creating an SELinux policy that allows

```
mysqld
to load the CrackLib dictionary. For example:
```

```

cd /usr/share/mysql/policy/selinux/
tee ./mariadb-plugin-cracklib-password-check.te <<EOF

module mariadb-plugin-cracklib-password-check 1.0;

require {
    type mysqld_t;
    type crack_db_t;
    class file { execute setattr read create getattr execute_no_trans write ioctl open append unlink };
    class dir { write search getattr add_name read remove_name open };
}

allow mysqld_t crack_db_t:dir { search read open };
allow mysqld_t crack_db_t:file { getattr read open };
EOF
sudo yum install selinux-policy-devel
make -f /usr/share/selinux/devel/Makefile mariadb-plugin-cracklib-password-check.pp
sudo semodule -i mariadb-plugin-cracklib-password-check.pp

```

See [MDEV-18374](#) for more information.

## Versions

Version	Status	Introduced
1.0	Stable	<a href="#">MariaDB 10.1.18</a>
1.0	Gamma	<a href="#">MariaDB 10.1.13</a>
1.0	Alpha	<a href="#">MariaDB 10.1.2</a>

## System Variables

### cracklib\_password\_check\_dictionary

- **Description:** Sets the path to the CrackLib dictionary. If not set, the default CrackLib dictionary path is used. The parameter expects the base name of a cracklib dictionary (a set of three files with endings

```

.hwm
,
.pwd
,
.pwi
), not a directory path.

```

- **Commandline:**

```
--cracklib-password-check-dictionary=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
string
```

- **Default Value:** Depends on the system. Often

```
/usr/share/cracklib/pw_dict
```

## Options

### cracklib\_password\_check

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- 

- OFF

- Disables the plugin without removing it from the

`mysql.plugins`

table.

- - ON
    - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
  - - FORCE
      - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - - FORCE\_PLUS\_PERMANENT
        - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

`UNINSTALL SONAME`

or

`UNINSTALL PLUGIN`

while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

`--cracklib-password-check=value`

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF

,

ON

,

FORCE

,

FORCE\_PLUS\_PERMANENT

## See Also

- [Password Validation](#)
- [simple\\_password\\_check plugin](#) - permits the setting of basic criteria for passwords

## 4.4.6.3 Password Reuse Check Plugin

MariaDB starting with 10.7

`password_reuse_check`  
is a [password validation](#) plugin introduced in [MariaDB 10.7.0](#).

### Contents

1. [Description](#)
  1. [Installing the Plugin](#)
  2. [Uninstalling the Plugin](#)
2. [Example](#)
3. [Versions](#)
4. [See Also](#)

## Description

The plugin is used to prevent a user from reusing a password, which can be a requirement in some security policies. The `password_reuse_check_interval` system variable determines the retention period, in days, for a password. By default this is zero, meaning unlimited retention. Old passwords are stored in the `mysql.password_reuse_check_history` table.

Note that passwords can be directly set as a hash, bypassing the password validation, if the `strict_password_validation` variable is OFF (it is ON by default).

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default.

You can install the plugin dynamically, without restarting the server, by executing `INSTALL SONAME` or `INSTALL PLUGIN`. For example:

```
INSTALL SONAME 'password_reuse_check';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the `--plugin-load` or the `--plugin-load-add` options. This can be specified as a command-line argument to `mysqld` or it can be specified in a relevant server option group in an option file . For example:

```
[mariadb]
...
plugin_load_add = password_reuse_check
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing `UNINSTALL SONAME` or `UNINSTALL PLUGIN`. For example:

```
UNINSTALL SONAME 'password_reuse_check';
```

If you installed the plugin by providing the `--plugin-load` or the `--plugin-load-add` options in a relevant server option group in an option file , then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Example

```
INSTALL SONAME 'password_reuse_check';

GRANT SELECT ON *.* TO user1@localhost identified by 'pwd1';
Query OK, 0 rows affected (0.038 sec)

GRANT SELECT ON *.* TO user1@localhost identified by 'pwd1';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements

GRANT SELECT ON *.* TO user1@localhost identified by 'pwd2';
Query OK, 0 rows affected (0.003 sec)

GRANT SELECT ON *.* TO user1@localhost identified by 'pwd1';
ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
```

## Versions

Version	Status	Introduced
1.0	Alpha	MariaDB 10.7.0

## See Also

- [Password Validation](#)
- [10.7 preview feature: Password Reuse Check plugin](#) (mariadb.org blog post)

## 4.4.6.4 Password Validation Plugin API

## Contents

- 1. SQL-Level Extensions
  - 1. Password-Changing Statements
    - 1. With Plain Text Password
    - 2. With Password Hash
  - 2. Examples
  - 3. Plugin API

“Password validation” means ensuring that user passwords meet certain minimal security requirements. A dedicated plugin API allows the creation of password validation plugins that will check user passwords as they are set (in `SET PASSWORD` and `GRANT` statements) and either allow or reject them.

## SQL-Level Extensions

MariaDB comes with three password validation plugins — the `simple_password_check` plugin, the `cracklib_password_check` plugin and the `password_reuse_check` plugin. They are not enabled by default; use `INSTALL SONAME` (or `INSTALL PLUGIN`) statement to install them.

When at least one password plugin is loaded, all new passwords will be validated and password-changing statements will fail if the password will not pass validation checks. Several password validation plugin can be loaded at the same time — in this case a password must pass **all** validation checks by **all** plugins.

### Password-Changing Statements

One can use various SQL statements to change a user password:

#### With Plain Text Password

```
SET PASSWORD = PASSWORD('plain-text password');
SET PASSWORD FOR `user`@`host` = PASSWORD('plain-text password');
SET PASSWORD = OLD_PASSWORD('plain-text password');
SET PASSWORD FOR `user`@`host` = OLD_PASSWORD('plain-text password');
CREATE USER `user`@`host` IDENTIFIED BY 'plain-text password';
GRANT privileges TO `user`@`host` IDENTIFIED BY 'plain-text password';
```

These statements are subject to password validation. If at least one password validation plugin is loaded, plain-text passwords specified in these statements will be validated.

#### With Password Hash

```
SET PASSWORD = 'password hash';
SET PASSWORD FOR `user`@`host` = 'password hash';
CREATE USER `user`@`host` IDENTIFIED BY PASSWORD 'password hash';
CREATE USER `user`@`host` IDENTIFIED VIA mysql_native_password USING 'password hash';
CREATE USER `user`@`host` IDENTIFIED VIA mysql_old_password USING 'password hash';
GRANT privileges TO `user`@`host` IDENTIFIED BY PASSWORD 'password hash';
GRANT privileges TO `user`@`host` IDENTIFIED VIA mysql_native_password USING 'password hash';
GRANT privileges TO `user`@`host` IDENTIFIED VIA mysql_old_password USING 'password hash';
```

These statements can not possibly use password validation — there is nothing to validate, the original plain-text password is not available. MariaDB introduces a **strict password validation** mode — controlled by a `strict_password_validation` global server variable. If the strict password validation is enabled and at least one password validation plugin is loaded then these “unvalidatable” passwords will be rejected. Otherwise they will be accepted. By default a strict password validation is enabled (but note that it has no effect if no password validation plugin is loaded).

## Examples

Failed password validation:

```
GRANT SELECT ON *.* to foobar IDENTIFIED BY 'raboof';
ERROR HY000: Your password does not satisfy the current policy requirements

SHOW WARNINGS;
+-----+-----+
| Level | Code | Message |
+-----+-----+
| Warning | 1819 | cracklib: it is based on your username |
| Error   | 1819 | Your password does not satisfy the current policy requirements |
+-----+-----+
```

Strict password validation:

```
GRANT SELECT ON *.* TO foo IDENTIFIED BY PASSWORD '2222222222222222';
ERROR HY000: The MariaDB server is running with the --strict-password-validation option so it cannot execute this statement
```

## Plugin API

Password validation plugin API is very simple. A plugin must implement only one method —

```
validate_password()
```

. This method takes two arguments — user name and the plain-text password. And it returns 0 when the password has passed the validation and 1 otherwise,

See also

```
mysql/plugin_password_validation.h  
and password validation plugins in  
plugin/simple_password_check/  
and  
plugins/cracklib_password_check/
```

### 4.4.6.5 password\_reuse\_check\_interval

The password\_reuse\_check\_interval system variable is available when the [password\\_reuse\\_check plugin](#) is installed. It determines the retention period for the password history in days. Zero, the default, means passwords are never discarded.

- **Commandline:**

```
--password_reuse_check_interval=#
```

- **Scope:** Global

- **Read-only:** No

- **Data Type:**

```
numeric
```

- **Default Value:**

```
0
```

- **Range:**

```
0
```

```
to
```

```
36500
```

## 4.4.7 Key Management and Encryption Plugin

### 4.4.7.1 Encryption Key Management

#### Contents

1. [Choosing an Encryption Key Management Solution](#)
  1. [File Key Management Plugin](#)
  2. [AWS Key Management Plugin](#)
  3. [Eperi Key Management Plugin](#)
2. [Using Multiple Encryption Keys](#)
3. [Key Rotation](#)
  1. [Support for Key Rotation in Encryption Plugins](#)
    1. [Encryption Plugins with Key Rotation Support](#)
    2. [Encryption Plugins without Key Rotation Support](#)
4. [Encryption Plugin API](#)

MariaDB's [data-at-rest encryption](#) requires the use of a key management and encryption plugin. These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of multiple encryption keys. Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports key rotation, then encryption keys can also be rotated, which creates a new version of the encryption key.

## Choosing an Encryption Key Management Solution

How MariaDB manages encryption keys depends on which encryption key management solution you choose. Currently, MariaDB has three options:

## File Key Management Plugin

The File Key Management plugin that ships with MariaDB is a basic key management and encryption plugin that reads keys from a plain-text file. It can also serve as example and as a starting point when developing a key management plugin.

For more information, see [File Key Management Plugin](#).

## AWS Key Management Plugin

The AWS Key Management plugin is a key management and encryption plugin that uses the Amazon Web Services (AWS) Key Management Service (KMS). The AWS Key Management plugin depends on the [AWS SDK for C++](#), which uses the [Apache License, Version 2.0](#). This license is not compatible with MariaDB Server's [GPL 2.0 license](#), so we are not able to distribute packages that contain the AWS Key Management plugin. Therefore, the only way to currently obtain the plugin is to install it from source.

For more information, see [AWS Key Management Plugin](#).

## Eperi Key Management Plugin

The Eperi Key Management plugin is a key management and encryption plugin that uses the [eperi Gateway for Databases](#). The [eperi Gateway for Databases](#) stores encryption keys on the key server outside of the database server itself, which provides an extra level of security. The [eperi Gateway for Databases](#) also supports performing all data encryption operations on the key server as well, but this is optional.

For more information, see [Eperi Key Management Plugin](#).

## Using Multiple Encryption Keys

Key management and encryption plugins support using multiple encryption keys. Each encryption key can be defined with a different 32-bit integer as a key identifier.

The support for multiple keys opens up some potential use cases. For example, let's say that a hypothetical key management and encryption plugin is configured to provide two encryption keys. One encryption key might be intended for "low security" tables. It could use short keys, which might not be rotated, and data could be encrypted with a fast encryption algorithm. Another encryption key might be intended for "high security" tables. It could use long keys, which are rotated often, and data could be encrypted with a slower, but more secure encryption algorithm. The user would specify the identifier of the key that they want to use for different tables, only using high level security where it's needed.

There are two encryption key identifiers that have special meanings in MariaDB. Encryption key

1

is intended for encrypting system data, such as InnoDB redo logs, binary logs, and so on. It must always exist when [data-at-rest encryption](#) is enabled. Encryption key

2

is intended for encrypting temporary data, such as temporary files and temporary tables. It is optional. If it doesn't exist, then MariaDB uses encryption key

1

for these purposes instead.

When [encrypting InnoDB tables](#), the key that is used to encrypt tables [can be changed](#).

When [encrypting Aria tables](#), the key that is used to encrypt tables [cannot currently be changed](#).

## Key Rotation

Encryption key rotation is optional in MariaDB Server. Key rotation is only supported if the backend key management service (KMS) supports key rotation, and if the corresponding key management and encryption plugin for MariaDB also supports key rotation. When a key management and encryption plugin supports key rotation, users can opt to rotate one or more encryption keys, which creates a new version of each rotated encryption key.

Key rotation allows users to improve data security in the following ways:

- If the server is configured to automatically re-encrypt table data with the newer version of the encryption key after the key is rotated, then that prevents an encryption key from being used for long periods of time.
- If the server is configured to simultaneously encrypt table data with multiple versions of the encryption key after the key is rotated, then that prevents all data from being leaked if a single encryption key version is compromised.

The [InnoDB storage engine](#) has [background encryption threads](#) that can [automatically re-encrypt pages when key rotations occur](#).

The [Aria storage engine](#) does [not currently have a similar mechanism to re-encrypt pages in the background when key rotations occur](#).

## Support for Key Rotation in Encryption Plugins

### Encryption Plugins with Key Rotation Support

- The [AWS Key Management Service \(KMS\)](#) supports encryption key rotation, and the corresponding [AWS Key Management Plugin](#) also supports encryption key rotation.

- The [eperi Gateway for Databases](#) supports encryption key rotation, and the corresponding [Eperi Key Management Plugin](#) also supports encryption key rotation.

## Encryption Plugins without Key Rotation Support

- The [File Key Management Plugin](#) does not support encryption key rotation, because it does not use a backend key management service (KMS).

# Encryption Plugin API

New key management and encryption plugins can be developed using the [encryption plugin API](#).

## 4.4.7.2 File Key Management Encryption Plugin

### Contents

1. [Overview](#)
2. [Installing the File Key Management Plugin's Package](#)
3. [Installing the Plugin](#)
4. [Uninstalling the Plugin](#)
5. [Creating the Key File](#)
  1. [Configuring the Path to an Unencrypted Key File](#)
6. [Encrypting the Key File](#)
  1. [Configuring the Path to an Encrypted Key File](#)
7. [Choosing an Encryption Algorithm](#)
  1. [Configuring the Encryption Algorithm](#)
8. [Using the File Key Management Plugin](#)
9. [Using Multiple Encryption Keys](#)
10. [Key Rotation](#)
11. [Versions](#)
12. [System Variables](#)
  1. `file_key_management_encryption_algorithm`
  2. `file_key_management_filekey`
  3. `file_key_management_filename`
13. [Options](#)
  1. `file_key_management`

MariaDB's [data-at-rest encryption](#) requires the use of a [key management and encryption plugin](#). These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of [multiple encryption keys](#). Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports [key rotation](#), then encryption keys can also be rotated, which creates a new version of the encryption key.

The File Key Management plugin that ships with MariaDB is a [key management and encryption plugin](#) that reads encryption keys from a plain-text file.

## Overview

The File Key Management plugin is the easiest [key management and encryption plugin](#) to set up for users who want to use [data-at-rest encryption](#). Some of the plugin's primary features are:

- It reads encryption keys from a plain-text key file.
- As an extra protection mechanism, the plain-text key file can be encrypted.
- It supports multiple encryption keys.
- It does **not** support key rotation.
- It supports two different algorithms for encrypting data.

It can also serve as an example and as a starting point when developing a key management and encryption plugin with the [encryption plugin API](#).

## Installing the File Key Management Plugin's Package

The File Key Management plugin is included in MariaDB packages as the

`file_key_management.so`

or

`file_key_management.dll`

shared library. The shared library is in the main server package, so no additional package installations are necessary.

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. The plugin can be

installed by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = file_key_management
```

## Uninstalling the Plugin

Before you uninstall the plugin, you should ensure that [data-at-rest encryption](#) is completely disabled, and that MariaDB no longer needs the plugin to decrypt tables or other files.

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'file_key_management';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Creating the Key File

In order to encrypt your tables with encryption keys using the File Key Management plugin, you first need to create the file that contains the encryption keys. The file needs to contain two pieces of information for each encryption key. First, each encryption key needs to be identified with a 32-bit integer as the key identifier. Second, the encryption key itself needs to be provided in hex-encoded form. These two pieces of information need to be separated by a semicolon. For example, the file is formatted in the following way:

```
<encryption_key_id1>;<hex-encoded_encryption_key1>
<encryption_key_id2>;<hex-encoded_encryption_key2>
```

You can also optionally encrypt the key file to make it less accessible from the file system. That is explained further in the section below.

The File Key Management plugin uses [Advanced Encryption Standard \(AES\)](#) to encrypt data, which supports 128-bit, 192-bit, and 256-bit encryption keys. Therefore, the plugin also supports 128-bit, 192-bit, and 256-bit encryption keys.

You can generate random encryption keys using the

```
openssl rand
```

command. For example, to create a random 256-bit (32-byte) encryption key, you would run the following command:

```
$ openssl rand -hex 32
a7add9adea9978fda19f21e6be987880e68ac92632ca052e5bb42b1a506939a
```

You can copy this encryption key to the key file using a text editor, or you can append a series of keys to a new key file. For example, to append three new encryption keys to a new key file, you could execute the following:

```
$ sudo openssl rand -hex 32 >> /etc/mysql/encryption/keyfile
$ sudo openssl rand -hex 32 >> /etc/mysql/encryption/keyfile
$ sudo openssl rand -hex 32 >> /etc/mysql/encryption/keyfile
```

The new key file would look something like the following after this step:

```
a7add9adea9978fda19f21e6be987880e68ac92632ca052e5bb42b1a506939a
49c16acc2dffe616710c9ba9a10b94944a737de1beccb52dc1560abfdd67388b
8db1ee74580e7e93ab8cf157f02656d356c2f437d548d5bf16bf2a56932954a3
```

The key file still needs to have a key identifier for each encryption key added to the beginning of each line. Key identifiers do not need to be contiguous. Open the new key file in your preferred text editor and add the key identifiers. For example, the key file would look something like the following after this step:

```
1;a7add9adea9978fda19f21e6be987880e68ac92632ca052e5bb42b1a506939a
2;49c16acc2dffe616710c9ba9a10b94944a737de1beccb52dc1560abfdd67388b
100;8db1ee74580e7e93ab8cf157f02656d356c2f437d548d5bf16bf2a56932954a3
```

The key identifiers give you a way to reference the encryption keys from MariaDB. In the example above, you could reference these encryption keys using the key identifiers

1  
,  
2  
or  
100  
with the

`ENCRYPTION_KEY_ID`

table option or with system variables such as

`innodb_default_encryption_key_id`

. You do not necessarily need multiple encryption keys--the encryption key with the key identifier  
1  
is the only mandatory encryption key.

## Configuring the Path to an Unencrypted Key File

If the key file is unencrypted, then the File Key Management plugin only requires the

`file_key_management_filename`

system variable to be configured.

This system variable can be specified as command-line arguments to

`mysqld`

or it can be specified in a relevant server `option group` in an `option file`. For example:

```
[mariadb]
...
loose_file_key_management_filename = /etc/mysql/encryption/keyfile
```

Note that the

`loose`

option prefix is specified. This option prefix is used in case the plugin hasn't been installed yet.

## Encrypting the Key File

By enabling the File Key Management plugin and setting the appropriate path on the

`file_key_management_filename`

system variable, you can begin using the plugin to manage your encryption keys. But, there is a security risk in doing so, given that the keys are stored in plain text on your system. You can reduce this exposure using file permissions, but it's better to encrypt the whole key file to further restrict access.

There are some important details to keep in mind about encrypting the key file, such as:

- The only algorithm that MariaDB currently supports to encrypt the key file is [Cipher Block Chaining \(CBC\)](#) mode of [Advanced Encryption Standard \(AES\)](#).
- The encryption key size can be 128-bits, 192-bits, or 256-bits.
- The encryption key is created from the [SHA-1](#) hash of the encryption password.
- The encryption password has a max length of 256 characters.

You can generate a random encryption password using the

```
openssl rand
```

command. For example, to create a random 256 character encryption password, you could execute the following:

```
$ sudo openssl rand -hex 128 > /etc/mysql/encryption/keyfile.key
```

You can encrypt the key file using the

```
openssl enc
```

command. For example, to encrypt the key file with the encryption password created in the previous step, you could execute the following:

```
$ sudo openssl enc -aes-256-cbc -md sha1 \  
-pass file:/etc/mysql/encryption/keyfile.key \  
-in /etc/mysql/encryption/keyfile \  
-out /etc/mysql/encryption/keyfile.enc
```

Running this command reads the unencrypted

```
keyfile  
file created above and creates a new encrypted  
keyfile.enc  
file, using the encryption password stored in  
keyfile.key  
. Once you've finished preparing your system, you can delete the unencrypted  
keyfile  
file, as it's no longer necessary.
```

## Configuring the Path to an Encrypted Key File

If the key file is encrypted, then the File Key Management plugin requires both the

```
file_key_management_filename
```

and the

```
file_key_management_filekey
```

system variables to be configured.

The

```
file_key_management_filekey
```

system variable can be provided in two forms:

- It can be the actual plain-text encryption password. This is not recommended, since the plain-text encryption password would be visible in the output of the

```
SHOW VARIABLES
```

statement.

- If it is prefixed with

```
FILE:
```

, then it can be a path to a file that contains the plain-text encryption password.

These system variables can be specified as command-line arguments to

```
mysqld
```

or they can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
loose_file_key_management_filename = /etc/mysql/encryption/keyfile.enc
loose_file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
```

Note that the

`loose`

option prefix is specified. This option prefix is used in case the plugin hasn't been installed yet.

