# SQL表达式

This chapter describes the syntax for the [SQL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_sql) statements supported by MySQL.

## 13.1 数据定义语句

### 13.1.1 Atomic Data Definition Statement Support

MySQL 8.0支持原子数据定义语言（DDL）语句。这个功能被称为原子DDL。原子DDL语句将与DDL操作相关的数据字典更新、存储引擎操作和二进制日志写入合并为一个单一的原子操作。该操作要么被提交，并将适用的变化持久化到数据字典、存储引擎和二进制日志，要么被回滚，即使服务器在操作过程中停止。

注意

原子DDL不是事务性DDL。DDL 语句，不管是原子的还是其他的，都隐含地结束了当前会话中的任何事务，就像你在执行该语句之前做了一个 COMMIT。这意味着DDL语句不能在另一个事务中，在事务控制语句中执行，如START TRANSACTION ... COMMIT，或者与同一事务中的其他语句相结合。

原子DDL是通过在MySQL 8.0中引入MySQL数据字典而实现的。在早期的MySQL版本中，元数据被存储在元数据文件、非事务性表和特定于存储引擎的字典中，这就需要进行中间提交。由MySQL数据字典提供的集中的、事务性的元数据存储消除了这一障碍，使得重组DDL语句操作成为可能，成为原子操作。The atomic DDL feature is described under the following topics in this section:

[Supported DDL Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl-supported-statements)

[Atomic DDL Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl-characteristics)

[Changes in DDL Statement Behavior](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl-statement-behavior)

[Storage Engine Support](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl-storage-engine-support)

[Viewing DDL Logs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl-view-logs)

#### Supported DDL Statements

The atomic DDL feature supports both table and non-table DDL statements. Table-related DDL operations require storage engine support, whereas non-table DDL operations do not. Currently, only the **InnoDB** storage engine supports atomic DDL.

Supported table DDL statements include **CREATE**, **ALTER**, and **DROP** statements for databases, tablespaces, tables, and indexes, and the [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement.

Supported non-table DDL statements include:

**CREATE** and **DROP** statements, and, if applicable, **ALTER** statements for stored programs, triggers, views, and user-defined functions (UDFs).

Account management statements: **CREATE**, **ALTER**, **DROP**, and, if applicable, **RENAME** statements for users and roles, as well as [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) and [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) statements.

The following statements are not supported by the atomic DDL feature:

Table-related DDL statements that involve a storage engine other than **InnoDB**.

[**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) and [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) statements.

[**INSTALL COMPONENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-component) and [**UNINSTALL COMPONENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-component) statements.

[**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server), [**ALTER SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-server), and [**DROP SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-server) statements.

#### Atomic DDL Characteristics

The characteristics of atomic DDL statements include the following:

Metadata updates, binary log writes, and storage engine operations, where applicable, are combined into a single atomic operation.

There are no intermediate commits at the SQL layer during the DDL operation.

Where applicable:

The state of data dictionary, routine, event, and UDF caches is consistent with the status of the DDL operation, meaning that caches are updated to reflect whether or not the DDL operation was completed successfully or rolled back.

The storage engine methods involved in a DDL operation do not perform intermediate commits, and the storage engine registers itself as part of the DDL operation.

The storage engine supports redo and rollback of DDL operations, which is performed in the Post-DDL phase of the DDL operation.

The visible behaviour of DDL operations is atomic, which changes the behavior of some DDL statements. See [Changes in DDL Statement Behavior](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl-statement-behavior).

#### Changes in DDL Statement Behavior

This section describes changes in DDL statement behavior due to the introduction of atomic DDL support.

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) operations are fully atomic if all named tables use an atomic DDL-supported storage engine. The statement either drops all tables successfully or is rolled back.