## Choosing an Encryption Algorithm

The File Key Management plugin currently supports two encryption algorithms for encrypting data:

`AES_CBC`

and

`AES_CTR`

. Both of these algorithms use [Advanced Encryption Standard \(AES\)](#) in different modes. AES uses 128-bit blocks, and supports 128-bit, 192-bit, and 256-bit keys. The modes are:

- The

`AES_CBC`

mode uses AES in the [Cipher Block Chaining \(CBC\)](#) mode.

- The

`AES_CTR`

mode uses AES in two slightly different modes in different contexts. When encrypting tablespace pages (such as pages in InnoDB, XtraDB, and Aria tables), it uses AES in the [Counter \(CTR\)](#) mode. When encrypting temporary files (where the cipher text is allowed to be larger than the plain text), it uses AES in the authenticated [Galois/Counter Mode \(GCM\)](#).

The recommended algorithm is

`AES_CTR`

, but this algorithm is only available when MariaDB is built with recent versions of [OpenSSL](#). If the server is built with [wolfSSL](#) or [yaSSL](#), then this algorithm is not available. See [TLS and Cryptography Libraries Used by MariaDB](#) for more information about which libraries are used on which platforms.

## Configuring the Encryption Algorithm

The encryption algorithm can be configured by setting the

`file_key_management_encryption_algorithm`

system variable.

This system variable can be set to one of the following values:

System Variable	Description
<code>AES_CBC</code>	Data is encrypted using AES in the <a href="#">Cipher Block Chaining (CBC)</a> mode. This is the default value.
<code>AES_CTR</code>	Data is encrypted using AES either in the <a href="#">Counter (CTR)</a> mode or in the authenticated <a href="#">Galois/Counter Mode (GCM)</a> mode, depending on context. This is only supported in some builds. See the previous section for more information.

This system variable can be specified as command-line arguments to

`mysqld`

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
loose_file_key_management_encryption_algorithm = AES_CTR
```

Note that the

`loose`

option prefix is specified. This option prefix is used in case the plugin hasn't been installed yet.

Note that this variable does not affect the algorithm that MariaDB uses to decrypt the key file. This variable only affects the encryption algorithm that MariaDB uses to encrypt and decrypt data. The only algorithm that MariaDB currently supports to encrypt the key file is [Cipher Block Chaining \(CBC\)](#) mode of [Advanced Encryption Standard \(AES\)](#).

## Using the File Key Management Plugin

Once the File Key Management Plugin is enabled, you can use it by creating an encrypted table:

```
CREATE TABLE t (i int) ENGINE=InnoDB ENCRYPTED=YES
```

Now, table

`t`  
will be encrypted using the encryption key from the key file.

For more information on how to use encryption, see [Data at Rest Encryption](#).

## Using Multiple Encryption Keys

The File Key Management Plugin supports [using multiple encryption keys](#). Each encryption key can be defined with a different 32-bit integer as a key identifier.

When [encrypting InnoDB tables](#), the key that is used to encrypt tables [can be changed](#).

When [encrypting Aria tables](#), the key that is used to encrypt tables [cannot currently be changed](#).

## Key Rotation

The File Key Management plugin does not currently support [key rotation](#). See [MDEV-20713](#) for more information.

## Versions

Version	Status	Introduced
1.0	Stable	<a href="#">MariaDB 10.1.18</a>
1.0	Gamma	<a href="#">MariaDB 10.1.13</a>
1.0	Alpha	<a href="#">MariaDB 10.1.3</a>

## System Variables

### `file_key_management_encryption_algorithm`

- **Description:** This system variable is used to determine which algorithm the plugin will use to encrypt data.

- The recommended algorithm is

`AES_CTR`

, but this algorithm is only available when MariaDB is built with recent versions of [OpenSSL](#). If the server is built with [wolfSSL](#) or [yaSSL](#), then this algorithm is not available. See [TLS and Cryptography Libraries Used by MariaDB](#) for more information about which libraries are used on which platforms.

- **Commandline:**

`--file-key-management-encryption-algorithm=value`

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

enumerated

- **Default Value:**

`AES_CBC`

- **Valid Values:**

```
AES_CBC  
,  
AES_CTR
```

---

## file\_key\_management\_filekey

- **Description:** This system variable is used to determine the encryption password that is used to decrypt the key file defined by

```
file_key_management_filename
```

- If this system variable's value is prefixed with

FILE:

, then it is interpreted as a path to a file that contains the plain-text encryption password.

- If this system variable's value is **not** prefixed with

FILE:

, then it is interpreted as the plain-text encryption password. However, this is not recommended.

- The encryption password has a max length of 256 characters.

- The only algorithm that MariaDB currently supports when decrypting the key file is [Cipher Block Chaining \(CBC\)](#) mode of [Advanced Encryption Standard \(AES\)](#). The encryption key size can be 128-bits, 192-bits, or 256-bits. The encryption key is calculated by taking a [SHA-1](#) hash of the encryption password.

- **Commandline:**

```
--file-key-management-filekey=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

- **Default Value:** (empty)
- 

## file\_key\_management\_filename

- **Description:** This system variable is used to determine the path to the file that contains the encryption keys. If

```
file_key_management_filekey
```

is set, then this file can be encrypted with [Cipher Block Chaining \(CBC\)](#) mode of [Advanced Encryption Standard \(AES\)](#).

- **Commandline:**

```
--file-key-management-filename=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

- **Default Value:** (empty)
- 

# Options

## file\_key\_management

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

■

OFF  
- Disables the plugin without removing it from the  
`mysql.plugins`  
table.

ON  
- Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

FORCE  
- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

FORCE\_PLUS\_PERMANENT  
- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

`UNINSTALL SONAME`

or

`UNINSTALL PLUGIN`

while the server is running.

◦ See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

`--file-key-management=value`

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF  
,

ON  
,

FORCE  
,

FORCE\_PLUS\_PERMANENT

---

#### 4.4.7.3 AWS Key Management Encryption Plugin

## Contents

1. Overview
2. Tutorials
3. Preparation
4. Installing the Plugin's Package
  1. Installing from Source
5. Installing the Plugin
6. Uninstalling the Plugin
7. Configuring the AWS Key Management Plugin
8. Using the AWS Key Management Plugin
9. Using Multiple Encryption Keys
10. Key Rotation
11. Versions
12. System Variables
  1. aws\_key\_management\_key\_spec
  2. aws\_key\_management\_log\_level
  3. aws\_key\_management\_master\_key\_id
  4. aws\_key\_management\_mock
  5. aws\_key\_management\_region
  6. aws\_key\_management\_request\_timeout
  7. aws\_key\_management\_rotate\_key
13. Options
  1. aws\_key\_management

MariaDB's [data-at-rest encryption](#) requires the use of a [key management and encryption plugin](#). These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of [multiple encryption keys](#). Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports [key rotation](#), then encryption keys can also be rotated, which creates a new version of the encryption key.

The AWS Key Management plugin is a [key management and encryption plugin](#) that uses the [Amazon Web Services \(AWS\) Key Management Service \(KMS\)](#).

## Overview

The AWS Key Management plugin uses the [Amazon Web Services \(AWS\) Key Management Service \(KMS\)](#) to generate and store AES keys on disk, in encrypted form, using the Customer Master Key (CMK) kept in AWS KMS. When MariaDB Server starts, the plugin will decrypt the encrypted keys, using the AWS KMS "Decrypt" API function. MariaDB data will then be encrypted and decrypted using the AES key. It supports multiple encryption keys. It supports key rotation.

## Tutorials

Tutorials related to the AWS Key Management plugin can be found at the following pages:

- [Amazon Web Services \(AWS\) Key Management Service \(KMS\) Encryption Plugin Setup Guide](#)
- [Amazon Web Services \(AWS\) Key Management Service \(KMS\) Encryption Plugin Advanced Usage](#)

## Preparation

- Before you use the plugin, you need to create a Customer Master Key (CMK). Create a key using the AWS Console as described in the [AWS KMS developer guide](#).
- The easiest way to give the AWS key management plugin access to the key is to create an IAM Role with access to the key, and to apply that IAM Role to an EC2 instance where MariaDB Server runs.
- Make sure that MariaDB Server runs under the correct AWS identity that has access to the above key. For example, you can store the AWS credentials in a AWS credentials file for the user who runs

mysql

. More information about the credentials file can be found in the [AWS CLI Getting Started Guide](#).

## Installing the Plugin's Package

The AWS Key Management plugin depends on the [AWS SDK for C++](#), which uses the [Apache License, Version 2.0](#). This license is not compatible with MariaDB Server's [GPL 2.0 license](#), so we are not able to distribute packages that contain the AWS Key Management plugin. Therefore, the only way to currently obtain the plugin is to install it from source.

### Installing from Source

When [compiling MariaDB from source](#), the AWS Key Management plugin is not built by default in [MariaDB 10.1](#), but it is built by default in [MariaDB 10.2](#) and later, on systems that support it.

Compilation is controlled by the

```
-DPLUGIN_AWS_KEY_MANAGEMENT=DYNAMIC -DAWS_SDK_EXTERNAL_PROJECT=1
```

`cmake`

arguments.

The plugin uses [AWS C++ SDK](#), which introduces the following restrictions:

- The plugin can only be built on Windows, Linux and macOS.
- The plugin requires that one of the following compilers is used:

```
gcc  
4.8 or later,  
clang  
3.3 or later, Visual Studio 2013 or later.
```

- On Unix, the

```
libcurl  
development package (e.g.  
libcurl3-dev  
on Debian Jessie),  
uuid  
development package and  
openssl  
need to be installed.
```

- You may need to use a newer version of

`cmake`

than is provided by default in your OS.

## Installing the Plugin

Even after the package that contains the plugin's shared library is installed on the operating system, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'aws_key_management';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = aws_key_management
```

## Uninstalling the Plugin

Before you uninstall the plugin, you should ensure that [data-at-rest encryption](#) is completely disabled, and that MariaDB no longer needs the plugin to decrypt tables or other files.

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'aws_key_management';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#) , then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Configuring the AWS Key Management Plugin

To enable the AWS Key Management plugin, you also need to set the plugin's system variables. The

```
aws_key_management_master_key_id
```

system variable is the primary one to set. These system variables can be specified as command-line arguments to

```
mysqld
```

or they can be specified in a relevant server [option group](#) in an [option file](#) . For example:

```
[mariadb]
...
aws_key_management_master_key_id=alias/<your key's alias>
```

Once you've updated the configuration file, [restart](#) the MariaDB server to apply the changes and make the key management and encryption plugin available for use.

## Using the AWS Key Management Plugin

Once the AWS Key Management Plugin is enabled, you can use it by creating an encrypted table:

```
CREATE TABLE t (i int) ENGINE=InnoDB ENCRYPTED=YES
```

Now, table

t

will be encrypted using the encryption key generated by AWS.

For more information on how to use encryption, see [Data at Rest Encryption](#) .

## Using Multiple Encryption Keys

The AWS Key Management Plugin supports [using multiple encryption keys](#) . Each encryption key can be defined with a different 32-bit integer as a key identifier. If a previously unused identifier is used, then the plugin will automatically generate a new key.

When [encrypting InnoDB tables](#) , the key that is used to encrypt tables [can be changed](#) .

When [encrypting Aria tables](#) , the key that is used to encrypt tables [cannot currently be changed](#) .

## Key Rotation

The AWS Key Management plugin does support [key rotation](#) . To rotate a key, set the

```
aws_key_management_rotate_key
```

system variable. For example, to rotate key with ID 2:

```
SET GLOBAL aws_key_management_rotate_key=2;
```

Or to rotate all keys, set the value to -1:

```
SET GLOBAL aws_key_management_rotate_key=-1;
```

## Versions

Version	Status	Introduced
1.0	Stable	<a href="#">MariaDB 10.2.6</a> , <a href="#">MariaDB 10.1.24</a>
1.0	Beta	<a href="#">MariaDB 10.1.18</a>
1.0	Experimental	<a href="#">MariaDB 10.1.13</a>

## System Variables

### aws\_key\_management\_key\_spec

- **Description:** Encryption algorithm used to create new keys

- **Commandline:**

```
--aws-key-management-key-spec=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

enumerated

- **Default Value:**

AES\_128

- **Valid Values:**

AES\_128

,

AES\_256

---

### aws\_key\_management\_log\_level

- **Description:** Dump log of the AWS SDK to MariaDB error log. Permitted values, in increasing verbosity, are **Off** (default), **Fatal** , **Error** , **Warn** , **Info** , **Debug** , and **Trace** .

- **Commandline:**

```
--aws-key-management-log-level=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

enumerated

- **Default Value:**

Off

- **Valid Values:**

Off

,

Fatal

,

Warn

,

Info  
,  
Debug  
and  
Trace

---

## aws\_key\_management\_master\_key\_id

- **Description:** AWS KMS Customer Master Key ID (ARN or alias prefixed by alias/) for the master encryption key. Used to create new data keys. If not set, no new data keys will be created.
  - **Commandline:**  
  --aws-key-management-master-key-id=value
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
  string
  - **Default Value:**
- 

## aws\_key\_management\_mock

- **Description:** Mock AWS KMS calls (for testing). Must be enabled at compile-time.
  - **Commandline:**  
  --aws-key-management-mock
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
  boolean
  - **Default Value:**  
  OFF
  - **Valid Values:**  
  OFF  
  ,  
  ON
- 

## aws\_key\_management\_region

- **Description:** AWS region name, e.g us-east-1 . Default is SDK default, which is us-east-1.
  - **Commandline:**  
  --aws-key-management-region=value
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
  string
  - **Default Value:**  
  'us-east-1'
-

## aws\_key\_management\_request\_timeout

- **Description:** Timeout in milliseconds for create HTTPS connection or execute AWS request. Specify 0 to use SDK default.
  - **Commandline:**  
  --aws-key-management-request-timeout=value
  - **Scope:** Global
  - **Dynamic:** No
  - **Data Type:**  
  integer
  - **Default Value:** 0
- 

## aws\_key\_management\_rotate\_key

- **Description:** Set this variable to a data key ID to perform rotation of the key to the master key given in  
  aws\_key\_management\_master\_key\_id  
  . Specify -1 to rotate all keys.
  - **Commandline:**  
  --aws-key-management-rotate-key=value
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
  integer
  - **Default Value:**
- 

# Options

## aws\_key\_management

- **Description:** Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - OFF
      - Disables the plugin without removing it from the  
`mysql.plugins`  
table.
    - ON
      - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - FORCE
      - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - FORCE\_PLUS\_PERMANENT
      - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with  
`UNINSTALL SONAME`  
or  
`UNINSTALL PLUGIN`  
while the server is running.
- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**  
--aws-key-management=value

- **Data Type:**  
enumerated

- **Default Value:**  
ON

- **Valid Values:**  
OFF  
,  
ON  
,  
FORCE  
,  
FORCE\_PLUS\_PERMANENT

## 4.4.7.4 Amazon Web Services (AWS) Key Management Service (KMS) Encryption Plugin Setup Guide

### Contents

1. [Overview](#)
2. [Installing the Plugin's Package](#)
  1. [Installing from Source](#)
3. [Installing the Plugin](#)
4. [Sign up for Amazon Web Services](#)
5. [Create an IAM User and/or Role](#)
  1. [Creating an IAM Role](#)
  2. [Creating an IAM User](#)
6. [Create a master encryption key](#)
7. [Configure AWS credentials](#)
8. [Configure MariaDB](#)
  1. [SELinux and outbound connections from MariaDB](#)
9. [Start MariaDB](#)
10. [Create encrypted tables](#)
11. [AWS KMS plugin option reference](#)
12. [Next steps](#)

## Overview

MariaDB Server 10.1 introduces robust, full instance, at-rest encryption. This feature uses a flexible plugin interface to allow actual encryption to be done using a key management approach that meets the customer's needs. MariaDB Server, starting with [MariaDB 10.2](#), includes a plugin that uses the Amazon Web Services (AWS) Key Management Service (KMS) to facilitate separation of responsibilities and remote logging & auditing of key access requests.

Rather than storing the encryption key in a local file, this plugin keeps the master key in AWS KMS. When you first start MariaDB, the AWS KMS plugin will connect to the AWS Key Management Service and ask it to generate a new key. MariaDB will store that key on-disk in an encrypted form. The key stored on-disk cannot be used to decrypt the data; rather, on each startup, MariaDB connects to AWS KMS and has the service decrypt the locally-stored key(s). The decrypted key is stored in-memory as long as the MariaDB server process is running, and that in-memory decrypted key is used to encrypt the local data.

This guide is based on CentOS 7, using systemd with SELinux enabled. Some steps will differ if you use other operating systems or configurations.

## Installing the Plugin's Package

The AWS Key Management plugin depends on the [AWS SDK for C++](#), which uses the [Apache License, Version 2.0](#). This license is not compatible with MariaDB Server's [GPL 2.0 license](#), so we are not able to distribute packages that contain the AWS Key Management plugin. Therefore, the only way to currently obtain the plugin is to install it from source.

## Installing from Source

When [compiling MariaDB from source](#), the AWS Key Management plugin is not built by default in [MariaDB 10.1](#), but it is built by default in [MariaDB 10.2](#) and later, on systems that support it.

Compilation is controlled by the

```
-DPLUGIN_AWS_KEY_MANAGEMENT=DYNAMIC -DAWS_SDK_EXTERNAL_PROJECT=1
```

```
cmake
```

arguments.

The plugin uses [AWS C++ SDK](#), which introduces the following restrictions:

- The plugin can only be built on Windows, Linux and macOS.
- The plugin requires that one of the following compilers is used:

```
gcc  
4.8 or later,  
clang  
3.3 or later, Visual Studio 2013 or later.
```

- On Unix, the

```
libcurl  
development package (e.g.  
libcurl3-dev  
on Debian Jessie),  
uuid  
development package and  
openssl  
need to be installed.
```

- You may need to use a newer version of

```
cmake
```

than is provided by default in your OS.

## Installing the Plugin

Even after the package that contains the plugin's shared library is installed on the operating system, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'aws_key_management';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqlld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]  
...  
plugin_load_add = aws_key_management
```

## Sign up for Amazon Web Services

If you already have an AWS account, you can skip this section.

1. Load <http://aws.amazon.com/>.
2. Click "Create a Free Account" and complete the steps.
3. You'll need to enter credit card information. Charges related only to your use of the AWS KMS service should be limited to about \$1/month for the

single master key we will create. If you use other services, additional charges may apply. Consult AWS Cloud Pricing Principles <https://aws.amazon.com/pricing/> for more information about pricing of AWS services.

4. You'll need to complete the AWS identity verification process.

## Create an IAM User and/or Role

After creating an account or logging in to an existing account, follow these steps to create an IAM User or Role with restricted privileges that will use (but not administer) your master encryption key.

If you intend to run MariaDB Server on an EC2 instance, you should create a Role (or modify an existing Role already attached to your instance). If you intent to run MariaDB Server outside of AWS, you may want to create a User.

### Creating an IAM Role

1. Load the Identity and Access Management Console at <https://console.aws.amazon.com/iam/> .
2. Click "Roles" in the left-hand sidebar
3. Click "Create new role"
4. Select "AWS Service Role"
5. Click the "Select" button next to "Amazon EC2 / Allows EC2 instances to call AWS services on your behalf."
6. Do not select any policies on the "Attach Policy" screen. Click "Next Step"
7. Click "Next Step"
8. Give your Role a "Role name"
9. Click "Create role"

### Creating an IAM User

1. Load the Identity and Access Management Console at <https://console.aws.amazon.com/iam/> .
2. Click "Users" in the left-hand sidebar.

The screenshot shows the AWS IAM service interface. The top navigation bar includes links for AWS Services and Edit. Below it, there's a search bar and a 'Create New Users' button. The main area has tabs for Details, Groups, and Users (which is selected), Roles, Policies, Identity Providers, Account Settings, and Credential Report. A 'Search IAM' button is also present. On the right, there's a 'User Actions' dropdown with options like 'Create New User', 'Edit User', 'Delete User', and 'View User'. A 'Filter' input field is available. The results table has columns for 'User Name' and 'Groups'. A message at the bottom states 'No records found.'

3. Click the "Create New Users" button
4. Enter a single User Name of your choosing. We'll use "MDBEnc" for this demonstration. Keep the "Generate an access key for each user" box checked.

The screenshot shows the 'Create User' form. The left sidebar has 'Create User' and 'Encryption Keys' options. The main form has a title 'Enter User Names:' followed by a list input field containing '1. MDBEnc'. Below it are four empty input fields labeled 2., 3., 4., and 5. A note says 'Maximum 64 characters each'. At the bottom is a checked checkbox for 'Generate an access key for each user'.

5. Click "Create".
6. Click "Show User Security Credentials".

The screenshot shows a confirmation message: 'Your 1 User(s) have been created successfully. This is the last time these User security credentials will be available for download. You can manage and recreate these credentials any time.' Below this is a link 'Show User Security Credentials'.

7. Copy the Access Key ID and Secret Access Key. Optionally, you can click "Download Credentials". We will need these in order for local programs to interact with AWS using its API.

Your 1 User(s) have been created successfully.  
This is the last time these User security credentials will be available for download.  
You can manage and recreate these credentials any time.  
▼ Hide User Security Credentials

	MDBEnc
Access Key ID:	AKIAIG6IZ6TKF52FVV5A
Secret Access Key:	o7CEf7KhZfsVF9cS0a2roqqZNmuzXtIR869zpSBT

8. Create a file on your computer to hold the credentials for this user. We'll use this file later. It should have this structure:

```
[default]
aws_access_key_id = AKIAIG6IZ6TKF52FVV5A
aws_secret_access_key = o7CEf7KhZfsVF9cS0a2roqqZNmuzXtIR869zpSBT
```

9. Click "Close". If prompted because you did not Download Credentials, ensure that you've saved them somewhere, and click "Close".

## Create a master encryption key

Now, we'll create a master encryption key. This key can *never* be retrieved by *any* application or user. This key is used remotely to encrypt (and decrypt) the actual encryption keys that will be used by MariaDB. If this key is deleted or you lose access to it, you will be unable to use the contents of your MariaDB data directory.

1. Click "Encryption Keys" in the left-hand sidebar.
2. Click the "Get Started Now" button.
3. Use the "Filter" dropdown to choose the region where you'd like to create your master key.
4. Click the "Create Key" button.

Dashboard

Search IAM

Details

Groups

Users

Roles

Policies

Identity Providers

Account Settings

Credential Report

Encryption Keys

Create Key

Key Actions

Filter: US East (N. Virginia)

No records found.

5. Enter an Alias and Description of your choosing.
6. Click "Next Step".
7. Do **not** check the box to make your IAM Role or IAM User user a Key Administrator.
8. Click "Next Step" again.
9. Check the boxes to give your IAM Role and/or IAM User permissions to use this key.

Create Key in US East (N. Virginia)

Step 1: Create Alias and Description

Step 2: Define Key Administrative Permissions

Step 3: Define Key Usage Permissions

Step 4: Preview Key Policy

Define Key Usage Permissions

This Account

Choose the IAM users and roles that can use this key to encrypt and decrypt data

Filter	Name	Path
	<input checked="" type="checkbox"/> MDBEnc	/

10. Click "Next Step".
11. Click "Finish".

You should now see your key listed in the console:

The screenshot shows the AWS IAM console with the URL <https://console.aws.amazon.com/iam/home#encryptionKeys/us-west-2>. A success message at the top says 'Your master key was created successfully. Alias: mariadb-encryption'. Below it, a table lists the key: 'Alias' is 'mariadb-encryption', 'Key ID' is '9da6b21e-e50f-43d4-95d5-a89947c2f7..', and 'Status' is 'Enabled'. The table has columns for Alias, Key ID, and Status.

You'll use the "Alias" you provided when you configure MariaDB later.

We now have a Customer Master Key and an IAM user that has privileges to access it using access credentials. This is enough to begin using the AWS KMS plugin.

## Configure AWS credentials

There are a number of ways to give the IAM credentials to the AWS KMS plugin. The plugin supports reading credentials from all standard locations used across the various AWS API clients.

The easiest approach is to run MariaDB Server in an EC2 instance that has an IAM Role with User access to the CMK you wish to use. You can give key access privileges to a Role already attached to your EC2 instance, or you can create a new IAM Role and attach it to an already-running EC2 instance. If you've done that, no further credentials management is required and you do not need to create a

```
credentials
file.
```

If you're not running MariaDB Server on an EC2 instance, you can also place the credentials in the MariaDB data directory. The AWS API client looks for a

```
credentials
file in the
.aws
subdirectory of the home directory of the user running the client process. In the case of MariaDB, its home directory is its
datadir
```

1. Create a

```
credentials
file that MariaDB can read. Use the region you selected when creating the key. Master keys cannot be used across regions. For example:
```

```
$ cat /var/lib/mysql/.aws/credentials
[default]
aws_access_key_id = AKIAIG6IZ6TKF52FVV5A
aws_secret_access_key = o7CEF7KhZfsVF9cS0a2roqqZNmuzXtIR869zpSBT
region = us-east-1
```

2. Change the permissions of the file so that it is owned by, and can only be read by, the

```
mysql
user:
```

```
chown mysql /var/lib/mysql/.aws/credentials
chmod 600 /var/lib/mysql/.aws/credentials
```

## Configure MariaDB

1. Create a new option file to tell MariaDB to enable encryption functionality and to use the AWS KMS plugin. Create a new file under

```
/etc/my.cnf.d/
(or wherever your OS may have you create such files) with contents like this:
```

```
[mariadb]
plugin_load_add = aws_key_management
aws-key-management = FORCE_PLUS_PERMANENT
aws-key-management-master-key-id = alias/mariadb-encryption
aws-key-management-region = us-east-1
!include /etc/my.cnf.d/enable_encryption.preset
```

1. Append the "Alias" value you copied above to

```
alias/
to use as the value for the
aws-key-management-master-key-id
option.
```

Note that you **must** include

```
aws-key-management-region
```

in your .cnf file if you are not using the us-east-1 region.

Now, you have told MariaDB to use the AWS KMS plugin and you've put credentials for the plugin in a location where the plugin will find them. The /etc/my.cnf.d/enable\_encryption.preset file contains a set of options that enable all available encryption functionality.

When you start MariaDB, the AWS KMS plugin will connect to the AWS Key Management Service and ask it to generate a new key. MariaDB will store that key on-disk in an encrypted form. The key stored on-disk cannot be used to decrypt the data; rather, on each startup, MariaDB must connect to AWS KMS and have the service decrypt the locally-stored key. The decrypted version is stored in-memory as long as the MariaDB server process is running, and that in-memory decrypted key is used to encrypt the local data.

## SELinux and outbound connections from MariaDB

Because MariaDB needs to connect to the AWS KMS service, you must ensure that the host has outbound network connectivity over port 443 to AWS and you must ensure that local policies allow the MariaDB server process to make those outbound connections. By default, SELinux restricts MariaDB from making such connections.

The most simple way to cause SELinux to allow outbound HTTPS connections from MariaDB is to enable to mysql\_connect\_any boolean, like this:

```
setsebool -P mysql_connect_any 1
```

There are more complex alternatives that have a more granular effect, but those are beyond the scope of this document.

## Start MariaDB

Start MariaDB using the

systemctl  
tool:

```
systemctl start mariadb
```

If you do not use systemd, you may have to start MariaDB using some other mechanism.

You should see journal output similar to this:

```
# journalctl --no-pager -o cat -u mariadb.service

[Note] /usr/sbin/mysqld (mysqld 10.1.9-MariaDB-enterprise-log) starting as process 19831 ...
[Note] AWS KMS plugin: generated encrypted datakey for key id=1, version=1
[Note] AWS KMS plugin: loaded key 1, version 1, key length 128 bit
[Note] InnoDB: Using mutexes to ref count buffer pool pages
[Note] InnoDB: The InnoDB memory heap is disabled
[Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
[Note] InnoDB: Memory barrier is not used
[Note] InnoDB: Compressed tables use zlib 1.2.7
[Note] InnoDB: Using CPU crc32 instructions
[Note] InnoDB: Initializing buffer pool, size = 2.0G
[Note] InnoDB: Completed initialization of buffer pool
[Note] InnoDB: Highest supported file format is Barracuda.
[Note] InnoDB: 128 rollback segment(s) are active.
[Note] InnoDB: Waiting for purge to start
[Note] InnoDB: Percona XtraDB (http://www.percona.com) 5.6.26-74.0 started; log sequence number 1616819
[Note] InnoDB: Dumping buffer pool(s) not yet started
[Note] Plugin 'FEEDBACK' is disabled.
[Note] AWS KMS plugin: generated encrypted datakey for key id=2, version=1
[Note] AWS KMS plugin: loaded key 2, version 1, key length 128 bit
[Note] Using encryption key id 2 for temporary files
[Note] Server socket created on IP: '::'.
[Note] Reading of all Master_info entries succeeded
[Note] Added new Master_info '' to hash table
[Note] /usr/sbin/mysqld: ready for connections.
```

Note the several lines of output that refer explicitly to the "AWS KMS plugin". You can see that the plugin generates a "datakey", loads that data key, and then later generates and loads a second data key. The 2nd data key is used to encrypt temporary files and temporary tables.

You can see the encrypted keys stored on-disk in the datadir:

```
# ls -l /var/lib/mysql/aws*
-rw-rw----. 1 mysql mysql 188 Feb 25 18:55 /var/lib/mysql/aws-kms-key.1.1
-rw-rw----. 1 mysql mysql 188 Feb 25 18:55 /var/lib/mysql/aws-kms-key.2.1
```

Note that those keys are not useful alone. They are encrypted. When MariaDB starts up, the AWS KMS plugin decrypts those keys by interacting with AWS KMS.

For maximum security, you should start from an empty datadir and run

```
mysql_install_db  
after configuring encryption. Then you should re-import your data so that it is fully encrypted. Use  
sudo  
to run  
mysql_install_db  
so that it finds your credentials file:
```

```
# sudo -u mysql mysql_install_db  
Installing MariaDB/MySQL system tables in '/var/lib/mysql' ...  
2016-02-25 23:16:06 139731553998976 [Note] /usr/sbin/mysqld (mysqld 10.1.11-MariaDB-enterprise-log) starting as process 39551 ...  
2016-02-25 23:16:07 139731553998976 [Note] AWS KMS plugin: generated encrypted datakey for key id=1, version=1  
2016-02-25 23:16:07 139731553998976 [Note] AWS KMS plugin: loaded key 1, version 1, key length 128 bit  
...
```

## Create encrypted tables

With

```
innodb-encrypt-tables=ON  
, new InnoDB tables will be encrypted by default, using the key ID set in  
innodb_default_encryption_key_id  
(default 1). With  
innodb-encrypt-tables=FORCE  
enabled, it is not possible to manually bypass encryption when creating a table.
```

You can cause the AWS KMS plugin to create new encryption keys at-will by specifying a new ENCRYPTION\_KEY\_ID when creating a table:

```
MariaDB [test]> create table t1 (id serial, v varchar(32)) ENCRYPTION_KEY_ID=3;  
Query OK, 0 rows affected (0.91 sec)
```

```
[Note] AWS KMS plugin: generated encrypted datakey for key id=3, version=1  
[Note] AWS KMS plugin: loaded key 3, version 1, key length 128 bit
```

```
# ls -l /var/lib/mysql/aws*  
-rw-rw---- 1 mysql mysql 188 Feb 25 18:55 /var/lib/mysql/aws-kms-key.1.1  
-rw-rw---- 1 mysql mysql 188 Feb 25 18:55 /var/lib/mysql/aws-kms-key.2.1  
-rw-rw---- 1 mysql mysql 188 Feb 25 19:10 /var/lib/mysql/aws-kms-key.3.1
```

Read more about encrypting data in the [Data at Rest Encryption](#) section of the MariaDB Documentation.

## AWS KMS plugin option reference

- `aws_key_management_master_key_id`  
: AWS KMS Customer Master Key ID (ARN or alias prefixed by  
alias/  
) for master encryption key. Used to create new data keys. If not set, no new data keys will be created.
- `aws_key_management_rotate_key`  
: Set this variable to a data key ID to perform rotation of the key to the master key given in  
`aws_key_management_master_key_id`  
. Specify -1 to rotate all keys.
- `aws_key_management_key_spec`  
: Encryption algorithm used to create new keys. Allowed values are AES\_128 (default) or AES\_256.
- `aws_key_management_log_level`  
: Logging for AWS API. Allowed values, in increasing verbosity, are "Off" (default), "Fatal", "Error", "Warn", "Info", "Debug", and "Trace".

## Next steps

For more information about advanced usage, including strategies to manage credentials, enforce separation of responsibilities, and even require 2-factor authentication to start your MariaDB server, please review [Amazon Web Services \(AWS\) Key Management Service \(KMS\) Encryption Plugin Advanced Usage](#).

## 4.4.7.5 Amazon Web Services (AWS) Key Management

# Service (KMS) Encryption Plugin Advanced Usage

## Contents

- 1. [Managing AWS credentials](#)
- 2. [AWS KMS Key Policy](#)
  - 1. Source IP restrictions
  - 2. [Using a Multi-Factor Authentication \(MFA\) device](#)
    - 1. [Wrapper program example](#)
    - 3. [Disabling keys when not needed](#)
      - 1. [Adding MFA](#)
  - 3. [Logging and auditing](#)
    - 1. [CloudTrail](#)
    - 2. [CloudWatch](#)

This document assumes you've already set up an Amazon Web Services (AWS) account, created a master key in the Key Management Service (KMS), and have done the basic work to set up the MariaDB AWS KMS plugin. These steps are all described in [Amazon Web Services \(AWS\) Key Management Service \(KMS\) Encryption Plugin Setup Guide](#).

Ultimately, keeping all the credentials required to read the key on a single host means that a user who has gained access to the host has enough information to read the encrypted files in the datadir, read the encrypted keys from the datadir, interact with AWS KMS to decrypt the encrypted keys, and then used the decrypted keys to decrypt the encrypted data.

Theoretically, a superuser can read the memory of the MariaDB server process to read the decrypted keys or restart MariaDB with password authentication disabled in order to dump data, or add new users to MariaDB in order to allow a user to connect and dump the data. Resolving these issues is beyond the scope of this document. A user who gains root access to your operating system or root access to your MariaDB server will have the ability to decrypt your data. Plan accordingly.

## Managing AWS credentials

Putting the AWS credentials in a file inside the MariaDB home directory is not ideal. By default, any user with the FILE privilege can read any files that the MariaDB server has permission to read, which would include the credentials file. To protect against this, you should set

```
secure_file_priv  
to restrict the location the server will allow a user to read from when executing  
LOAD DATA INFILE  
or the  
LOAD_FILE()  
function.
```

But putting them in other locations requires passing additional data to the server, which in the case of CentOS 7 requires customizing the systemd startup procedure. This is most easily done by creating a "drop-in" file in /etc/systemd/system/mariadb.service.d/. The file should end in ".conf" but can otherwise be named whatever you like. After making any changes to systemd files, execute

```
systemctl daemon-reload  
and then start (or restart) the service as usual.
```

You can place the credentials file in a location of your choosing and then refer to that file by setting the AWS\_CREDENTIAL\_PROFILE\_FILE environment variable in the drop-in file:

```
[Service]  
Environment=AWS_CREDENTIAL_PROFILE_FILE=/etc/aws-kms-credentials
```

The credentials file will need to be readable by the "mysql" user, but it does not need to be readable by any other user.

AWS credentials can also be put directly into a "drop-in" systemd file that will be read when starting the MariaDB service:

```
# cat /etc/systemd/system/mariadb.service.d/aws-kms.conf  
[Service]  
Environment=AWS_ACCESS_KEY_ID=AKIAIRSG2XYZATCJLZ4A  
Environment=AWS_SECRET_ACCESS_KEY=ux91LZIxCp4ZXabcdefgIViQNtTan42QAmJqJVqV
```

However, any OS user can read this information from systemd, which could be considered a security risk. Another solution is to put the credentials in a separate file that is only readable by root and then refer to that file using an

```
EnvironmentFile  
directive in a drop-in systemd file.
```

```
# cat /etc/systemd/system/mariadb.service.d/aws-kms.env
AWS_ACCESS_KEY_ID=AKIAIRSG2XYZATCJLZ4A
AWS_SECRET_ACCESS_KEY=ux91LZIxCp4ZXabcdefgIViQNTan42QAmJqJVqV

# chown root /etc/systemd/system/mariadb.service.d/aws-kms.env
# chmod 600 /etc/systemd/system/mariadb.service.d/aws-kms.env

# cat /etc/systemd/system/mariadb.service.d/aws-kms.conf
[Service]
EnvironmentFile=/etc/systemd/system/mariadb.service.d/aws-kms.env
```

That has the advantage the the credentials can only be read directly by root. systemd adds those variables to the environment of the MariaDB server when starting it, and MariaDB can use the credentials to interact with AWS. Note, though, that any process running as the "mysql" user can still read the credentials from the proc filesystem on Linux.

```
$ whoami
mysql
$ cat /proc/$(pgrep mysqld)/environ | tr '\0' '\n' | grep AWS
AWS_ACCESS_KEY_ID=AKIAIRSG2XYZATCJLZ4A
AWS_SECRET_ACCESS_KEY=ux91LZIxCp4ZXabcdefgIViQNTan42QAmJqJVqV
```

## AWS KMS Key Policy

AWS KMS allows flexible, user-editable key policy. This offers fine-grained control over which users can operate on keys. The possibilities range from simply restricting which IP addresses are allowed to perform operations on the key, to requiring a MFA (Multi-Factor Authentication) device to use the key, to enforcing separation of responsibilities by creating an additional user with limited privileges to enable and disable the key. All 3 of these options will be outlined in this section.

For more details about customizing the Key Policy for your master keys, please consult the [AWS Key Management Service Key Policy](#) documentation.

### Source IP restrictions

A simple, common-sense restriction to put in place is to restrict the range of IP addresses that are allowed to use your master key. This way, even if someone obtains API credentials, they'll be unable to use them to decrypt your encryption keys from a different host.

To restrict API access from only a specific IP address or range of IP addresses, you'll need to manually edit the key policy.

1. Load the IAM console at <https://console.aws.amazon.com/iam/>.
2. Click "Encryption Keys" in the left-hand sidebar.
3. Click the name of your encryption key to view its details.
4. Click the link labeled "Switch to policy view", to the right of the heading of the "Key Policy" section.
5. Locate the section that contains

```
"Sid": "Allow use of the key"
```

6. Add this text below the

```
"Sid"
line:
```

```
"Condition": {
    "IpAddress": {
        "aws:SourceIp": [
            "10.1.2.3/32"
        ]
    }
},
```

... replacing

```
10.1.2.3/32
```

above with an IP address or range of IP addresses in CIDR format. For example, a single address would be

```
192.168.12.34/32
```

, while a range of addresses might be

```
192.168.0.0/24
```

7. Click "Save Changes".
8. Click "Proceed" if prompted with a warning about using the default view in the future.

Access to the API will now be restricted to requests coming from the IP address or range of IP addresses specified in the policy.

## Using a Multi-Factor Authentication (MFA) device

One approach is to modify the key policy for the master key so that MFA (Multi-Factor Authentication) is required in order to use the key. This is achieved with a wrapper that handles prompting the user for an MFA token, acquires temporary, limited-privilege credentials from the AWS Security Token Service

(STS), and puts those credentials into the environment of the MariaDB server process. The credentials can expire after as little as 15 minutes.

To require an MFA token for users of the key, you'll need to manually edit the key policy.

1. Load the IAM console at <https://console.aws.amazon.com/iam/>.
2. Click "Encryption Keys" in the left-hand sidebar.
3. Click the name of your encryption key to view its details.
4. Click the link labeled "Switch to policy view", to the right of the heading of the "Key Policy" section.
5. Locate the section that contains  

```
"Sid": "Allow use of the key"
```

6. Add this text below the

```
"Sid"  
line:
```

```
"Condition": {  
    "Bool": {  
        "aws:MultiFactorAuthPresent": "True"  
    }  
},
```

7. Click "Save Changes".
8. Click "Proceed" if prompted with a warning about using the default view in the future.

Now, add an MFA device for your user. You'll need to have a hardware MFA device or an application such as Google Authenticator installed on your smartphone.

1. Click "Users" in the left-hand sidebar.
2. Click the name of your user.
3. Click the "Security Credentials" tab.
4. In the "Sign-In Credentials" section, click the "Manage MFA Device" button.
5. Complete the steps to activate your MFA device.
6. Copy the ARN for your MFA device. You will need to use this when configuring the wrapper program.

Now, set up the wrapper program.

1. Copy the iam-kms-wrapper file to /usr/local/bin/, and ensure that it is executable.
2. Create a drop-in systemd config file in

```
/etc/systemd/system/mariadb.service.d/  
:  
[Service]  
EnvironmentFile=/etc/systemd/system/mariadb.service.d/aws-kms.env  
ExecStart=  
ExecStart=/usr/local/bin/iam-kms-wrapper --config=/etc/my.cnf.d/iam-kms-wrapper.config /usr/sbin/mysqld $MYSQLD_OPTS
```

3. Execute

```
systemctl daemon-reload
```

4. Create a file at

```
/etc/my.cnf.d/iam-kms-wrapper.config  
with these contents, using the ARN for your MFA device as the value for  
kms_mfa_id  
:
```

```
[kms]  
kms_mfa_id = arn:aws:iam::551888187628:mfa/MDBEnc  
kms_mfa_socket = /tmp/kms_mfa_socket
```

When you start the MariaDB service now, the wrapper will temporarily create a socket file at the location given by the

```
kms_mfa_socket
```

option. The wrapper will read the MFA code from the socket and will use it to authenticate to KMS. To give the MFA code, simply write the digits to the socket file using

```
echo  
:  
echo 111676 > /tmp/kms_mfa_socket
```

The

```
systemctl  
command will block until MariaDB starts, so you will need to write the code to the socket file via a separate terminal.
```

Note that the temporary credentials put into the environment of the MariaDB process will expire after a period of time defined by the request to the AWS Security Token Service (STS). In the example below, they will expire after 900 seconds. After that time, MariaDB may be unable to generate new encrypted data keys, which means that, for example, an attempt to create a table with a previously-unused key ID would fail.

## Wrapper program example

Here's an example wrapper program written in go. Build this into an executable named

iam-kms-wrapper

and use it as instructed above. This could of course be written in any language for which an appropriate AWS SDK exists, but go has the benefit of compiling to a static binary, which means you do not have to worry about interpreter versions or installing complex dependencies on the host that runs your MariaDB server.