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) fails with an error if a named table does not exist, and no changes are made, regardless of the storage engine. This change in behavior is demonstrated in the following example, where the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement fails because a named table does not exist:

mysql> **CREATE TABLE t1 (c1 INT);**

mysql> **DROP TABLE t1, t2;**

ERROR 1051 (42S02): Unknown table 'test.t2'

mysql> **SHOW TABLES;**

+----------------+

| Tables\_in\_test |

+----------------+

| t1 |

+----------------+

Prior to the introduction of atomic DDL, [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) reports an error for the named table that does not exist but succeeds for the named table that does exist:

mysql> **CREATE TABLE t1 (c1 INT);**

mysql> **DROP TABLE t1, t2;**

ERROR 1051 (42S02): Unknown table 'test.t2'

mysql> **SHOW TABLES;**

Empty set (0.00 sec)

**Note**

Due to this change in behavior, a partially completed [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement on a MySQL 5.7 replication source server fails when replicated on a MySQL 8.0 replica. To avoid this failure scenario, use **IF EXISTS** syntax in [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statements to prevent errors from occurring for tables that do not exist.

[**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) is atomic if all tables use an atomic DDL-supported storage engine. The statement either drops all objects successfully or is rolled back. However, removal of the database directory from the file system occurs last and is not part of the atomic operation. If removal of the database directory fails due to a file system error or server halt, the [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) transaction is not rolled back.

For tables that do not use an atomic DDL-supported storage engine, table deletion occurs outside of the atomic [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) or [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) transaction. Such table deletions are written to the binary log individually, which limits the discrepancy between the storage engine, data dictionary, and binary log to one table at most in the case of an interrupted [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) or [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) operation. For operations that drop multiple tables, the tables that do not use an atomic DDL-supported storage engine are dropped before tables that do.

[**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), [**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table), [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table), [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace), and [**DROP TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-tablespace) operations for tables that use an atomic DDL-supported storage engine are either fully committed or rolled back if the server halts during their operation. In earlier MySQL releases, interruption of these operations could cause discrepancies between the storage engine, data dictionary, and binary log, or leave behind orphan files. [**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table) operations are only atomic if all named tables use an atomic DDL-supported storage engine.

As of MySQL 8.0.21, on storage engines that support atomic DDL, the [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statement is logged as one transaction in the binary log when row-based replication is in use. Previously, it was logged as two transactions, one to create the table, and the other to insert data. A server failure between the two transactions or while inserting data could result in replication of an empty table. With the introduction of atomic DDL support, [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statements are now safe for row-based replication and permitted for use with GTID-based replication.

On storage engines that support both atomic DDL and foreign key constraints, creation of foreign keys is not permitted in [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statements when row-based replication is in use. Foreign key constraints can be added later using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

When [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) is applied as an atomic operation, a metadata lock is held on the table while data is inserted, which prevents concurrent access to the table for the duration of the operation.

[**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) fails if a named view does not exist, and no changes are made. The change in behavior is demonstrated in this example, where the [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) statement fails because a named view does not exist:

mysql> **CREATE VIEW test.viewA AS SELECT \* FROM t;**

mysql> **DROP VIEW test.viewA, test.viewB;**

ERROR 1051 (42S02): Unknown table 'test.viewB'

mysql> **SHOW FULL TABLES IN test WHERE TABLE\_TYPE LIKE 'VIEW';**

+----------------+------------+

| Tables\_in\_test | Table\_type |

+----------------+------------+

| viewA | VIEW |

+----------------+------------+

Prior to the introduction of atomic DDL, [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) returns an error for the named view that does not exist but succeeds for the named view that does exist:

mysql> **CREATE VIEW test.viewA AS SELECT \* FROM t;**

mysql> **DROP VIEW test.viewA, test.viewB;**

ERROR 1051 (42S02): Unknown table 'test.viewB'

mysql> **SHOW FULL TABLES IN test WHERE TABLE\_TYPE LIKE 'VIEW';**

Empty set (0.00 sec)

**Note**

Due to this change in behavior, a partially completed [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) operation on a MySQL 5.7 replication source server fails when replicated on a MySQL 8.0 replica. To avoid this failure scenario, use **IF EXISTS** syntax in [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) statements to prevent an error from occurring for views that do not exist.

Partial execution of account management statements is no longer permitted. Account management statements either succeed for all named users or roll back and have no effect if an error occurs. In earlier MySQL versions, account management statements that name multiple users could succeed for some users and fail for others.

The change in behavior is demonstrated in this example, where the second [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) statement returns an error but fails because it cannot succeed for all named users.

mysql> **CREATE USER userA;**

mysql> **CREATE USER userA, userB;**

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

mysql> **SELECT User FROM mysql.user WHERE User LIKE 'user%';**

+-------+

| User |

+-------+

| userA |

+-------+

Prior to the introduction of atomic DDL, the second **CREATE USER** statement returns an error for the named user that does not exist but succeeds for the named user that does exist:

mysql> **CREATE USER userA;**

mysql> **CREATE USER userA, userB;**

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

mysql> **SELECT User FROM mysql.user WHERE User LIKE 'user%';**

+-------+

| User |

+-------+

| userA |

| userB |

+-------+

**Note**

Due to this change in behavior, partially completed account management statements on a MySQL 5.7 replication source server fail when replicated on a MySQL 8.0 replica. To avoid this failure scenario, use **IF EXISTS** or **IF NOT EXISTS** syntax, as appropriate, in account management statements to prevent errors related to named users.

#### Storage Engine Support

Currently, only the **InnoDB** storage engine supports atomic DDL. Storage engines that do not support atomic DDL are exempted from DDL atomicity. DDL operations involving exempted storage engines remain capable of introducing inconsistencies that can occur when operations are interrupted or only partially completed.

To support redo and rollback of DDL operations, **InnoDB** writes DDL logs to the **mysql.innodb\_ddl\_log** table, which is a hidden data dictionary table that resides in the mysql.ibd data dictionary tablespace.

To view DDL logs that are written to the **mysql.innodb\_ddl\_log** table during a DDL operation, enable the [**innodb\_print\_ddl\_logs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_ddl_logs) configuration option. For more information, see [Viewing DDL Logs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl-view-logs).

**Note**

The redo logs for changes to the **mysql.innodb\_ddl\_log** table are flushed to disk immediately regardless of the [**innodb\_flush\_log\_at\_trx\_commit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_flush_log_at_trx_commit) setting. Flushing the redo logs immediately avoids situations where data files are modified by DDL operations but the redo logs for changes to the **mysql.innodb\_ddl\_log** table resulting from those operations are not persisted to disk. Such a situation could cause errors during rollback or recovery.

The **InnoDB** storage engine executes DDL operations in phases. DDL operations such as [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) may perform the Prepare and Perform phases multiple times prior to the Commit phase.

Prepare: Create the required objects and write the DDL logs to the **mysql.innodb\_ddl\_log** table. The DDL logs define how to roll forward and roll back the DDL operation.

Perform: Perform the DDL operation. For example, perform a create routine for a **CREATE TABLE** operation.

Commit: Update the data dictionary and commit the data dictionary transaction.

Post-DDL: Replay and remove DDL logs from the **mysql.innodb\_ddl\_log** table. To ensure that rollback can be performed safely without introducing inconsistencies, file operations such as renaming or removing data files are performed in this final phase. This phase also removes dynamic metadata from the **mysql.innodb\_dynamic\_metadata** data dictionary table for [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table), [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table), and other DDL operations that rebuild the table.

DDL logs are replayed and removed from the **mysql.innodb\_ddl\_log** table during the Post-DDL phase, regardless of whether the DDL operation is committed or rolled back. DDL logs should only remain in the **mysql.innodb\_ddl\_log** table if the server is halted during a DDL operation. In this case, the DDL logs are replayed and removed after recovery.

In a recovery situation, a DDL operation may be committed or rolled back when the server is restarted. If the data dictionary transaction that was performed during the Commit phase of a DDL operation is present in the redo log and binary log, the operation is considered successful and is rolled forward. Otherwise, the incomplete data dictionary transaction is rolled back when **InnoDB** replays data dictionary redo logs, and the DDL operation is rolled back.

#### Viewing DDL Logs

To view DDL logs that are written to the **mysql.innodb\_ddl\_log** data dictionary table during atomic DDL operations that involve the **InnoDB** storage engine, enable [**innodb\_print\_ddl\_logs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_ddl_logs) to have MySQL write the DDL logs to **stderr**. Depending on the host operating system and MySQL configuration, **stderr** may be the error log, terminal, or console window. See [Section 5.4.2.2, “Default Error Log Destination Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#error-log-destination-configuration).