```
package main

import (
    "syscall"
    "os"
    "log"
    "flag"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/aws/awser"
    "github.com/aws/aws-sdk-go/service/sts"
    "github.com/robfig/config"
)

func main() {

    config_file_p := flag.String("config", "", "location of the config file")
    flag.Parse()

    if flag.NArg() < 1 {
        log.Fatal("Command to wrap must be given as first command-line argument")
    }

    cmd := flag.Arg(0)
    args := flag.Args()[0:]

    conf, err := config.ReadDefault(*config_file_p)
    if err != nil {
        log.Fatal(err)
    }

    kms_mfa_id, err := conf.String("kms", "kms_mfa_id")
    mfa_socket_file, err := conf.String("kms", "kms_mfa_socket")

    sess := session.New()
    svc := sts.New(sess)

    syscall.Umask(0044)

    log.Printf("Reading MFA token from %s\n", mfa_socket_file)

    if err := syscall.Mknod(mfa_socket_file, syscall.S_IFIFO|uint32(os.FileMode(0622)), 0); err != nil {
        log.Fatal(err)
    }

    file, err := os.Open(mfa_socket_file)
    if err != nil {
        log.Fatal(err)
    }

    token := make([]byte, 6)

    if _, err := file.Read(token); err != nil {
        log.Fatal(err)
    }

    file.Close()

    if err := syscall.Unlink(mfa_socket_file); err != nil {
        log.Fatal(err)
    }

    mfa_token := string(token)

    token_params := &sts.GetSessionTokenInput{
        DurationSeconds: aws.Int64(900),
        SerialNumber:   aws.String(kms_mfa_id),
        TokenCode:      aws.String(mfa_token),
    }
```

```

resp, err := svc.GetSessionToken(token_params)
if err != nil {
    if awsErr, ok := err.(awserr.Error); ok {
        // Prints out full error message, including original error if there was one.
        log.Fatal("Error:", awsErr.Error())
    } else {
        log.Fatal("Error:", err.Error())
    }
}

creds := resp.Credentials

os.Setenv("AWS_ACCESS_KEY_ID", *creds.AccessKeyId)
os.Setenv("AWS_SECRET_ACCESS_KEY", *creds.SecretAccessKey)
os.Setenv("AWS_SESSION_TOKEN", *creds.SessionToken)

execErr := syscall.Exec(cmd, args, os.Environ())
if execErr != nil {
    panic(execErr)
}
}

```

## Disabling keys when not needed

Another possibility is to use the API to disable access to the master key and enable it only when a trusted administrator knows that the MariaDB service needs to be started. A specialized tool on a separate host could be used to enable the key for a very short period of time while the service starts and then quickly disable the key.

To do this, you can create an extra IAM User that can only use the kms:EnableKey and kms:DisableKey API endpoints for your key. This user will not be able to encrypt or decrypt any data using the key.

First, create a new user.

1. Load the IAM console at <https://console.aws.amazon.com/iam/>.
2. Click "Users" in the left-hand sidebar.
3. Click "Create New Users".
4. Enter a new user name. (The examples will use "MDBEncAdmin".)
5. Click "Show User Security Credentials".
6. Copy the credentials and put them in a

credentials  
file with this structure:

```
[MDBEncAdmin]
aws_access_key_id=AKIAJMPN07EBKABCDEF
aws_secret_access_key=pVdGwbuK5/jG64aBK1oEJOXR1kdM0aAylgabCDef
```

7. Click "Close".
8. Click "Close" again if prompted.
9. Click the name of your new user to open the details view.
10. Copy the "User ARN" value for your user (for example "arn:aws:iam::551888181234:user/MDBEncAdmin"). You will need this for the next step.

Now, give the new user permission to perform API operations on your key.

1. Click "Encryption Keys" in the left-hand sidebar.
2. Click the name of your key to open the details view.
3. Click "Switch to policy view" if it is not already open. (The "policy view" is a large text field that contains JSON describing the key policy.)
4. Create a new item in the

Statement  
array with this structure:

```
{
    "Sid": "Allow Enable and Disable of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::551888181234:user/MDBEncAdmin"
    },
    "Action": [
        "kms:EnableKey",
        "kms:DisableKey"
    ]
},
```

...so that your Key Policy looks like this:

```
{  
    "Version": "2012-10-17",  
    "Id": "key-consolepolicy-2",  
    "Statement": [  
        {  
            "Sid": "Allow Enable and Disable of the key",  
            "Effect": "Allow",  
            "Principal": {  
                ...  
            }  
        }  
    ]  
}
```

#### 5. Click "Save Changes".

You've now added a new IAM user and you've given that user privileges to enable and disable your key. This user will be able to perform those operations using the AWS CLI or via a script of your own design using the AWS API. For example, using the [AWS CLI](#):

```
$ cat ~/.aws/credentials  
[MDBEncAdmin]  
aws_access_key_id=AKIAJMPN07EBKABCDEF  
aws_secret_access_key=pVdGwbuK5/jG64aBK1oEJOXR1kdM0aAylgabCDef  
  
$ AWS_PROFILE=MDBEncAdmin aws --region us-east-1 kms disable-key --key-id arn:aws:kms:us-east-1:551888181234:key/abcd8f6-084b-4  
...
```

In order for MariaDB to start, this new user will have to enable the master key, then the DBA can start MariaDB, and this user can once again disable the master key after the service has started. Note that in this workflow, MariaDB will be unable to create new encryption keys, such as would be done when a user creates a table that refers to a non-existent key ID. The AWS KMS plugin will encounter an error if it tries to generate a new encryption key while the master key is disabled. In that scenario, the key administrator would have to enable the key before the operation could succeed. Here's what you should expect to see in the journal if MariaDB tries to interact with a disabled master key:

```
[ERROR] AWS KMS plugin : GenerateDataKeyWithoutPlaintext failed : DisabledException - Unable to parse ExceptionName: DisabledExc  
...
```

## Adding MFA

It's also possible to add MFA to this technique so that the user that enables & disables the master key has to authenticate using an MFA device. Adapt the instructions in the MFA section above to add MFA to the policy section for the user with `EnableKey` and `DisableKeys` privileges, add an MFA device for that user, use Security Token Service (STS) to get temporary security credentials, and then use those credentials to make the API calls. Here's an example Python script that follows that workflow:

```

#!/usr/bin/env python

import boto3
import sys

# Command-Line argument processing should be more robust than this...
action= sys.argv[1]
mfa_token= sys.argv[2]

# These should perhaps go into a config file instead of here
mfa_serial= 'arn:aws:iam::551888181234:mfa/MDBEncAdmin'
key_id= 'arn:aws:kms:us-east-1:551888181234:key/abcd8f6-084b-4cff-99ca-abcdef6c7907c'

# Make the connection to the Security Token Service to get temporary credentials
token_client= boto3.client('sts')
token_response= token_client.get_session_token(
    DurationSeconds= 900,
    SerialNumber= mfa_serial,
    TokenCode= mfa_token
)
cred= token_response['Credentials']

# Start new session using temporary, MFA-authenticated credentials
kms_session= boto3.session.Session(
    aws_access_key_id= cred['AccessKeyId'],
    aws_secret_access_key= cred['SecretAccessKey'],
    aws_session_token= cred['SessionToken'],
    region_name= key_id.split(':')[3]
)

# Start KMS client and execute operation against key
kms_client= kms_session.client('kms')
if action == 'enable' or action == 'e':
    action_f= kms_client.enable_key
elif action == 'disable' or action == 'd':
    action_f= kms_client.disable_key
else:
    raise Exception('Action must be either "disable" or "enable"')

action_f(KeyId=key_id)

```

```

$ AWS_PROFILE=MDBEncAdmin python kms-manage-key disable 575290
$ AWS_PROFILE=MDBEncAdmin python kms-manage-key enable 799870

```

## Logging and auditing

### CloudTrail

Amazon's CloudTrail service creates JSON-formatted text log files for every API interaction. Enabling CloudTrail requires S3, which incurs additional fees.

First, enable CloudTrail and add a trail.

1. Load the CloudTrail console at <https://console.aws.amazon.com/cloudtrail/> .
2. If you've never used CloudTrail before, click "Get Started Now".
3. Enter a value for "trail name". This example uses "mariadb-encryption-key".
4. Create a new S3 bucket, using a globally unique name, or use an existing S3 bucket, according to your needs.
5. Click "Turn On".

If you navigate to the S3 bucket you created, you should find log files that contain JSON-formatted descriptions of your API interactions.

### CloudWatch

Amazon's CloudWatch service allows you to create alarms and event rules that monitor log information.

First, send your CloudTrail logs to CloudWatch.

1. Load the CloudTrail console at <https://console.aws.amazon.com/cloudtrail/> .
2. Click "Trails" in the left-hand navigation sidebar.
3. Click the name of your trail to open the Configuration view.
4. In the "CloudWatch Logs" section, click "Configure".
5. Click "Continue".
6. Click "Allow".

Now, set up an SNS topic to facilitate email notifications.

1. Open <https://console.aws.amazon.com/sns/>.
2. Make sure the region in the console (look in the upper-right corner) is the same as the region where you created your key!
3. Click "Get Started" is prompted.
4. Click "Events" in the left-hand sidebar.
5. Click "Create new topic".
6. Enter a *Topic name* of your choosing.
7. Enter a *Display name* of your choosing.
8. Click "Create topic".
9. Click the ARN of your new SNS topic.
10. Click "Create Subscription".
11. Select "Email" from the Protocol dropdown.
12. Enter the desired notification email address in the Endpoint field.
13. Wait for the confirmation email to show up and follow the instructions.

Now, configure CloudWatch and create an Event Rule.

1. Open <https://console.aws.amazon.com/cloudwatch/>.
2. Make sure the region in the console (look in the upper-right corner) is the same as the region where you created your key and your SNS topic!
3. Click "Events" in the left-hand sidebar.
4. Click "Create rule".
5. Choose "AWS API call" from the "Select event source" dropdown.
6. Choose "KMS" from the "Service name" dropdown.
7. Decide which operations should trigger the event. (You can keep "Any operation" selected for simplicity.)
8. Click "Add target".
9. Select "SNS target" from the dropdown.
10. Select the SNS topic you created in the previous steps.
11. Click "Configure details".
12. Enter a *Name* and *Description* of your choosing.
13. Click "Create rule".

You should now get emails any time someone executes API calls on the KMS service in the region where you've created the CloudWatch Event rule. That means you should get email any time the key is enabled or disabled, and any time the AWS KMS plugin generates new keys or decrypts the keys stored on disk on the MariaDB server.

You may also wish to create an event rule (or an additional event) that matches only when an unauthorized user tries to access the key. You might accomplish that by manually editing the Event selector of the rule to look something like this:

```
{  
  "detail-type": [  
    "AWS API Call via CloudTrail"  
>  ],  
  "detail": {  
    "eventSource": [  
      "kms.amazonaws.com"  
>  ],  
    "errorCode": [  
      "AccessDenied",  
      "UnauthorizedOperation"  
>  ]  
>  }  
> }
```

The emails are formatted as JSON. Further customization of the CloudWatch email workflow is beyond the scope of this document.

There are many other workflows available using CloudWatch, including workflows with alarms and dashboards. Those are beyond the scope of this document.

#### 4.4.7.6 Eperi Key Management Encryption Plugin

## Contents

1. Overview
2. Installing the Eperi Key Management Plugin's Package
3. Installing the Plugin
4. Uninstalling the Plugin
5. Configuring the Eperi Key Management Plugin
6. Using the Eperi Key Management Plugin
7. Using Multiple Encryption Keys
8. Key Rotation
9. Versions
10. System Variables
  1. `eperi_key_management_plugin_database`
  2. `eperi_key_management_plugin_encrypt`
  3. `eperi_key_management_plugin_encrypt`
  4. `eperi_key_management_plugin_osslm`
  5. `eperi_key_management_plugin_port`
  6. `eperi_key_management_plugin_url`
  7. `eperi_key_management_plugin_url_crea`
11. Options
  1. `eperi_key_management_plugin`
12. See Also

MariaDB's [data-at-rest encryption](#) requires the use of a [key management and encryption plugin](#). These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of [multiple encryption keys](#). Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports [key rotation](#), then encryption keys can also be rotated, which creates a new version of the encryption key.

The Eperi Key Management plugin is a [key management and encryption plugin](#) that integrates with [eperi Gateway for Databases](#).

## Overview

The Eperi Key Management plugin is one of the [key management and encryption plugins](#) that can be set up for users who want to use [data-at-rest encryption](#). Some of the plugin's primary features are:

- It reads encryption keys from [eperi Gateway for Databases](#).
- It supports multiple encryption keys.
- It supports key rotation.
- It supports two different algorithms for encrypting data.

The [eperi Gateway for Databases](#) stores encryption keys on the key server outside of the database server itself, which provides an extra level of security. The [eperi Gateway for Databases](#) also supports performing all data encryption operations on the key server as well, but this is optional.

It also provides the following benefits:

- Key management outside the database
- No keys on database server hard disk
- Graphical user interface for configuration
- Encryption and decryption outside the database, supporting HSM's for maximum security.

Support for MariaDB is provided in [eperi Gateway for Databases 3.4](#).

## Installing the Eperi Key Management Plugin's Package

For information on how to install the package, see Eperi's documentation at the [Eperi Customer Portal](#).

## Installing the Plugin

Even after the package that contains the plugin's shared library is installed on the operating system, the plugin is not actually installed by MariaDB by default. The plugin can be installed by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = eperi_key_management_plugin
```

## Uninstalling the Plugin

Before you uninstall the plugin, you should ensure that [data-at-rest encryption](#) is completely disabled, and that MariaDB no longer needs the plugin to decrypt tables or other files.

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'eperi_key_management_plugin';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Configuring the Eperi Key Management Plugin

For information on how to configure the plugin, see Eperi's documentation at the [Eperi Customer Portal](#).

## Using the Eperi Key Management Plugin

Once the Eperi Key Management Plugin is enabled, you can use it by creating an encrypted table:

```
CREATE TABLE t (i int) ENGINE=InnoDB ENCRYPTED=YES
```

Now, table

t

will be encrypted using the encryption key from the key server.

For more information on how to use encryption, see [Data at Rest Encryption](#).

## Using Multiple Encryption Keys

The Eperi Key Management Plugin supports [using multiple encryption keys](#). Each encryption key can be defined with a different 32-bit integer as a key identifier.

When [encrypting InnoDB tables](#), the key that is used to encrypt tables [can be changed](#).

When [encrypting Aria tables](#), the key that is used to encrypt tables [cannot currently be changed](#).

## Key Rotation

The Eperi Key Management plugin supports [key rotation](#).

## Versions

Version	Status	Introduced
---------	--------	------------

## System Variables

### eperi\_key\_management\_plugin\_databaseId

- **Description:** Determines the database ID which is processed in the eperi Gateway.

- **Commandline:**

```
--eperi-key-management-plugin-databaseid=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

integer

- **Default Value:**

1

### eperi\_key\_management\_plugin\_encryption\_algorithm

- **Description:** This system variable is used to determine which algorithm the plugin will use to encrypt data.

- The recommended algorithm is

AES\_CTR

, but this algorithm is only available when MariaDB is built with recent versions of [OpenSSL](#). If the server is built with [wolfSSL](#) or [yaSSL](#), then this algorithm is not available. See [TLS and Cryptography Libraries Used by MariaDB](#) for more information about which libraries are used on which platforms.

- **Commandline:**

```
--eperi-key-management-plugin-encryption-algorithm=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

enumerated

- **Default Value:**

AES\_CBC

- **Valid Values:**

AES\_CBC

,

AES\_CTR

### eperi\_key\_management\_plugin\_encryption\_mode

- **Description:** Encryption mode.

- **Commandline:**

```
--eperi-key-management-plugin-encryption-mode=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

enumerated

- **Default Value:**

database

- **Valid Values:**

```
database
,
gateway
```

---

## eperi\_key\_management\_plugin\_oss1mt

- **Description:** Determines, whether the plugin should register callback functions for OpenSSL thread support.

- **Commandline:**

```
--eperi-key-management-plugin-oss1mt=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
boolean
```

- **Default Value:**

```
0
(Linux),
1
(Windows)
```

---

## eperi\_key\_management\_plugin\_port

- **Description:** Listener port for plugin.

- **Commandline:**

```
--eperi-key-management-plugin-port=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
integer
```

- **Default Value:**

```
14332
```

---

## eperi\_key\_management\_plugin\_url

- **Description:** URL to key server. The expected format of the URL is <host>:<port>. The port number is optional, and the port number defaults to 14333 if it is not specified.

- **Commandline:**

```
--eperi-key-management-plugin-url=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

```
string
```

- **Default Value:**

```
NULL
```

---

## eperi\_key\_management\_plugin\_url\_check\_disabled

- **Description:** Determines, whether the connection between plugin and eperi Gateway is tested at server startup.

- **Commandline:**

```
--eperi-key-management-plugin-url-check-disabled=value
```

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

boolean

- **Default Value:**

1

---

## Options

### eperi\_key\_management\_plugin

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF

- Disables the plugin without removing it from the

- `mysql.plugins`

- table.

- ON

- Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

- FORCE

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

- FORCE\_PLUS\_PERMANENT

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

- `UNINSTALL SONAME`

- or

- `UNINSTALL PLUGIN`

- while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

```
--eperi-key-management-plugin=value
```

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

- OFF

- ,

- ON

- ,

- FORCE

- ,

- FORCE\_PLUS\_PERMANENT

## See Also

- [Database Encryption - eperi](#)
- [eperi Gateway for Databases version 3.4 offers native MariaDB support](#)
- [eperi Customer Portal](#)

## 4.4.7.7 Encryption Plugin API

### Contents

1. [Encryption Plugin API](#)
2. [Current Encryption Plugins](#)
  1. [file\\_key\\_management](#)
    1. [Versions](#)
  2. [aws\\_key\\_management](#)
    1. [Versions](#)
  3. [example\\_key\\_management](#)
    1. [Versions](#)
  4. [debug\\_key\\_management](#)
    1. [Versions](#)
3. [Encryption Service](#)

MariaDB's [data-at-rest encryption](#) requires the use of a [key management and encryption plugin](#). These plugins are responsible both for the management of encryption keys and for the actual encryption and decryption of data.