**InnoDB** writes DDL logs to the **mysql.innodb\_ddl\_log** table to support redo and rollback of DDL operations. The **mysql.innodb\_ddl\_log** table is a hidden data dictionary table that resides in the mysql.ibd data dictionary tablespace. Like other hidden data dictionary tables, the **mysql.innodb\_ddl\_log** table cannot be accessed directly in non-debug versions of MySQL. (See [Section 14.1, “Data Dictionary Schema”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-dictionary.html#data-dictionary-schema).) The structure of the **mysql.innodb\_ddl\_log** table corresponds to this definition:

CREATE TABLE mysql.innodb\_ddl\_log (

id BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,

thread\_id BIGINT UNSIGNED NOT NULL,

type INT UNSIGNED NOT NULL,

space\_id INT UNSIGNED,

page\_no INT UNSIGNED,

index\_id BIGINT UNSIGNED,

table\_id BIGINT UNSIGNED,

old\_file\_path VARCHAR(512) COLLATE UTF8\_BIN,

new\_file\_path VARCHAR(512) COLLATE UTF8\_BIN,

KEY(thread\_id)

);

**id**: A unique identifier for a DDL log record.

**thread\_id**: Each DDL log record is assigned a **thread\_id**, which is used to replay and remove DDL logs that belong to a particular DDL operation. DDL operations that involve multiple data file operations generate multiple DDL log records.

**type**: The DDL operation type. Types include **FREE** (drop an index tree), **DELETE** (delete a file), **RENAME** (rename a file), or **DROP** (drop metadata from the **mysql.innodb\_dynamic\_metadata** data dictionary table).

**space\_id**: The tablespace ID.

**page\_no**: A page that contains allocation information; an index tree root page, for example.

**index\_id**: The index ID.

**table\_id**: The table ID.

**old\_file\_path**: The old tablespace file path. Used by DDL operations that create or drop tablespace files; also used by DDL operations that rename a tablespace.

**new\_file\_path**: The new tablespace file path. Used by DDL operations that rename tablespace files.

This example demonstrates enabling [**innodb\_print\_ddl\_logs**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_print_ddl_logs) to view DDL logs written to **strderr** for a **CREATE TABLE** operation.

mysql> SET GLOBAL innodb\_print\_ddl\_logs=1;

mysql> CREATE TABLE t1 (c1 INT) ENGINE = InnoDB;

[Note] [000000] InnoDB: DDL log insert : [DDL record: DELETE SPACE, id=18, thread\_id=7,

space\_id=5, old\_file\_path=./test/t1.ibd]

[Note] [000000] InnoDB: DDL log delete : by id 18

[Note] [000000] InnoDB: DDL log insert : [DDL record: REMOVE CACHE, id=19, thread\_id=7,

table\_id=1058, new\_file\_path=test/t1]

[Note] [000000] InnoDB: DDL log delete : by id 19

[Note] [000000] InnoDB: DDL log insert : [DDL record: FREE, id=20, thread\_id=7,

space\_id=5, index\_id=132, page\_no=4]

[Note] [000000] InnoDB: DDL log delete : by id 20

[Note] [000000] InnoDB: DDL log post ddl : begin for thread id : 7

[Note] [000000] InnoDB: DDL log post ddl : end for thread id : 7

### 13.1.2 ALTER DATABASE Statement

ALTER {DATABASE | SCHEMA} [***db\_name***]

***alter\_option*** ...

***alter\_option***: {

[DEFAULT] CHARACTER SET [=] ***charset\_name***

| [DEFAULT] COLLATE [=] ***collation\_name***

| [DEFAULT] ENCRYPTION [=] {'Y' | 'N'}

| READ ONLY [=] {DEFAULT | 0 | 1}

}

[**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) enables you to change the overall characteristics of a database. These characteristics are stored in the data dictionary. This statement requires the [**ALTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter) privilege on the database. [**ALTER SCHEMA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) is a synonym for [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database).

If the database name is omitted, the statement applies to the default database. In that case, an error occurs if there is no default database.

For any ***alter\_option*** omitted from the statement, the database retains its current option value, with the exception that changing the character set may change the collation and vice versa.

[Character Set and Collation Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database-charset)

[Encryption Option](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database-encryption)

[Read Only Option](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database-read-only)

#### Character Set and Collation Options

The **CHARACTER SET** option changes the default database character set. The **COLLATE** option changes the default database collation. For information about character set and collation names, see [Chapter 10, *Character Sets, Collations, Unicode*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html).