MariaDB supports the use of [multiple encryption keys](#). Each encryption key uses a 32-bit integer as a key identifier. If the specific plugin supports [key rotation](#), then encryption keys can also be rotated, which creates a new version of the encryption key.

See [Data at Rest Encryption](#) and [Encryption Key Management](#) for more information.

## Encryption Plugin API

The Encryption plugin API was created to allow a plugin to:

- implement key management, provide encryption keys to the server on request and change them according to internal policies.
- implement actual data encryption and decryption with the algorithm defined by the plugin.

This is how the API reflects that:

```
/* Returned from get_latest_key_version() */
#define ENCRYPTION_KEY_VERSION_INVALID (~(unsigned int)0)
#define ENCRYPTION_KEY_NOT_ENCRYPTED (0)

#define ENCRYPTION_KEY_SYSTEM_DATA 1
#define ENCRYPTION_KEY_TEMPORARY_DATA 2

/* Returned from get_key() */
#define ENCRYPTION_KEY_BUFFER_TOO_SMALL (100)

#define ENCRYPTION_FLAG_DECRYPT 0
#define ENCRYPTION_FLAG_ENCRYPT 1
#define ENCRYPTION_FLAG_NOPAD 2

struct st_mariadb_encryption {
    int interface_version; /*< version plugin uses */

    /******
     * KEY MANAGEMENT *****/
    /**
     * Function returning Latest key version for a given key id.
     *
     * @return A version or ENCRYPTION_KEY_VERSION_INVALID to indicate an error.
     */
    unsigned int (*get_latest_key_version)(unsigned int key_id);

    /**
     * Function returning a key for a key version
     *
     * @param key_id      The requested key id
     * @param version     The requested key version
     * @param key         The key will be stored there. Can be NULL -
     *                   in which case no key will be returned
     * @param key_Length  in: key buffer size
     *                   out: the actual length of the key
     */
}
```

*This method can be used to query the key Length - the required*

```

buffer size - by passing key==NULL.

If the buffer size is less than the key length the content of the
key buffer is undefined (the plugin is free to partially fill it with
the key data or leave it untouched).

@return 0 on success, or
    ENCRYPTION_KEY_VERSION_INVALID, ENCRYPTION_KEY_BUFFER_TOO_SMALL
    or any other non-zero number for errors
*/
unsigned int (*get_key)(unsigned int key_id, unsigned int version,
                      unsigned char *key, unsigned int *key_length);

/********************* ENCRYPTION *****/
/*
The caller uses encryption as follows:
1. Create the encryption context object of the crypt_ctx_size() bytes.
2. Initialize it with crypt_ctx_init().
3. Repeat crypt_ctx_update() until there are no more data to encrypt.
4. Write the remaining output bytes and destroy the context object
   with crypt_ctx_finish().
*/
/***
    Returns the size of the encryption context object in bytes
*/
unsigned int (*crypt_ctx_size)(unsigned int key_id, unsigned int key_version);
/***
    Initializes the encryption context object.
*/
int (*crypt_ctx_init)(void *ctx, const unsigned char *key, unsigned int klen,
                      const unsigned char *iv, unsigned int ivlen, int flags,
                      unsigned int key_id, unsigned int key_version);
/***
    Processes (encrypts or decrypts) a chunk of data

    Writes the output to the dst buffer. note that it might write
    more bytes that were in the input. or less. or none at all.
*/
int (*crypt_ctx_update)(void *ctx, const unsigned char *src,
                       unsigned int slen, unsigned char *dst,
                       unsigned int *dlen);
/***
    Writes the remaining output bytes and destroys the encryption context

    crypt_ctx_update might've cached part of the output in the context,
    this method will flush these data out.
*/
int (*crypt_ctx_finish)(void *ctx, unsigned char *dst, unsigned int *dlen);
/***
    Returns the Length of the encrypted data

    It returns the exact length, given only the source length.
    Which means, this API only supports encryption algorithms where
    the length of the encrypted data only depends on the length of the
    input (a.k.a. compression is not supported).
*/
unsigned int (*encrypted_length)(unsigned int slen, unsigned int key_id,
                                unsigned int key_version);
};


```

The first method is used for key rotation. A plugin that doesn't support key rotation — for example, **file\_key\_management** — can return a fixed version for any valid key id. Note that it still has to return an error for an invalid key id. The version

ENCRYPTION\_KEY\_NOT\_ENCRYPTED

means that the data should not be encrypted.

The second method is used for key management, the server uses it to retrieve the key corresponding to a specific key identifier and a specific key version.

The last five methods deal with encryption. Note that they take the key to use and key identifier and version. This is needed because the server can derive a session-specific, user-specific, or a tablespace-specific key from the original encryption key as returned by

get\_key()  
, so the  
key

argument doesn't have to match the encryption key as the plugin knows it. On the other hand, the encryption algorithm may depend on the key identifier and version (and in the **example\_key\_management** plugin it does) so the plugin needs to know them to be able to encrypt the data.

Encryption methods are optional — if unset (as in the **debug\_key\_management** plugin), the server will fall back to AES\_CBC.

## Current Encryption Plugins

The MariaDB source tree has four encryption plugins. All these plugins are fairly simple and can serve as good examples of the Encryption plugin API.

### file\_key\_management

It reads encryption keys from a plain-text file. It supports two different encryption algorithms. It supports multiple encryption keys. It does not support key rotation. See the [File Key Management Plugin](#) article for more details.

#### Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.1.18
1.0	Gamma	MariaDB 10.1.13
1.0	Alpha	MariaDB 10.1.3

### aws\_key\_management

The AWS Key Management plugin uses the [Amazon Web Services \(AWS\) Key Management Service \(KMS\)](#) to generate and store AES keys on disk, in encrypted form, using the Customer Master Key (CMK) kept in AWS KMS. When MariaDB Server starts, the plugin will decrypt the encrypted keys, using the AWS KMS "Decrypt" API function. MariaDB data will then be encrypted and decrypted using the AES key. It supports multiple encryption keys. It supports key rotation.

See the [AWS Key Management Plugin](#) article for more details.

#### Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.2.6 , MariaDB 10.1.24
1.0	Beta	MariaDB 10.1.18
1.0	Experimental	MariaDB 10.1.13

### example\_key\_management

Uses random time-based generated keys, ignores key identifiers, supports key versions and key rotation. Uses AES\_ECB and AES\_CBC as encryption algorithms and changes them automatically together with key versions.

#### Versions

Version	Status	Introduced
1.0	Experimental	MariaDB 10.1.3

### debug\_key\_management

Key is generated from the version, user manually controls key rotation. Only supports key identifier 1, uses only AES\_CBC.

#### Versions

Version	Status	Introduced
1.0	Experimental	MariaDB 10.1.3

## Encryption Service

Encryption is generally needed on the very low level **inside the storage engine**. That is, the storage engine needs to support encryption and have access to the encryption and key management functionality. The usual way for a plugin to access some functionality in the server is via a service . In this case the server provides the Encryption Service for storage engines (and other interested plugins) to use. These service functions are directly hooked into encryption plugin methods (described above).

Service functions are declared as follows:

```

unsigned int encryption_key_get_latest_version(unsigned int key_id);
unsigned int encryption_key_get(unsigned int key_id, unsigned int key_version,
                               unsigned char *buffer, unsigned int *length);
unsigned int encryption_ctx_size(unsigned int key_id, unsigned int key_version);
int encryption_ctx_init(void *ctx, const unsigned char *key, unsigned int klen,
                       const unsigned char *iv, unsigned int ivlen, int flags,
                       unsigned int key_id, unsigned int key_version);
int encryption_ctx_update(void *ctx, const unsigned char *src,
                          unsigned int slen, unsigned char *dst,
                          unsigned int *dlen);
int encryption_ctx_finish(void *ctx, unsigned char *dst, unsigned int *dlen);
unsigned int encryption_encrypted_length(unsigned int slen, unsigned int key_id,
   unsigned int key_version);

```

There are also convenience helpers to check for a key or key version existence and to encrypt or decrypt a block of data with one function call.

```

unsigned int encryption_key_id_exists(unsigned int id);
unsigned int encryption_key_version_exists(unsigned int id,
   unsigned int version);
int encryption_crypt(const unsigned char *src, unsigned int slen,
                     unsigned char *dst, unsigned int *dlen,
                     const unsigned char *key, unsigned int klen,
                     const unsigned char *iv, unsigned int ivlen, int flags,
                     unsigned int key_id, unsigned int key_version);

```

## 4.4.8

### 4.4.8.1 Semisynchronous Replication

MariaDB starting with [10.3.3](#)

Semisynchronous replication is no longer a plugin in [MariaDB 10.3.3](#) and later. This removes some overhead and improves performance. See [MDEV-13073](#) for more information.

#### Contents

1. [Description](#)
2. [Installing the Plugin](#)
3. [Uninstalling the Plugin](#)
4. [Enabling Semisynchronous Replication](#)
  1. [Enabling Semisynchronous Replication on the Primary](#)
  2. [Enabling Semisynchronous Replication on the Replica](#)
5. [Configuring the Primary Timeout](#)
6. [Configuring the Primary Wait Point](#)
7. [Versions](#)
8. [System Variables](#)
  1. [rpl\\_semi\\_sync\\_master\\_enabled](#)
  2. [rpl\\_semi\\_sync\\_master\\_timeout](#)
  3. [rpl\\_semi\\_sync\\_master\\_trace\\_level](#)
  4. [rpl\\_semi\\_sync\\_master\\_wait\\_no\\_slave](#)
  5. [rpl\\_semi\\_sync\\_master\\_wait\\_point](#)
  6. [rpl\\_semi\\_sync\\_slave\\_delay\\_master](#)
  7. [rpl\\_semi\\_sync\\_slave\\_enabled](#)
  8. [rpl\\_semi\\_sync\\_slave\\_kill\\_conn\\_timeout](#)
  9. [rpl\\_semi\\_sync\\_slave\\_trace\\_level](#)
9. [Options](#)
  1. [rpl\\_semi\\_sync\\_master](#)
  2. [rpl\\_semi\\_sync\\_slave](#)
10. [Status Variables](#)

## Description

[Standard MariaDB replication](#) is asynchronous, but MariaDB also provides a semisynchronous replication option.

With regular asynchronous replication, replicas request events from the primary's binary log whenever the replicas are ready. The primary does not wait for a replica to confirm that an event has been received.

With fully synchronous replication, all replicas are required to respond that they have received the events. See [Galera Cluster](#).

Semisynchronous replication waits for just one replica to acknowledge that it has received and logged the events.

Semisynchronous replication therefore comes with some negative performance impact, but increased data integrity. Since the delay is based on the roundtrip time to the replica and back, this delay is minimized for servers in close proximity over fast networks.

In MariaDB 10.3 and later, semisynchronous replication is built into the server, so it can be enabled immediately in those versions.

In MariaDB 10.2 and before, semisynchronous replication requires the user to install a plugin on both the primary and the replica before it can be enabled.

## Installing the Plugin

MariaDB starting with 10.3.3

In MariaDB 10.3.3 and later, the Semisynchronous Replication feature is built into MariaDB server and is no longer provided by a plugin. **This means that installing the plugin is not supported on those versions.** In MariaDB 10.3.3 and later, you can skip right to [Enabling Semisynchronous Replication](#).

The semisynchronous replication plugin is actually two different plugins--one for the primary, and one for the replica. Shared libraries for both plugins are included with MariaDB. Although the plugins' shared libraries distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default prior to MariaDB 10.3.3. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing [INSTALL SONAME](#) or [INSTALL PLUGIN](#).

For example, if it's a primary:

```
INSTALL SONAME 'semisync_master';
```

Or if it's a replica:

```
INSTALL SONAME 'semisync_slave';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the [--plugin-load](#) or the [--plugin-load-add](#) options. This can be specified as a command-line argument to [mysqld](#) or it can be specified in a relevant server [option group](#) in an [option file](#).

For example, if it's a primary:

```
[mariadb]
...
plugin_load_add = semisync_master
```

Or if it's a replica:

```
[mariadb]
...
plugin_load_add = semisync_slave
```

## Uninstalling the Plugin

MariaDB starting with 10.3.3

In MariaDB 10.3.3 and later, the Semisynchronous Replication feature is built into MariaDB server and is no longer provided by a plugin. **This means that uninstalling the plugin is not supported on those versions.**

You can uninstall the plugin dynamically by executing [UNINSTALL SONAME](#) or [UNINSTALL PLUGIN](#).

For example, if it's a primary:

```
UNINSTALL SONAME 'semisync_master';
```

Or if it's a replica:

```
UNINSTALL SONAME 'semisync_slave';
```

If you installed the plugin by providing the [--plugin-load](#) or the [--plugin-load-add](#) options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Enabling Semisynchronous Replication

Semisynchronous replication can be enabled by setting the relevant system variables on the primary and the replica.

If a server needs to be able to switch between acting as a primary and a replica, then you can enable both the primary and replica system variables on the server. For example, you might need to do this if [MariaDB MaxScale](#) is being used to enable [auto-failover](#) or [switchover](#) with [MariaDB Monitor](#).

## Enabling Semisynchronous Replication on the Primary

Semisynchronous replication can be enabled on the primary by setting the `rpl_semi_sync_master_enabled` system variable to `ON`. It can be set dynamically with [SET GLOBAL](#). For example:

```
SET GLOBAL rpl_semi_sync_master_enabled=ON;
```

It can also be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
rpl_semi_sync_master_enabled=ON
```

## Enabling Semisynchronous Replication on the Replica

Semisynchronous replication can be enabled on the replica by setting the `rpl_semi_sync_slave_enabled` system variable to `ON`. It can be set dynamically with [SET GLOBAL](#). For example:

```
SET GLOBAL rpl_semi_sync_slave_enabled=ON;
```

It can also be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
rpl_semi_sync_slave_enabled=ON
```

If semisynchronous replication is enabled on a server when [replica threads](#) were already running, the replica I/O thread will need to be restarted to enable the replica to register as a semisynchronous replica when it connects to the primary. For example:

```
STOP SLAVE IO_THREAD;
START SLAVE IO_THREAD;
```

If this is not done, and the replica thread is already running, then it will continue to use asynchronous replication.

## Configuring the Primary Timeout

In semisynchronous replication, only after the events have been written to the relay log and flushed does the replica acknowledge receipt of a transaction's events. If the replica does not acknowledge the transaction before a certain amount of time has passed, then a timeout occurs and the primary switches to asynchronous replication. This will be reflected in the primary's [error log](#) with messages like the following:

```
[Warning] Timeout waiting for reply of binlog (file: mariadb-1-bin.000002, pos: 538), semi-sync up to file , position 0.
[Note] Semi-sync replication switched OFF.
```

When this occurs, the `Rpl_semi_sync_master_status` status variable will be switched to `OFF`.

When at least one semisynchronous replica catches up, semisynchronous replication is resumed. This will be reflected in the primary's [error log](#) with messages like the following:

```
[Note] Semi-sync replication switched ON with replica (server_id: 184137206) at (mariadb-1-bin.000002, 215076)
```

When this occurs, the `Rpl_semi_sync_master_status` status variable will be switched to `ON`.

The number of times that semisynchronous replication has been switched off can be checked by looking at the value of the `Rpl_semi_sync_master_no_times` status variable.

If you see a lot of timeouts like this in your environment, then you may want to change the timeout period. The timeout period can be changed by setting the `rpl_semi_sync_master_timeout` system variable. It can be set dynamically with [SET GLOBAL](#). For example:

```
SET GLOBAL rpl_semi_sync_master_timeout=20000;
```

It can also be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
rpl_semi_sync_master_timeout=20000
```

To determine a good value for the `rpl_semi_sync_master_timeout` system variable, you may want to look at the values of the `Rpl_semi_sync_master_net_avg_wait_time` and `Rpl_semi_sync_master_tx_avg_wait_time` status variables.

## Configuring the Primary Wait Point

In semisynchronous replication, there are two potential points at which the primary can wait for the replica acknowledge the receipt of a transaction's events. These two wait points have different advantages and disadvantages.

The wait point is configured by the `rpl_semi_sync_master_wait_point` system variable. The supported values are:

- AFTER\_SYNC
- AFTER\_COMMIT

It can be set dynamically with `SET GLOBAL`. For example:

```
SET GLOBAL rpl_semi_sync_master_wait_point='AFTER_SYNC';
```

It can also be set in a server [option group](#) in an [option file](#) prior to starting up the server. For example:

```
[mariadb]
...
rpl_semi_sync_master_wait_point=AFTER_SYNC
```

When this variable is set to

AFTER\_SYNC

, the primary performs the following steps:

1. Prepares the transaction in the storage engine.
2. Syncs the transaction to the [binary log](#).
3. Waits for acknowledgement from the replica.
4. Commits the transaction to the storage engine.
5. Returns an acknowledgement to the client.

The effects of the

AFTER\_SYNC

wait point are:

- All clients see the same data on the primary at the same time; after acknowledgement by the replica and after being committed to the storage engine on the primary.
- If the primary crashes, then failover should be lossless, because all transactions committed on the primary would have been replicated to the replica.
- However, if the primary crashes, then its [binary log](#) may also contain events for transactions that were prepared by the storage engine and written to the binary log, but that were never actually committed by the storage engine. As part of the server's [automatic crash recovery](#) process, the server may recover these prepared transactions when the server is restarted. This could cause the "old" crashed primary to become inconsistent with its former replicas when they have been reconfigured to replace the old primary with a new one. The old primary in such a scenario can be re-introduced only as a [semisync slave](#). The server post-crash recovery of the server configured with

```
rpl_semi_sync_slave_enabled = ON
```

ensures through [MDEV-21117](#) that the server will not have extra transactions. The reconfigured as semisync replica server's binlog gets truncated to discard transactions proven not to be committed, in any of their branches if they are multi-engine. Truncation does not occur though when there exists a non-transactional group of events beyond the truncation position in which case recovery reports an error. When the semisync replica recovery can't be carried out, the crashed primary may need to be rebuilt.

When this variable is set to

AFTER\_COMMIT

, the primary performs the following steps:

1. Prepares the transaction in the storage engine.
2. Syncs the transaction to the [binary log](#).
3. Commits the transaction to the storage engine.
4. Waits for acknowledgement from the replica.
5. Returns an acknowledgement to the client.

The effects of the

AFTER\_COMMIT  
wait point are:

- Other clients may see the committed transaction before the committing client.
- If the primary crashes, then failover may involve some data loss, because the primary may have committed transactions that had not yet been acknowledged by the replicas.

## Versions

Version	Status	Introduced
N/A	N/A	MariaDB 10.3.3
1.0	Stable	MariaDB 10.1.13
1.0	Gamma	MariaDB 10.0.13
1.0	Unknown	MariaDB 10.0.11
1.0	N/A	MariaDB 5.5

## System Variables

### rpl\_semi\_sync\_master\_enabled

- **Description:** Set to  
ON  
to enable semi-synchronous replication primary. Disabled by default.
- **Commandline:**  
--rpl-semi-sync-master-enabled[={0|1}]
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean
- **Default Value:**  
OFF

---

### rpl\_semi\_sync\_master\_timeout

- **Description:** The timeout value, in milliseconds, for semi-synchronous replication in the primary. If this timeout is exceeded in waiting on a commit for acknowledgement from a replica, the primary will revert to asynchronous replication.
  - When a timeout occurs, the [Rpl\\_semi\\_sync\\_master\\_status](#) status variable will also be switched to  
OFF
  - See [Configuring the Primary Timeout](#) for more information.
- **Commandline:**  
--rpl-semi-sync-master-timeout[=#]
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric
- **Default Value:**  
10000  
(10 seconds)
- **Range:**  
0  
to  
18446744073709551615

## rpl\_semi\_sync\_master\_trace\_level

- **Description:** The tracing level for semi-sync replication. Four levels are defined:

- 1 : General level, including for example time function failures.
- 16 : More detailed level, with more verbose information.
- 32 : Net wait level, including more information about network waits.
- 64 : Function level, including information about function entries and exits.