To see the available character sets and collations, use the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) and [**SHOW COLLATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) statements, respectively. See [Section 13.7.7.3, “SHOW CHARACTER SET Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set), and [Section 13.7.7.4, “SHOW COLLATION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation).

A stored routine that uses the database defaults when the routine is created includes those defaults as part of its definition. (In a stored routine, variables with character data types use the database defaults if the character set or collation are not specified explicitly. See [Section 13.1.17, “CREATE PROCEDURE and CREATE FUNCTION Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure).) If you change the default character set or collation for a database, any stored routines that are to use the new defaults must be dropped and recreated.

#### Encryption Option

The **ENCRYPTION** option, introduced in MySQL 8.0.16, defines the default database encryption, which is inherited by tables created in the database. The permitted values are **'Y'** (encryption enabled) and **'N'** (encryption disabled). Only newly created tables inherit the default database encryption. For existing tables associated with the database, their encryption remains unchanged. If the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) system variable is enabled, the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege is required to specify a default encryption setting that differs from the value of the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) system variable. For more information, see [Defining an Encryption Default for Schemas and General Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-schema-tablespace-encryption-default).

#### Read Only Option

The **READ ONLY** option, introduced in MySQL 8.0.22, controls whether to permit modification of the database and objects within it. The permitted values are **DEFAULT** or **0** (not read only) and **1** (read only). This option is useful for database migration because a database for which **READ ONLY** is enabled can be migrated to another MySQL instance without concern that the database might be changed during the operation.

With NDB Cluster, making a database read only on one [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) server is synchronized to other [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) servers in the same cluster, so that the database becomes read only on all [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) servers.

The **READ ONLY** option, if enabled, is displayed in the **INFORMATION\_SCHEMA** [**SCHEMATA\_EXTENSIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-schemata-extensions-table) table. See [Section 26.3.32, “The INFORMATION\_SCHEMA SCHEMATA\_EXTENSIONS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-schemata-extensions-table).

The **READ ONLY** option cannot be enabled for these system schemas: **mysql**, **information\_schema**, **performance\_schema**.

In [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) statements, the **READ ONLY** option interacts with other instances of itself and with other options as follows:

An error occurs if multiple instances of **READ ONLY** conflict (for example, **READ ONLY = 1 READ ONLY = 0**).

An [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) statement that contains only (nonconflicting) **READ ONLY** options is permitted even for a read-only database.

A mix of (nonconflicting) **READ ONLY** options with other options is permitted if the read-only state of the database either before or after the statement permits modifications. If the read-only state both before and after prohibits changes, an error occurs.

This statement succeeds whether or not the database is read only:

ALTER DATABASE mydb READ ONLY = 0 DEFAULT COLLATE utf8mb4\_bin;

This statement succeeds if the database is not read only, but fails if it is already read only:

ALTER DATABASE mydb READ ONLY = 1 DEFAULT COLLATE utf8mb4\_bin;

Enabling **READ ONLY** affects all users of the database, with these exceptions that are not subject to read-only checks:

Statements executed by the server as part of server initialization, restart, upgrade, or replication.

Statements in a file named at server startup by the [**init\_file**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_init_file) system variable.

**TEMPORARY** tables; it is possible to create, alter, drop, and write to **TEMPORARY** tables in a read-only database.

NDB Cluster non-SQL inserts and updates.

Other than for the excepted operations just listed, enabling **READ ONLY** prohibits write operations to the database and its objects, including their definitions, data, and metadata. The following list details affected SQL statements and operations:

The database itself:

[**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database)

[**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) (except to change the **READ ONLY** option)

[**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database)

Views:

[**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view)

[**ALTER VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-view)

[**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view)

Selecting from views that invoke functions with side effects.

Updating updatable views.

Statements that create or drop objects in a writable database are rejected if they affect metadata of a view in a read-only database (for example, by making the view valid or invalid).

Stored routines:

[**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure)

[**DROP PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-procedure)

[**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) (of procedures with side effects)

[**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function)

[**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function)

[**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) (of functions with side effects)

For procedures and functions, read-only checks follow prelocking behavior. For [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statements, read-only checks are done on a per-statement basis, so if some conditionally executed statement writing to a read-only database does not actually execute, the call still succeeds. On the other hand, for a function called within a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), execution of the function body happens in prelocked mode. As long as a some statement within the function writes to a read-only database, execution of the function fails with an error regardless of whether the statement actually executes.