- **Commandline:**

```
--rpl-semi-sync-master-trace-level[=#]
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

numeric

- **Default Value:**

32

- **Range:**

0  
to  
18446744073709551615

## rpl\_semi\_sync\_master\_wait\_no\_slave

- **Description:** If set to

ON

, the default, the replica count (recorded by [Rpl\\_semi\\_sync\\_master\\_clients](#)) may drop to zero, and the primary will still wait for the timeout period. If set to

OFF

, the primary will revert to asynchronous replication as soon as the replica count drops to zero.

- **Commandline:**

```
--rpl-semi-sync-master-wait-no-slave[={0|1}]
```

- **Scope:** Global

- **Dynamic:** Yes

- **Data Type:**

boolean

- **Default Value:**

ON

## rpl\_semi\_sync\_master\_wait\_point

- **Description:** Whether the transaction should wait for semi-sync acknowledgement after having synced the binlog (

AFTER\_SYNC

), or after having committed in storage engine (

AFTER\_COMMIT

, the default).

- When this variable is set to

AFTER\_SYNC

, the primary performs the following steps:

1. Prepares the transaction in the storage engine.

2. Syncs the transaction to the [binary log](#).

3. Waits for acknowledgement from the replica.
  4. Commits the transaction to the storage engine.
  5. Returns an acknowledgement to the client.
- When this variable is set to  
AFTER\_COMMIT  
, the primary performs the following steps:
    1. Prepares the transaction in the storage engine.
    2. Syncs the transaction to the [binary log](#).
    3. Commits the transaction to the storage engine.
    4. Waits for acknowledgement from the replica.
    5. Returns an acknowledgement to the client.
  - In [MariaDB 10.1.2](#) and before, this system variable does not exist. However, in those versions, the primary waits for the acknowledgement from replicas at a point that is equivalent to  
AFTER\_COMMIT  
    - See [Configuring the Primary Wait Point](#) for more information.
- **Commandline:**  
--rpl-semi-sync-master-wait-point=value
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
enum
  - **Default Value:**  
AFTER\_COMMIT
  - **Valid Values:**  
AFTER\_SYNC  
,  
AFTER\_COMMIT
  - **Introduced:** [MariaDB 10.1.3](#)
- 

### rpl\_semi\_sync\_slave\_delay\_master

- **Description:** Only write primary info file when ack is needed.
  - **Commandline:**  
--rpl-semi-sync-slave-delay-master[={0|1}]
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
boolean
  - **Default Value:**  
OFF
  - **Introduced:** [MariaDB 10.3.3](#)
- 

### rpl\_semi\_sync\_slave\_enabled

- **Description:** Set to  
ON  
to enable semi-synchronous replication replica. Disabled by default.
- **Commandline:**  
--rpl-semi-sync-slave-enabled[={0|1}]
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean
- **Default Value:**  
OFF

## rpl\_semi\_sync\_slave\_kill\_conn\_timeout

- **Description:** Timeout for the mysql connection used to kill the replica io\_thread's connection on primary. This timeout comes into play when stop slave is executed.
- **Commandline:**  
--rpl-semi-sync-slave-kill-conn-timeout[={0|1}]
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric
- **Default Value:**  
5
- **Range:**  
0  
to  
4294967295
- **Introduced:** MariaDB 10.3.3

## rpl\_semi\_sync\_slave\_trace\_level

- **Description:** The tracing level for semi-sync replication. The levels are the same as for [rpl\\_semi\\_sync\\_master\\_trace\\_level](#).
- **Commandline:**  
--rpl-semi-sync-slave-trace\_level[=#]
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
numeric
- **Default Value:**  
32
- **Range:**  
0  
to  
18446744073709551615

# Options

## rpl\_semi\_sync\_master

- **Description:** Controls how the server should treat the plugin when the server starts up.
  - Valid values are:
    - OFF
      - Disables the plugin without removing it from the [mysql.plugins](#) table.
    - ON
      - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
    - FORCE
      - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

- - FORCE\_PLUS\_PERMANENT
      - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with [UNINSTALL SONAME](#) or [UNINSTALL PLUGIN](#) while the server is running.
    - See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.
  - **Commandline:**

```
--rpl-semi-sync-master=value
```
  - **Data Type:**

enumerated
  - **Default Value:**

ON
  - **Valid Values:**

OFF  
,  
ON  
,  
FORCE  
,  
FORCE\_PLUS\_PERMANENT
  - **Removed:** [MariaDB 10.3.3](#)
- 

## rpl\_semi\_sync\_slave

- **Description:** Controls how the server should treat the plugin when the server starts up.
    - Valid values are:
      - OFF
        - Disables the plugin without removing it from the [mysql.plugins](#) table.
      - ON
        - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
      - FORCE
        - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
      - FORCE\_PLUS\_PERMANENT
        - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with [UNINSTALL SONAME](#) or [UNINSTALL PLUGIN](#) while the server is running.
    - See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.
  - **Commandline:**

```
--rpl-semi-sync-slave=value
```
  - **Data Type:**

enumerated
  - **Default Value:**

ON
  - **Valid Values:**

OFF  
,  
ON  
,  
FORCE  
,  
FORCE\_PLUS\_PERMANENT
  - **Removed:** [MariaDB 10.3.3](#)
- 

## Status Variables

For a list of status variables added when the plugin is installed, see [Semisynchronous Replication Plugin Status Variables](#).

## 4.4.8.2 WSREP\_INFO Plugin

The

WSREP\_INFO  
plugin library contains the following plugins:

- WSREP\_MEMBERSHIP
- WSREP\_STATUS

The

WSREP\_MEMBERSHIP  
plugin creates the `WSREP_MEMBERSHIP` table in the `INFORMATION_SCHEMA` database. The plugin also adds the `SHOW WSREP_MEMBERSHIP` statement.

The

WSREP\_STATUS  
plugin creates the `WSREP_STATUS` table in the `INFORMATION_SCHEMA` database. The plugin also adds the `SHOW WSREP_STATUS` statement.

These tables and statements provide information about `Galera`. Only users with the `SUPER` privilege can access this information.

### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Example](#)
4. [Versions](#)
5. [Options](#)
  1. [wsrep\\_membership](#)
  2. [wsrep\\_status](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing `INSTALL SONAME` or `INSTALL PLUGIN`. For example:

```
INSTALL SONAME 'wsrep_info';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the `--plugin-load` or the `--plugin-load-add` options. This can be specified as a command-line argument to `mysql` or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = wsrep_info
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing `UNINSTALL SONAME` or `UNINSTALL PLUGIN`. For example:

```
UNINSTALL SONAME 'wsrep_info';
```

If you installed the plugin by providing the `--plugin-load` or the `--plugin-load-add` options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Example

```
SHOW TABLES FROM information_schema LIKE 'WSREP%';
+-----+
| Tables_in_information_schema (WSREP%) |
+-----+
| WSREP_STATUS                         |
| WSREP_MEMBERSHIP                      |
+-----+
```

## Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.1.18
1.0	Gamma	MariaDB 10.1.13
1.0	Alpha	MariaDB 10.1.2

## Options

### wsrep\_membership

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF
  - Disables the plugin without removing it from the `mysql.plugins` table.
- ON
  - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
- FORCE
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
- FORCE\_PLUS\_PERMANENT
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with `UNINSTALL SONAME` or `UNINSTALL PLUGIN` while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

`--wsrep-membership=value`

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF  
,  
ON  
,  
FORCE  
,  
FORCE\_PLUS\_PERMANENT

### wsrep\_status

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

-

- OFF
  - Disables the plugin without removing it from the `mysql.plugins` table.
- ON
  - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
- FORCE
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
- FORCE\_PLUS\_PERMANENT
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with `UNINSTALL SONAME` or `UNINSTALL PLUGIN` while the server is running.

◦ See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

```
--wsrep-status=value
```

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF  
,

ON  
,

FORCE  
,

FORCE\_PLUS\_PERMANENT

## 4.4.9 Storage Engines

### 4.4.10

#### 4.4.10.1 Feedback Plugin

The `feedback` plugin is designed to collect and, optionally, upload configuration and usage information to [MariaDB.org](#) or to any other configured URL.

See the [MariaDB User Feedback](#) page on MariaDB.org to see collected MariaDB usage statistics.

The `feedback` plugin exists in all MariaDB versions.

MariaDB is distributed with this plugin included, but it is not enabled by default. On Windows, this plugin is part of the server and has a special checkbox in the installer window. Either way, you need to explicitly install and enable it in order for feedback data to be sent.

### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Enabling the Plugin](#)
4. [Verifying the Plugin's Status](#)
5. [Collecting Data](#)
6. [Sending Data](#)
7. [Versions](#)
8. [System Variables](#)
  1. [feedback\\_http\\_proxy](#)
  2. [feedback\\_send\\_retry\\_wait](#)
  3. [feedback\\_send\\_timeout](#)
  4. [feedback\\_server\\_uid](#)
  5. [feedback\\_url](#)
  6. [feedback\\_user\\_info](#)
9. [Options](#)
  1. [feedback](#)

# Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'feedback';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#) . For example:

```
[mariadb]
...
plugin_load_add = feedback
```

# Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'feedback';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#) , then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

# Enabling the Plugin

You can enable the plugin by setting the

```
feedback
```

option to  
ON

in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
feedback=ON
```

In Windows, the plugin can also be enabled during a new [MSI](#) installation. The MSI GUI installation provides the "Enable feedback plugin" checkbox to enable the plugin. The MSI command-line installation provides the FEEDBACK=1 command-line option to enable the plugin.

See the next section for how to verify the plugin is installed and active and (if needed) install the plugin.

## Verifying the Plugin's Status

To verify whether the

```
feedback
plugin is installed and enabled, execute the
```

[SHOW PLUGINS](#)

statement or query the

[information\\_schema.plugins](#)

table. For example:

```
SELECT plugin_status FROM information_schema.plugins
WHERE plugin_name = 'feedback';
```

If that

```
SELECT
returns no rows, then you still need to install the plugin.
```

When the plugin is installed and enabled, you will see:

```
SELECT plugin_status FROM information_schema.plugins
WHERE plugin_name = 'feedback';
+-----+
| plugin_status |
+-----+
| ACTIVE        |
+-----+
```

If you see

```
DISABLED
instead of
ACTIVE
, then you still need to enable the plugin.
```

## Collecting Data

The

```
feedback
plugin will collect:
```

- Certain rows from [SHOW STATUS](#) and [SHOW VARIABLES](#).
- All installed [plugins](#) and their versions.
- System information such as CPU count, memory, architecture, and OS/linux distribution.
- The [feedback\\_server\\_uid](#), which is a SHA1 hash of the MAC address of the first network interface and the TCP port that the server listens on.

The

```
feedback
plugin creates the FEEDBACK table in the INFORMATION\_SCHEMA database. To see the data that has been collected by the plugin, you can execute:
```

```
SELECT * FROM information_schema.feedback;
```

Only the contents of this table are sent to the [feedback\\_url](#).

MariaDB stores collation usage statistics. Each collation that has been used by the server will have a record in "SELECT \* FROM [information\\_schema.feedback](#)" output, for example:

VARIABLE_NAME	VARIABLE_VALUE
Collation used utf8_unicode_ci	10
Collation used latin1_general_ci	20

Collations that have not been used will not be included into the result.

## Sending Data

The

feedback  
plugin sends the data using a  
POST

request to any URL or a list of URLs that you specify by setting the [feedback\\_url](#) system variable. By default, this is set to the following URL:

- [https://mariadb.org/feedback\\_plugin/post](https://mariadb.org/feedback_plugin/post)

Both HTTP and HTTPS protocols are supported.

If HTTP traffic requires a proxy in your environment, then you can specify the proxy by setting the [feedback\\_http\\_proxy](#) system variable.

If the [feedback\\_url](#) system variable is not set to an empty string, then the plugin will automatically send a report to all URLs in the list a few minutes after the server starts up and then once a week after that.

If the [feedback\\_url](#) system variable is set to an empty string, then the plugin will **not** automatically send any data. This may be necessary if outbound HTTP communication from your database server is not permitted. In this case, you can still upload the data manually, if you'd like.

First, generate the report file with the MariaDB command-line [mysql](#) client:

```
$ mysql -e 'select * from information_schema.feedback' > report.txt
```

Then you can upload the generated

report.txt  
[here](#) using your web browser.

Or you can do it from the command line with tools such as [curl](#). For example:

```
$ curl -F data=@report.txt https://mariadb.org/feedback_plugin/post
```

Manual uploading allows you to be absolutely sure that we receive only the data shown in the [INFORMATION\\_SCHEMA.FEEDBACK](#) table and that no private or sensitive information is being sent.

## Versions

Version	Status	Introduced
1.1	Stable	<a href="#">MariaDB 10.0.10</a>
1.1	Beta	<a href="#">MariaDB 5.5.20</a> , <a href="#">MariaDB 5.3.3</a>

## System Variables

### [feedback\\_http\\_proxy](#)

- **Description:** Proxy server for use when http calls cannot be made, such as in a firewall environment. The format is host:port
- **Commandline:**  
--feedback-http=proxy=value
- **Read-only:** Yes
- **Data Type:** string
- **Default Value:**  
''  
(empty)

## `feedback_send_retry_wait`

- **Description:** Time in seconds before retrying if the plugin failed to send the data for any reason.
- **Commandline:**

```
--feedback-send-retry-wait=#
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:** numeric
- **Default Value:**

60

- **Valid Values:**

```
1  
to  
86400
```

---

## `feedback_send_timeout`

- **Description:** An attempt to send the data times out and fails after this many seconds.
- **Commandline:**

```
--feedback-send-timeout=#
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:** numeric
- **Default Value:**

60

- **Valid Values:**

```
1  
to  
86400
```

---

## `feedback_server_uid`

- **Description:** Automatically calculated server unique id hash.
- **Scope:** Global
- **Dynamic:** No
- **Data Type:** string

## `feedback_url`

- **Description:** URL to which the data is sent. More than one URL, separated by spaces, can be specified. Set it to an empty string to disable data sending.
- **Commandline:**

```
--feedback-url=url
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:** string
- **Default Value:**

[https://mariadb.org/feedback\\_plugin/post](https://mariadb.org/feedback_plugin/post)

## `feedback_user_info`

- **Description:** The value of this option is not used by the plugin, but it is included in the feedback data. It can be used to add any user-specified string to the report. This could be used to help to identify it. For example, a support contract number, or a computer name (if you collect reports internally by specifying your own

```
    feedback-url  
).
```

- **Commandline:**

```
--feedback-user-info=string
```

- **Scope:** Global
- **Dynamic:** No
- **Data Type:** string
- **Default Value:** Empty string

# Options

## `feedback`

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF
  - Disables the plugin without removing it from the `mysql.plugins` table.
- ON
  - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
- FORCE
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
- FORCE\_PLUS\_PERMANENT
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

`UNINSTALL SONAME`

or

`UNINSTALL PLUGIN`

while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

```
--feedback=value
```

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF

,

```
ON
,
FORCE
,
FORCE_PLUS_PERMANENT
```

## 4.4.10.2 Locales Plugin

The

### LOCALES

plugin creates the `LOCALES` table in the `INFORMATION_SCHEMA` database. The plugin also adds the `SHOW LOCALES` statement. The table and statement can be queried to see all `locales` that are compiled into the server.

### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Example](#)
4. [Versions](#)
5. [Options](#)
  1. [locales](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing `INSTALL SONAME` or `INSTALL PLUGIN`. For example:

```
INSTALL SONAME 'locales';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the `--plugin-load` or the `--plugin-load-add` options. This can be specified as a command-line argument to `mysql` or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = locales
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing `UNINSTALL SONAME` or `UNINSTALL PLUGIN`. For example:

```
UNINSTALL SONAME 'locales';
```

If you installed the plugin by providing the `--plugin-load` or the `--plugin-load-add` options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Example

ID	NAME	DESCRIPTION	MAX_MONTH_NAME_LENGTH	MAX_DAY_NAME_LENGTH	DECIMAL_POINT	THOUSAND_SEP
0	en_US	English - United States	9	9	.	,
1	en_GB	English - United Kingdom	9	9	.	,
2	ja_JP	Japanese - Japan	3	3	.	,
3	sv_SE	Swedish - Sweden	9	7	,	
4	de_DE	German - Germany	9	10	,	.
5	fr_FR	French - France	9	8	,	
6	ar_AE	Arabic - United Arab Emirates	6	8	.	,
7	ar_BH	Arabic - Bahrain	6	8	.	,
8	ar_JO	Arabic - Jordan	12	8	.	,
...						
106	no_NO	Norwegian - Norway	9	7	,	.
107	sv_FI	Swedish - Finland	9	7	,	
108	zh_HK	Chinese - Hong Kong SAR	3	3	.	,
109	el_GR	Greek - Greece	11	9	,	.
110	rm_CH	Romanish - Switzerland	9	9	,	.

## Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.1.13
1.0	Gamma	MariaDB 10.0.10
1.0	Alpha	MariaDB 10.0.4

## Options

### locales

- **Description:** Controls how the server should treat the plugin when the server starts up.

◦ Valid values are:

- OFF
  - Disables the plugin without removing it from the `mysql.plugins` table.
- ON
  - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
- FORCE
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
- FORCE\_PLUS\_PERMANENT
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with `UNINSTALL SONAME` or `UNINSTALL PLUGIN` while the server is running.

◦ See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

`--locales=value`

- **Data Type:**

    enumerated

- **Default Value:**

    ON

- **Valid Values:**

    OFF

    ,

    ON

    ,

    FORCE

    ,

## 4.4.10.3 METADATA\_LOCK\_INFO Plugin

The

### METADATA\_LOCK\_INFO

plugin creates the `METADATA_LOCK_INFO` table in the `INFORMATION_SCHEMA` database. This table shows active `metadata locks`. The table will be empty if there are no active metadata locks.

### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Examples](#)
  1. [Viewing all Metadata Locks](#)
  2. [Matching Metadata Locks with Threads and Queries](#)
4. [Versions](#)
5. [Options](#)
  1. [metadata\\_lock\\_info](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing `INSTALL SONAME` or `INSTALL PLUGIN`. For example:

```
INSTALL SONAME 'metadata_lock_info';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the `--plugin-load` or the `--plugin-load-add` options. This can be specified as a command-line argument to `mysql` or it can be specified in a relevant server `option group` in an `option file`. For example:

```
[mariadb]
...
plugin_load_add = metadata_lock_info
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing `UNINSTALL SONAME` or `UNINSTALL PLUGIN`. For example:

```
UNINSTALL SONAME 'metadata_lock_info';
```

If you installed the plugin by providing the `--plugin-load` or the `--plugin-load-add` options in a relevant server `option group` in an `option file`, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Examples

### Viewing all Metadata Locks

```
SELECT * FROM information_schema.metadata_lock_info;
+-----+-----+-----+-----+-----+
| THREAD_ID | LOCK_MODE          | LOCK_DURATION | LOCK_TYPE        | TABLE_SCHEMA | TABLE_NAME |
+-----+-----+-----+-----+-----+
| 31       | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT   | Global read lock |            |            |
| 31       | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT   | Commit lock      |            |            |
| 31       | MDL_INTENTION_EXCLUSIVE | MDL_EXPLICIT   | Schema metadata lock | dbname     |            |
| 31       | MDL_SHARED_NO_READ_WRITE | MDL_EXPLICIT   | Table metadata lock | dbname     | exotics    |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

### Matching Metadata Locks with Threads and Queries

```

SELECT
CONCAT('Thread ',P.ID,' executing ''',P.INFO,'" IS LOCKED BY Thread ',
M.THREAD_ID) WhoLocksWho
FROM INFORMATION_SCHEMA.PROCESSLIST P,
INFORMATION_SCHEMA.METADATA_LOCK_INFO M
WHERE LOCATE(lower(lock_type), lower(state))>0;
+-----+
| WhoLocksWho |
+-----+
| Thread 3 executing "INSERT INTO foo ( b ) VALUES ( 'FOO' )" IS LOCKED BY Thread 2 |
+-----+
1 row in set (0.00 sec)

SHOW PROCESSLIST;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host      | db   | Command | Time | State          | Info           | Prog
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | root | localhost | test | Sleep   | 123 | NULL            |                | 0.0
| 3 | root | localhost | test | Query   | 103 | Waiting for global read lock | INSERT INTO foo ( b ) VALUES ( 'FOO' ) | 0.0
| 4 | root | localhost | test | Query   | 0    | init            | SHOW PROCESSLIST | 0.0
+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

## Versions

Version	Status	Introduced
0.1	Stable	MariaDB 10.1.13
0.1	Beta	MariaDB 10.0.10
0.1	Alpha	MariaDB 10.0.7

## Options

### metadata\_lock\_info

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF
  - Disables the plugin without removing it from the `mysql.plugins` table.
- ON
  - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
- FORCE
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
- FORCE\_PLUS\_PERMANENT
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with `UNINSTALL SONAME` or `UNINSTALL PLUGIN` while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

--metadata-lock-info=value

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF

,

ON

,

```
FORCE
,
FORCE_PLUS_PERMANENT
```

## 4.4.10.4 Query Cache Information Plugin

The

`QUERY_CACHE_INFO`

plugin creates the `QUERY_CACHE_INFO` table in the `INFORMATION_SCHEMA` database. This table shows all queries in the query cache.

Querying this table acquires the query cache lock and will result in lock waits for queries that are using or expiring from the query cache. You must have the `PROCESS` privilege to query this table.

### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Example](#)
4. [Versions](#)
5. [Options](#)
  1. [query\\_cache\\_info](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing `INSTALL SONAME` or `INSTALL PLUGIN`. For example:

```
INSTALL SONAME 'query_cache_info';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the `--plugin-load` or the `--plugin-load-add` options. This can be specified as a command-line argument to `mysql` or it can be specified in a relevant server `option group` in an `option file`. For example:

```
[mariadb]
...
plugin_load_add = query_cache_info
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing `UNINSTALL SONAME` or `UNINSTALL PLUGIN`. For example:

```
UNINSTALL SONAME 'query_cache_info';
```

If you installed the plugin by providing the `--plugin-load` or the `--plugin-load-add` options in a relevant server `option group` in an `option file`, then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Example

```
select statement_schema, statement_text, result_blocks_count,
       result_blocks_size from information_schema.query_cache_info;
+-----+-----+-----+
| statement_schema | statement_text    | result_blocks_count | result_blocks_size |
+-----+-----+-----+
| test           | select * from t1 |          1 |            512 |
+-----+-----+-----+
```

## Versions

Version	Status	Introduced
1.1	Stable	<a href="#">MariaDB 10.1.13</a>
1.1	Gamma	<a href="#">MariaDB 10.1.8</a>
1.0	Gamma	<a href="#">MariaDB 10.0.10</a>

## Options

### query\_cache\_info

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF
    - Disables the plugin without removing it from the `mysql.plugins` table.
  - ON
    - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
  - FORCE
    - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
  - FORCE\_PLUS\_PERMANENT
    - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with `UNINSTALL SONAME` or `UNINSTALL PLUGIN` while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

```
--query-cache-info=value
```

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF  
,  
ON  
,  
FORCE  
,  
FORCE\_PLUS\_PERMANENT

---

## 4.4.10.5 Query Response Time Plugin

The

```
query_response_time
```

plugin creates the `QUERY_RESPONSE_TIME` table in the `INFORMATION_SCHEMA` database. The plugin also adds the `SHOW QUERY_RESPONSE_TIME` and `FLUSH QUERY_RESPONSE_TIME` statements.

The `slow query log` provides exact information about queries that take a long time to execute. However, sometimes there are a large number of queries that each take a very short amount of time to execute. This feature provides a tool for analyzing that information by counting and displaying the number of queries according to the the length of time they took to execute.

This feature is based on Percona's [Response Time Distribution](#).

## Contents

- 1. [Installing the Plugin](#)
- 2. [Uninstalling the Plugin](#)
- 3. [Response Time Distribution](#)
- 4. [Using the Plugin](#)
  - 1. [Using the Information Schema Table](#)
  - 2. [Using the SHOW Statement](#)
  - 3. [Flushing Plugin Data](#)
- 5. [Versions](#)
- 6. [System Variables](#)
  - 1. [query\\_response\\_time\\_flush](#)
  - 2. [query\\_response\\_time\\_range\\_base](#)
  - 3. [query\\_response\\_time\\_exec\\_time\\_debug](#)
  - 4. [query\\_response\\_time\\_stats](#)
- 7. [Options](#)
  - 1. [query\\_response\\_time](#)
  - 2. [query\\_response\\_time\\_audit](#)

## Installing the Plugin

This shared library actually consists of two different plugins:

- - QUERY\_RESPONSE\_TIME
    - An INFORMATION\_SCHEMA plugin that exposes statistics.
- - QUERY\_RESPONSE\_TIME\_AUDIT
    - audit plugin, collects statistics.

Both plugins need to be installed to get meaningful statistics.

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing [INSTALL SONAME](#) or [INSTALL PLUGIN](#). For example:

```
INSTALL SONAME 'query_response_time';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the [--plugin-load](#) or the [--plugin-load-add](#) options. This can be specified as a command-line argument to [mysqld](#) or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = query_response_time
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing [UNINSTALL SONAME](#) or [UNINSTALL PLUGIN](#). For example:

```
UNINSTALL SONAME 'query_response_time';
```

If you installed the plugin by providing the [--plugin-load](#) or the [--plugin-load-add](#) options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Response Time Distribution

The user can define time intervals that divide the range 0 to positive infinity into smaller intervals and then collect the number of commands whose execution times fall into each of those intervals.

Each interval is described as:

```
(range_base ^ n; range_base ^ (n+1)]
```

The range\_base is some positive number (see Limitations). The interval is defined as the difference between two nearby powers of the range base.

For example, if the range base=10, we have the following intervals:

```
(0; 10 ^ -6], (10 ^ -6; 10 ^ -5], (10 ^ -5; 10 ^ -4], ...,
 (10 ^ -1; 10 ^1], (10^1; 10^2]...(10^7; positive infinity]
```

or

```
(0; 0.000001], (0.000001; 0.000010], (0.000010; 0.000100], ...,
 (0.100000; 1.0]; (1.0; 10.0]...(100000; positive infinity]
```

For each interval, a count is made of the queries with execution times that fell into that interval.

You can select the range of the intervals by changing the range base. For example, for base range=2 we have the following intervals:

```
(0; 2 ^ -19], (2 ^ -19; 2 ^ -18], (2 ^ -18; 2 ^ -17], ...,
 (2 ^ -1; 2 ^1], (2 ^ 1; 2 ^ 2]...(2 ^ 25; positive infinity]
```

or

```
(0; 0.000001], (0.000001, 0.000003], ...,
 (0.25; 0.5], (0.5; 2], (2; 4]...(8388608; positive infinity]
```

Small numbers look strange (i.e., don't look like powers of 2), because we lose precision on division when the ranges are calculated at runtime. In the resulting table, you look at the high boundary of the range.

For example, you may see:

```
SELECT * FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME;
+-----+-----+
| TIME      | COUNT | TOTAL      |
+-----+-----+
| 0.000001 |     0 | 0.000000 |
| 0.000010 |    17 | 0.000094 |
| 0.000100 |   4301 | 0.236555 |
| 0.001000 |  1499 | 0.824450 |
| 0.010000 | 14851 | 81.680502 |
| 0.100000 |  8066 | 443.635693 |
| 1.000000 |     0 | 0.000000 |
| 10.000000 |     0 | 0.000000 |
| 100.000000 |    1 | 55.937094 |
| 1000.000000 |   0 | 0.000000 |
| 10000.000000 |   0 | 0.000000 |
| 100000.000000 |   0 | 0.000000 |
| 1000000.000000 |   0 | 0.000000 |
| TOO LONG |     0 | TOO LONG  |
+-----+-----+
```

This means there were:

```
* 17 queries with 0.000001 < query execution time < = 0.000010 seconds; total execution time of the 17 queries = 0.000094 seconds

* 4301 queries with 0.000010 < query execution time < = 0.000100 seconds; total execution time of the 4301 queries = 0.236555 seconds

* 1499 queries with 0.000100 < query execution time < = 0.001000 seconds; total execution time of the 1499 queries = 0.824450 seconds

* 14851 queries with 0.001000 < query execution time < = 0.010000 seconds; total execution time of the 14851 queries = 81.680502 seconds

* 8066 queries with 0.010000 < query execution time < = 0.100000 seconds; total execution time of the 8066 queries = 443.635693 seconds

* 1 query with 10.000000 < query execution time < = 100.0000 seconds; total execution time of the 1 query = 55.937094 seconds
```

## Using the Plugin

### Using the Information Schema Table

You can get the distribution by querying the the `QUERY_RESPONSE_TIME` table in the `INFORMATION_SCHEMA` database. For example:

```
SELECT * FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME;
```

You can also write more complex queries. For example:

```
SELECT c.count, c.time,
(SELECT SUM(a.count) FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME AS a
 WHERE a.count != 0) AS query_count,
(SELECT COUNT(*)      FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME AS b
 WHERE b.count != 0) AS not_zero_region_count,
(SELECT COUNT(*)      FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME) AS region_count
FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME AS c
WHERE c.count > 0;
```

Note: If `query_response_time_stats` is set to

ON

, then the execution times for these two SELECT queries will also be collected.

## Using the SHOW Statement

As an alternative to the `QUERY_RESPONSE_TIME` table in the `INFORMATION_SCHEMA` database, you can also use the `SHOW QUERY_RESPONSE_TIME` statement. For example:

```
SHOW QUERY_RESPONSE_TIME;
```

## Flushing Plugin Data

Flushing the plugin data does two things:

- Clears the collected times from the `QUERY_RESPONSE_TIME` table in the `INFORMATION_SCHEMA` database.
- Reads the value of `query_response_time_range_base` and uses it to set the range base for the table.

Plugin data can be flushed with the `FLUSH QUERY_RESPONSE_TIME` statement. For example:

```
FLUSH QUERY_RESPONSE_TIME;
```

Setting the `query_response_time_flush` system variable has the same effect. For example:

```
SET GLOBAL query_response_time_flush=1;
```

## Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.1.13
1.0	Gamma	MariaDB 10.0.10
1.0	Alpha	MariaDB 10.0.4

## System Variables

### query\_response\_time\_flush

- **Description:** Updating this variable flushes the statistics and re-reads `query_response_time_range_base`.
- **Commandline:** None
- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean
- **Default Value:**  
OFF

### query\_response\_time\_range\_base

- **Description:** Select base of log for  
    QUERY\_RESPONSE\_TIME  
    ranges. WARNING: variable change takes affect only after flush.
  - **Commandline:**  
    --query-response-time-range-base=#
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    numeric
  - **Default Value:**  
    10
  - **Range:**  
    2  
    to  
    1000
- 

## query\_response\_time\_exec\_time\_debug

- **Description:** Pretend queries take this many microseconds. When 0 (the default) use the actual execution time.
    - This system variable is only available when the plugin is a [debug build](#).
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    numeric
  - **Default Value:**  
    0
  - **Range:**  
    0  
    to  
    31536000
- 

## query\_response\_time\_stats

- **Description:** Enable or disable query response time statistics collecting
  - **Commandline:**  
    query-response-time-stats[={0|1}]
  - **Scope:** Global
  - **Dynamic:** Yes
  - **Data Type:**  
    boolean
  - **Default Value:**  
    OFF
- 

# Options

## query\_response\_time

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF
  - Disables the plugin without removing it from the [mysql.plugins](#) table.
- ON
  - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
- FORCE
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
- FORCE\_PLUS\_PERMANENT
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with [UNINSTALL SONAME](#) or [UNINSTALL PLUGIN](#) while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

- `--query-response-time=value`

- **Data Type:**

- enumerated

- **Default Value:**

- ON

- **Valid Values:**

- OFF
- ,
- ON
- ,
- FORCE
- ,
- FORCE\_PLUS\_PERMANENT

## query\_response\_time\_audit

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF
  - Disables the plugin without removing it from the [mysql.plugins](#) table.
- ON
  - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
- FORCE
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
- FORCE\_PLUS\_PERMANENT
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with [UNINSTALL SONAME](#) or [UNINSTALL PLUGIN](#) while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

- `--query-response-time-audit=value`

- **Data Type:**

- enumerated

- **Default Value:**

- ON

- **Valid Values:**

- OFF
- ,
- ON

```
,  
FORCE  
,
```

## 4.4.10.6 SQL Error Log Plugin

### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Example](#)
4. [Versions](#)
5. [System Variables](#)
  1. [sql\\_error\\_log\\_filename](#)
  2. [sql\\_error\\_log\\_rate](#)
  3. [sql\\_error\\_log\\_rotate](#)
  4. [sql\\_error\\_log\\_rotations](#)
  5. [sql\\_error\\_log\\_size\\_limit](#)
6. [Options](#)
  1. [sql\\_error\\_log](#)

The

```
SQL_ERROR_LOG
```

plugin collects errors sent to clients in a log file defined by [sql\\_error\\_log\\_filename](#) , so that they can later be analyzed. The log file can be rotated if [sql\\_error\\_log\\_rotate](#) is set.

Errors are logged as they happen and an error will be logged even if it was handled by a [condition handler](#) and was never technically *sent* to the client.

Comments are also logged, which can make the log easier to search. But this is only possible if the client does not strip the comments away. For example, [mysql](#) command-line client only leaves comments when started with the [--comments](#) option.

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing [INSTALL SONAME](#) or [INSTALL PLUGIN](#) . For example:

```
INSTALL SONAME 'sql_errlog';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the [--plugin-load](#) or the [--plugin-load-add](#) options. This can be specified as a command-line argument to [mysqld](#) or it can be specified in a relevant server [option group](#) in an [option file](#) . For example:

```
[mariadb]  
...  
plugin_load_add = sql_errlog
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing [UNINSTALL SONAME](#) or [UNINSTALL PLUGIN](#) . For example:

```
UNINSTALL SONAME 'sql_errlog';
```

If you installed the plugin by providing the [--plugin-load](#) or the [--plugin-load-add](#) options in a relevant server [option group](#) in an [option file](#) , then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Example

```

install plugin SQL_ERROR_LOG soname 'sql_errlog';
Query OK, 0 rows affected (0.00 sec)

use test;
Database changed

set sql_mode='STRICT_ALL_TABLES,NO_ENGINE_SUBSTITUTION';
Query OK, 0 rows affected (0.00 sec)

CREATE TABLE foo2 (id int) ENGINE=WHOOPSIE;
ERROR 1286 (42000): Unknown storage engine 'WHOOPSIE'
\! cat data/sql_errors.log
2013-03-19  9:38:40 msandbox[msandbox] @ localhost [] ERROR 1286: Unknown storage engine 'WHOOPSIE' : CREATE TABLE foo2 (id int)

```

## Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.1.13
1.0	Gamma	MariaDB 10.0.10
1.0	Alpha	MariaDB 5.5.22

## System Variables

### sql\_error\_log\_filename

- **Description:** The name of the logfile. Rotation of it will be named like

*sql\_error\_log\_filename*

.001

- **Commandline:**

--sql-error-log-filename=value

- **Scope:** Global

- **Dynamic:** No

- **Data Type:**

string

- **Default Value:**

sql\_errors.log

### sql\_error\_log\_rate

- **Description:** The rate of logging.

SET sql\_error\_log\_rate=300;

means that one of 300 errors will be written to the log.

If

sql\_error\_log\_rate

is

0

the logging is disabled.

The default rate is

1

(every error is logged).

- **Commandline:**

--sql-error-log-rate=#

- **Scope:** Global

- **Dynamic:** Yes
- **Data Type:**  
    numeric

- **Default Value:**  
    1
- 

## sql\_error\_log\_rotate

- **Description:** This is the 'write-only' variable. Assigning TRUE to this variable forces the log rotation.
- **Commandline:**

```
--sql-error-log-rotate={0|1}
```

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
    boolean

- **Default Value:**  
    OFF
- 

## sql\_error\_log\_rotations

- **Description:** The number of rotations. When rotated, the current log file is stored and the new empty one created.  
The sql\_error\_log\_rotations logs are stored, older are removed.
- **Default Value:**  
    9

- **Commandline:**  
    --sql-error-log-rotations

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    numeric

- **Default Value:**  
    9

- **Range:**  
    1  
    to  
    999
- 

## sql\_error\_log\_size\_limit

- **Description:** The limitation for the size of the log file. After reaching the specified limit, the log file is rotated.  
1M limit set by default.

- **Commandline:**  
    --sql-error-log-size-limit=#

- **Scope:** Global
- **Dynamic:** No
- **Data Type:**  
    numeric

- **Default Value:**  
1000000

- **Range:**  
100  
to  
9223372036854775807

---

## Options

### sql\_error\_log

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF
  - Disables the plugin without removing it from the `mysql.plugins` table.
- ON
  - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
- FORCE
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
- FORCE\_PLUS\_PERMANENT
  - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with `UNINSTALL SONAME` or `UNINSTALL PLUGIN` while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

```
--sql-error-log=value
```

- **Data Type:**

enumerated

- **Default Value:**

ON

- **Valid Values:**

OFF  
,  
ON  
,  
FORCE  
,  
FORCE\_PLUS\_PERMANENT

---

## 4.4.10.7 User Statistics

The User Statistics feature was first released in [MariaDB 5.2.0](#), and moved to the

`userstat`  
plugin in [MariaDB 10.1.1](#).

The

`userstat`  
plugin creates the `USER_STATISTICS`, `CLIENT_STATISTICS`, the `INDEX_STATISTICS`, and the `TABLE_STATISTICS` tables in the `INFORMATION_SCHEMA` database. As an alternative to these tables, the plugin also adds the `SHOW USER_STATISTICS`, the `SHOW CLIENT_STATISTICS`, the `SHOW INDEX_STATISTICS`, and the `SHOW TABLE_STATISTICS` statements.

These tables and commands can be used to understand the server activity better and to identify the sources of your database's load.

The plugin also adds the `FLUSH USER_STATISTICS`, `FLUSH CLIENT_STATISTICS`, `FLUSH INDEX_STATISTICS`, and `FLUSH TABLE_STATISTICS` statements.

The MariaDB implementation of this plugin is based on the [userstatv2 patch](#) from Percona and OurDelta. The original code comes from Google (Mark

Callaghan's team) with additional work from Percona, Ourdelta, and Weldon Whipple. The MariaDB implementation provides the same functionality as the userstatv2 patch but a lot of changes have been made to make it faster and to better fit the MariaDB infrastructure.

## Contents

- 1. [How it Works](#)
- 2. [Enabling the Plugin](#)
- 3. [Using the Plugin](#)
  - 1. [Using the Information Schema Table](#)
  - 2. [Using the SHOW Statements](#)
  - 3. [Flushing Plugin Data](#)
- 4. [Versions](#)
  - 1. [USER\\_STATISTICS](#)
  - 2. [CLIENT\\_STATISTICS](#)
  - 3. [INDEX\\_STATISTICS](#)
  - 4. [TABLE\\_STATISTICS](#)
- 5. [System Variables](#)
  - 1. [userstat](#)

## How it Works

The

```
userstat
```

plugin works by keeping several hash tables in memory. All variables are incremented while the query is running. At the end of each statement the global values are updated.

## Enabling the Plugin

By default statistics are not collected. This is to ensure that statistics collection does not cause any extra load on the server unless desired.

Set the `userstat=ON` system variable in a relevant server [option group](#) in an [option file](#) to enable the plugin. For example:

```
[mariadb]
...
userstat = 1
```

The value can also be changed dynamically. For example:

```
SET GLOBAL userstat=1;
```

## Using the Plugin

### Using the Information Schema Table

The

```
userstat
```

plugin creates the `USER_STATISTICS`, `CLIENT_STATISTICS`, the `INDEX_STATISTICS`, and the `TABLE_STATISTICS` tables in the `INFORMATION_SCHEMA` database.

```

SELECT * FROM INFORMATION_SCHEMA.USER_STATISTICS\G
*****
***** 1. row *****
    USER: root
    TOTAL_CONNECTIONS: 1
    CONCURRENT_CONNECTIONS: 0
    CONNECTED_TIME: 297
        BUSY_TIME: 0.001725
        CPU_TIME: 0.001982
    BYTES RECEIVED: 388
    BYTES SENT: 2327
    BINLOG_BYTES_WRITTEN: 0
        ROWS_READ: 0
        ROWS_SENT: 12
        ROWS_DELETED: 0
    ROWS_INSERTED: 13
        ROWS_UPDATED: 0
    SELECT_COMMANDS: 4
    UPDATE_COMMANDS: 0
    OTHER_COMMANDS: 3
    COMMIT_TRANSACTIONS: 0
    ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
    LOST_CONNECTIONS: 0
    ACCESS_DENIED: 0
    EMPTY_QUERIES: 1

```

```

SELECT * FROM INFORMATION_SCHEMA.CLIENT_STATISTICS\G
*****
***** 1. row *****
    CLIENT: localhost
    TOTAL_CONNECTIONS: 3
    CONCURRENT_CONNECTIONS: 0
    CONNECTED_TIME: 4883
        BUSY_TIME: 0.009722
        CPU_TIME: 0.0102131
    BYTES RECEIVED: 841
    BYTES SENT: 13897
    BINLOG_BYTES_WRITTEN: 0
        ROWS_READ: 0
        ROWS_SENT: 214
        ROWS_DELETED: 0
    ROWS_INSERTED: 207
        ROWS_UPDATED: 0
    SELECT_COMMANDS: 10
    UPDATE_COMMANDS: 0
    OTHER_COMMANDS: 13
    COMMIT_TRANSACTIONS: 0
    ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
    LOST_CONNECTIONS: 0
    ACCESS_DENIED: 0
    EMPTY_QUERIES: 1
1 row in set (0.00 sec)

```

```

SELECT * FROM INFORMATION_SCHEMA.INDEX_STATISTICS WHERE TABLE_NAME = "author";
+-----+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME | INDEX_NAME | ROWS_READ |
+-----+-----+-----+-----+
| books       | author      | by_name     |      15 |
+-----+-----+-----+-----+

```

```

SELECT * FROM INFORMATION_SCHEMA.TABLE_STATISTICS WHERE TABLE_NAME='user';
+-----+-----+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME | ROWS_READ | ROWS_CHANGED | ROWS_CHANGED_X_INDEXES |
+-----+-----+-----+-----+
| mysql       | user        |      5 |          2 |                  2 |
+-----+-----+-----+-----+

```

## Using the SHOW Statements

As an alternative to the [INFORMATION\\_SCHEMA](#) tables, the

`userstat`

plugin also adds the [SHOW USER\\_STATISTICS](#), the [SHOW CLIENT\\_STATISTICS](#), the [SHOW INDEX\\_STATISTICS](#), and the [SHOW](#)

[TABLE\\_STATISTICS](#) statements.

These commands are another way to display the information stored in the information schema tables. WHERE clauses are accepted. LIKE clauses are accepted but ignored.

```
SHOW USER_STATISTICS  
SHOW CLIENT_STATISTICS  
SHOW INDEX_STATISTICS  
SHOW TABLE_STATISTICS
```

## Flushing Plugin Data

The `userstat` plugin also adds the [FLUSH USER\\_STATISTICS](#), [FLUSH CLIENT\\_STATISTICS](#), [FLUSH INDEX\\_STATISTICS](#), and [FLUSH TABLE\\_STATISTICS](#) statements, which discard the information stored in the specified information schema table.

```
FLUSH USER_STATISTICS  
FLUSH CLIENT_STATISTICS  
FLUSH INDEX_STATISTICS  
FLUSH TABLE_STATISTICS
```

## Versions

### USER\_STATISTICS

Version	Status	Introduced
2.0	Stable	MariaDB 10.1.18
2.0	Gamma	MariaDB 10.1.1

### CLIENT\_STATISTICS

Version	Status	Introduced
2.0	Stable	MariaDB 10.1.13
2.0	Gamma	MariaDB 10.1.1

### INDEX\_STATISTICS

Version	Status	Introduced
2.0	Stable	MariaDB 10.1.13
2.0	Gamma	MariaDB 10.1.1

### TABLE\_STATISTICS

Version	Status	Introduced
2.0	Stable	MariaDB 10.1.18
2.0	Gamma	MariaDB 10.1.1

## System Variables

### userstat

- **Description:** If set to

1  
, `user statistics` will be activated.

- **Commandline:**

--userstat=1

- **Scope:** Global
- **Dynamic:** Yes
- **Data Type:**  
boolean

- **Default Value:**  
OFF

## 4.4.10.8 User Variables Plugin

MariaDB starting with [10.2.0](#)

The

`user_variables`  
plugin was first released in MariaDB [10.2.0](#).

The

`user_variables`  
plugin creates the

[USER\\_VARIABLES](#)

table in the

[INFORMATION\\_SCHEMA](#)

database. This table contains information about [user-defined variables](#).

### Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Example](#)
4. [Versions](#)
5. [Options](#)
  1. [user\\_variables](#)

## Installing the Plugin

MariaDB starting with [10.2.6](#)

In [MariaDB 10.2.6](#) and later, the

`user_variables`  
plugin is statically linked into the server by default, so it does not need to be installed.

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default prior to [MariaDB 10.2.6](#). There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

[INSTALL SONAME](#)

or

[INSTALL PLUGIN](#)

. For example:

```
INSTALL SONAME 'user_variables';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

[--plugin-load](#)

or the

[--plugin-load-add](#)

options. This can be specified as a command-line argument to

```
mysql
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = user_variables
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'user_variables';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

## Example

```
SELECT * FROM information_schema.USER_VARIABLES ORDER BY VARIABLE_NAME;
+-----+-----+-----+
| VARIABLE_NAME | VARIABLE_VALUE | VARIABLE_TYPE | CHARACTER_SET_NAME |
+-----+-----+-----+
| var          | 0           | INT          | utf8            |
| var2         | abc         | VARCHAR      | utf8            |
+-----+-----+-----+
```

## Versions

Version	Status	Introduced
1.0	Stable	MariaDB 10.3.13
1.0	Gamma	MariaDB 10.2.6
1.0	Alpha	MariaDB 10.2.0

## Options

### user\_variables

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- 

OFF

- Disables the plugin without removing it from the `mysql.plugins` table.
- - `ON`
    - Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.
  - - `FORCE`
      - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.
    - - `FORCE_PLUS_PERMANENT`
        - Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

`UNINSTALL SONAME`

or

`UNINSTALL PLUGIN`

while the server is running.

  - See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.
  - **Commandline:**
  - `--user-variables=value`
  - **Data Type:**
  - enumerated
  - **Default Value:**
  - `ON`
  - **Valid Values:**
  - `OFF`
  - `,`
  - `ON`
  - `,`
  - `FORCE`
  - `,`
  - `FORCE_PLUS_PERMANENT`

## 4.4.10.9 Disks Plugin

MariaDB starting with [10.1.32](#)

The `DISKS` plugin was first released in [MariaDB 10.3.6](#) , [MariaDB 10.2.14](#) and [MariaDB 10.1.32](#) .

The `DISKS` plugin creates the

`DISKS`

table in the

`INFORMATION_SCHEMA`

database. This table shows metadata about disks on the system.

Before [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#) and [MariaDB 10.1.41](#) , this plugin did **not** check `user privileges` . When it is enabled, **any** user can query the

`INFORMATION_SCHEMA.DISKS` table and see all the information it provides.

Since [MariaDB 10.4.7](#) , [MariaDB 10.3.17](#) , [MariaDB 10.2.26](#) and [MariaDB 10.1.41](#) , it requires the `FILE privilege` .

The plugin only works on Linux.

## Contents

1. [Installing the Plugin](#)
2. [Uninstalling the Plugin](#)
3. [Example](#)
4. [Versions](#)
5. [Options](#)
  1. [disks](#)

## Installing the Plugin

Although the plugin's shared library is distributed with MariaDB by default, the plugin is not actually installed by MariaDB by default. There are two methods that can be used to install the plugin with MariaDB.

The first method can be used to install the plugin without restarting the server. You can install the plugin dynamically by executing

```
INSTALL SONAME
```

or

```
INSTALL PLUGIN
```

. For example:

```
INSTALL SONAME 'disks';
```

The second method can be used to tell the server to load the plugin when it starts up. The plugin can be installed this way by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options. This can be specified as a command-line argument to

```
mysqld
```

or it can be specified in a relevant server [option group](#) in an [option file](#). For example:

```
[mariadb]
...
plugin_load_add = disks
```

## Uninstalling the Plugin

You can uninstall the plugin dynamically by executing

```
UNINSTALL SONAME
```

or

```
UNINSTALL PLUGIN
```

. For example:

```
UNINSTALL SONAME 'disks';
```

If you installed the plugin by providing the

```
--plugin-load
```

or the

```
--plugin-load-add
```

options in a relevant server [option group](#) in an [option file](#), then those options should be removed to prevent the plugin from being loaded the next time the server is restarted.

# Example

```
SELECT * FROM information_schema.DISKS;
```

Disk	Path	Total	Used	Available
/dev/vda1	/	26203116	2178424	24024692
/dev/vda1	/boot	26203116	2178424	24024692
/dev/vda1	/etc	26203116	2178424	24024692

## Versions

Version	Status	Introduced
1.1	Stable	MariaDB 10.4.7 , MariaDB 10.3.17 , MariaDB 10.2.26 , MariaDB 10.1.41
1.0	Beta	MariaDB 10.3.6 , MariaDB 10.2.14 , MariaDB 10.1.32

## Options

### disks

- **Description:** Controls how the server should treat the plugin when the server starts up.

- Valid values are:

- OFF

- Disables the plugin without removing it from the

- `mysql.plugins`

- table.

- ON

- Enables the plugin. If the plugin cannot be initialized, then the server will still continue starting up, but the plugin will be disabled.

- FORCE

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error.

- FORCE\_PLUS\_PERMANENT

- Enables the plugin. If the plugin cannot be initialized, then the server will fail to start with an error. In addition, the plugin cannot be uninstalled with

- `UNINSTALL SONAME`

- or

- `UNINSTALL PLUGIN`

- while the server is running.

- See [Plugin Overview: Configuring Plugin Activation at Server Startup](#) for more information.

- **Commandline:**

- `--disks=value`

- **Data Type:**

- enumerated

- **Default Value:**

- ON

- **Valid Values:**

- OFF

- ,

- ON

```
,  
FORCE  
,
```

## 4.4.10.10 Compression Plugins

MariaDB starting with [10.7.0](#)

Compressions plugins were added in a [MariaDB 10.7.0](#) preview release.

### Contents

1. [Installing](#)
2. [Upgrading](#)
3. [See Also](#)

The various MariaDB storage engines, such as [InnoDB](#) , [RocksDB](#) , [Mroonga](#) , can use different compression libraries.

Before [MariaDB 10.7.0](#) , each separate library would have to be compiled in in order to be available for use, resulting in numerous runtime/rpm/deb dependencies, most of which would never be used by users.

From [MariaDB 10.7.0](#) , five additional MariaDB compression libraries (besides the default zlib) are available as plugins (note that these affect InnoDB and Mroonga only; RocksDB still uses the compression algorithms from its own library):

- bzip2
- lzma
- lz4
- lzo
- snappy

## Installing

To use, these simply need to be [installed as a plugin](#) :

```
INSTALL SONAME 'provider_lz4';
```

The compression algorithm can then be used, for example, in [InnoDB compression](#) :

```
SET GLOBAL innodb_compression_algorithm = lz4;
```

## Upgrading

When upgrading from a release without compression plugins, if a non-zlib compression algorithm was used, those tables will be unreadable until the appropriate compression library is installed. [mariadb-upgrade](#) should be run. The

```
--force  
option (to run mariadb-check ) or  
mariadb-check  
itself will indicate any problems with compression, for example:
```

```
Warning : MariaDB tried to use the LZMA compression, but its provider plugin is not loaded  
  
Error   : Table 'test.t' doesn't exist in engine  
  
status  : Operation failed
```

or

```
Error   : Table test/t is compressed with lzma, which is not currently loaded.  
Please load the lzma provider plugin to open the table  
  
error   : Corrupt
```

In this case, the appropriate compression plugin should be installed, and the server restarted.

## See Also

- 10.7 preview feature: Compression Provider Plugins (mariadb.org blog)

# 5 MariaDB Community Bug Reporting

## Contents

1. Known Bugs
2. Reporting a Bug
  1. JIRA Privacy
  2. Reporting Security Vulnerabilities
  3. Contents of a Good Bug Report
  4. JIRA Fields
    1. Project
    2. Type
    3. Summary
    4. Priority
    5. Affected Versions
    6. Environment
    7. Description
    8. Attachments
    9. Links
    10. Tags
  5. Bugs that also Affect MySQL or XtraDB/TokuDB in Percona
3. Collecting Additional Information for a Bug Report
  1. Getting a Stack Trace with Details
  2. Extracting a Portion of a Binary Log
4. Getting Help with your Servers

MariaDB's bug and feature tracker is found at <https://jira.mariadb.org>.

This page contains general guidelines for the community for reporting bugs in MariaDB products. If you want to discuss a problem or a new feature with other MariaDB developers, you can find the email lists and forums [here](#).

## Known Bugs

First, check that the bug isn't already filed in the [MariaDB bugs database](#).

For the MariaDB bugs database, use JIRA search to check if a report you are going to submit already exists. You are not expected to be a JIRA search guru, but please at least make some effort.

- Choose
  - Issues => Search for issues
  - ;
- If the form opens for you with a long blank line at top, press
  - Basic
  - on the right to switch to a simpler mode;
- In the
  - Project
  - field, choose the related project, (
  - MDEV
  - for generic MariaDB server and clients);
- In the
  - Contains text
  - text field, enter the most significant key words from your future report;
- Press
  - Enter
  - or the magnifying glass icon to search.

If you see bug reports which are already closed, pay attention to the 'Fix version/s' field -- it is possible that they were fixed in the *upcoming* release. If they are said to be fixed in the release that you are currently using or earlier, you can ignore them and file a new one (although please mention in your bug report that you found them, it might be useful).

If you find an open bug report, please vote/add a comment that the bug also affects you along with any additional information you have that may help us to find and fix the bug.

If the bug is not in the MariaDB bugs database yet, then it's time to file a bug report. If you're filing a bug report about a bug that's already in the [MySQL bugs database](#), please indicate so at the start of the report. Filing bug reports from MySQL in the MariaDB bugs database makes sense, because:

- It shows the MariaDB team that there is interest in having this bug fixed in MariaDB.
- It allows work to start on fixing the bug in MariaDB - assigning versions, assigning MariaDB developers to the bug, etc.

## Reporting a Bug

## JIRA Privacy

Please note that our JIRA entries are public, and JIRA is very good at keeping a record of everything that has been done. What this means is that if you ever include confidential information in the description there will be a log containing it, even after you've deleted it. The only way to get rid of it will be removing the JIRA entry completely.

Attachments in JIRA are also public.

Access to a comment can be restricted to a certain group (e.g. Developers only), but the existing groups are rather wide, so you should not rely on it either.

If you have private information -- SQL fragments, logs, database dumps, etc. -- that you are willing to share with MariaDB team, but not with the entire world, put it into a file, compress if necessary, upload to the [mariadb-ftp-server](#), and just mention it in the JIRA description. This way only the MariaDB team will have access to it.

## Reporting Security Vulnerabilities

As explained above, all JIRA issues are public. If you believe you have found a security vulnerability, send an email to [security@mariadb.org](mailto:security@mariadb.org), please, do not use JIRA for that. We will enter it in JIRA ourselves, following the [responsible disclosure](#) practices.

## Contents of a Good Bug Report

Below is the information we need to be able to fix bugs. The more information we get and the easier we can repeat the bug, the faster it will be fixed.

A good bug report consists of:

- a. The environment (Operating system, hardware and MariaDB version) where the bug happened.

- b. Any related errors or warnings from the server error log file. Normally it is

```
hostname.err  
file in your database directory, but it can be different depending on the distribution and version; if you cannot find it, run
```

```
SELECT @@log_error
```

on the running server. If either the variable or the file it points at is empty, the error log most likely goes to your system log. If this is  
systemd you can get the last 50 lines of the MariaDB log with

```
journalctl -n 50 -u mariadb.service
```

. If possible, attach the full unabridged error log at least from the last server restart and till the end of the log.,

- c. If the problem is related to MariaDB updates, or otherwise changing the version of the server, recovery from a previous crash, and such, then  
include the previous versions used, and the error log from previous server sessions.

- d. The content of your my.cnf file or alternatively the output from

```
mysqld --print-defaults
```

or

```
SHOW VARIABLES
```

- e. Any background information you can provide ( [stack trace](#) , tables, table definitions (

```
show-create-table SHOW CREATE TABLE {tablename}
```

), data dumps, query logs).

- f. If the bug is about server producing wrong query results: the actual result (what you are getting), the expected result (what you think should be produced instead), and, unless it is obvious, the reason why you think the current result is wrong.

- g. If the bug about a performance problem, e.g. a certain query is slower on one version than on another, output of

```
EXPLAIN EXTENDED <query>
```

on both servers. If its a

```
SELECT
```

```
query use analyze-format-json ANALYZE FORMAT=JSON .
```

- h. A test case or some other way to repeat the bug. This should preferably be in plain SQL or in mysqltest format. See [mysqltest/README](#) for  
information about this.

- i. If it's impossible to do a test case, then providing us with a [core dump + the corresponding binary](#) would be of great help.

## JIRA Fields

The section below describes which JIRA fields need to be populated while filing reports, and what should be put there. Apart from what's mentioned below, you don't have to fill or change any fields while creating a new bug report.

### Project

If you are filing a report for MariaDB server, client programs, or MariaDB Galera cluster, the target project is

```
MDEV
```

. Connectors and MaxScale have separate projects with corresponding names. If you choose a wrong project, bug processing can be delayed,  
but there is no reason to panic -- we'll correct it. If you inform us about the mistake, we'll change it faster.

Some project names include:

- CONC - MariaDB Connector/C
- CONJ - MariaDB Connector/J
- CONJS - MariaDB Connector/node.js
- CONPY - MariaDB Connector/Python
- MCOL - ColumnStore
- MDEV - MariaDB server, client programs, or MariaDB Galera Cluster
- MXS - MaxScale
- ODBC - MariaDB Connector/ODBC

## Type

Feature requests are not the same as bug reports. Specify a

Task

type for feature requests in [Jira](#), and a

Bug

type for bug reports. Like with the project field, choosing a wrong type will put the request to the wrong queue and can delay its processing, but eventually it will be noticed and amended.

See also [plans for next release](#) for things that we are considering to have in the next MariaDB release.

## Summary

Please make sure the summary line is informative and distinctive. It should always be easy to recognize your report among other similar ones, otherwise a reasonable question arises -- why are they not duplicates?

Examples:

- good summary: *Server crash with insert statement containing DEFAULT into view*
- not a good summary: *mysqld crash*

Generally, we try not to change the original summary without a good reason to do it, so that you can always recognize your own reports easily.

## Priority

We do not have separate Severity/Priority fields in JIRA, so this Priority field serves a double purpose. For original reports, it indicates the importance of the problem from the reporter's point of view. The default is 'Major'; there are two lower and two higher values. Please set the value accurately. While we do take it into account during initial processing, increasing the value above reasonable won't do any good, the only effect will be the waste of time while somebody will be trying to understand why a trivial problem got such a high priority. After that, the value will be changed, and the report will be processed in its due time anyway.

## Affected Versions

Put everything you know about which versions are affected. There are both major versions (10.6, 10.5 etc.) and minor versions (10.5.9, 10.4.12, etc.) available for choosing. Please always specify there the exact version(s) (X.Y.Z) which you are working with, and where you experience the problem.

Additionally, If you know the exact version where the problem appeared, please put it as well. If the problem has been present, as far as you know, in all previous releases, you can also put there the major version, e.g. 10.0. Alternatively, you can mention all of it in the description or comments.

Please also note in the description or comments which versions you know as *not* affected. This information will help to shorten further processing.

## Environment

Put here environment-related information that might be important for reproducing or analyzing the problem: operating system, hardware, related 3rd-party applications, compilers, etc.

## Description

The most important part of the description are steps to reproduce the problem. See more details about bug report contents above in the section [Contents of a good bug report](#).

If in the process of reproducing, you executed some SQL, don't describe it in words such as "I created a table with text columns and date columns and populated it with some rows" -- instead, whenever possible, put the exact SQL queries that you ran. The same goes for problems that you encountered: instead of saying "it did not work, the query failed, I got an error", always paste the exact output that you received.

*Use {noformat}...{noformat} and {code}...{code} blocks for code and console output in the description.*

## Attachments

If you have SQL code, a database dump, a log etc. of a reasonable size, attach them to the report (archive them first if necessary). If they are too big, you can upload them to [ftp.askmonty.org/private](http://ftp.askmonty.org/private). It is always a good idea to attach your cnf file(s), unless it is absolutely clear from the nature of the report that configuration is irrelevant.

## Links

If you found or filed a bug report either in MariaDB or MySQL or Percona bug base which you think is related to yours, you can put them in the

## Links

section; same for any external links to 3rd-party resources which you find important to mention. Alternatively, you can just mention them in the description or comments.

## Tags

You don't have to set any tags, but if you want to use any for your convenience, feel free to do so. However, please don't put too generic values -- for example, the tag

```
mariadb  
is meaningless, because everything there is  
mariadb  
. Don't be surprised if some tags are removed later during report processing.
```

## Bugs that also Affect MySQL or XtraDB/TokuDB in Percona

Our normal practice is to report a bug upstream if it's applicable to their version. While we can do it on your behalf, it is always better if you do it yourself - it will be easier for you to track it further.

If the bug affects MySQL, it should also be reported at [MySQL bugs database](#). If the bug affects XtraDB or TokuDB and reproducible with Percona server, it should go to [Percona Launchpad](#).

## Collecting Additional Information for a Bug Report

### Getting a Stack Trace with Details

See the article [How to produce a stack trace from a core file](#).

### Extracting a Portion of a Binary Log

See the article [here](#).

## Getting Help with your Servers

If you require personalized assistance, want to ensure that the bug is fixed with high priority, or want someone to login to your server to find out what's wrong, you can always purchase a [Support](#) contract from MariaDB Corporation or use their consulting services.