Triggers:

[**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger)

[**DROP TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-trigger)

Trigger invocation.

Events:

[**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event)

[**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event)

[**DROP EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-event)

Event execution:

Executing an event in the database fails because that would change the last-execution timestamp, which is event metadata stored in the data dictionary. Failure of event execution also has the effect of causing the event scheduler to stop.

If an event writes to an object in a read-only database, execution of the event fails with an error, but the event scheduler is not stopped.

Tables:

[**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table)

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table)

[**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index)

[**DROP INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-index)

[**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table)

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table)

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table)

[**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete)

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)

[**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table)

[**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data)

[**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml)

[**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace)

[**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update)

For cascading foreign keys where the child table is in a read-only database, updates and deletes on the parent are rejected even if the child table is not directly affected.

For a **MERGE** table such as **CREATE TABLE s1.t(i int) ENGINE MERGE UNION (s2.t, s3.t), INSERT\_METHOD=...**, the following behavior applies:

Inserting into the **MERGE** table (**INSERT into s1.t**) fails if at least one of **s1**, **s2**, **s3** is read only, regardless of insert method. The insert is refused even if it would actually end up in a writable table.

Dropping the **MERGE** table (**DROP TABLE s1.t**) succeeds as long as **s1** is not read only. It is permitted to drop a **MERGE** table that refers to a read-only database.

An [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) statement blocks until all concurrent transactions that have already accessed an object in the database being altered have committed. Conversely, a write transaction accessing an object in a database being altered in a concurrent [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) blocks until the [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) has committed.

If the Clone plugin is used to clone a local or remote data directory, the databases in the clone retain the read-only state they had in the source data directory. The read-only state does not affect the cloning process itself. If it is not desirable to have the same database read-only state in the clone, the option must be changed explicitly for the clone after the cloning process has finished, using [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) operations on the clone.

When cloning from a donor to a recipient, if the recipient has a user database that is read only, cloning fails with an error message. Cloning may be retried after making the database writable.

**READ ONLY** is permitted for [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database), but not for [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database). However, for a read-only database, the statement produced by [**SHOW CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-database) does include **READ ONLY=1** within a comment to indicate its read-only status:

mysql> **ALTER DATABASE mydb READ ONLY = 1;**

mysql> **SHOW CREATE DATABASE mydb\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Database: mydb

Create Database: CREATE DATABASE `mydb`

/\*!40100 DEFAULT CHARACTER SET utf8mb4

COLLATE utf8mb4\_0900\_ai\_ci \*/

/\*!80016 DEFAULT ENCRYPTION='N' \*/

/\* READ ONLY = 1 \*/

If the server executes a [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) statement containing such a comment, the server ignores the comment and the **READ ONLY** option is not processed. This has implications for [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) and [**mysqlpump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlpump), which use [**SHOW CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-database) to produce [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) statements in dump output:

In a dump file, the [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) statement for a read-only database contains the commented **READ ONLY** option.

The dump file can be restored as usual, but because the server ignores the commented **READ ONLY** option, the restored database is not read only. If the database is to be read ony after being restored, you must execute [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) manually to make it so.

Suppose that **mydb** is read only and you dump it as follows:

shell> **mysqldump --databases mydb > mydb.sql**

A restore operation later must be followed by [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) if **mydb** should still be read only:

shell> **mysql**

mysql> **SOURCE mydb.sql;**

mysql> **ALTER DATABASE mydb READ ONLY = 1;**

MySQL Enterprise Backup is not subject to this issue. It backs up and restores a read-only database like any other, but enables the **READ ONLY** option at restore time if it was enabled at backup time.

[**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) is written to the binary log, so a change to the **READ ONLY** option on a replication source server also affects replicas. To prevent this from happening, binary logging must be disabled prior to execution of the [**ALTER DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-database) statement. For example, to prepare for migrating a database without affecting replicas, perform these operations:

Within a single session, disable binary logging and enable **READ ONLY** for the database:

mysql> **SET sql\_log\_bin = OFF;**

mysql> **ALTER DATABASE mydb READ ONLY = 1;**

Dump the database, for example, with [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) or [**mysqlpump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlpump):

shell> **mysqldump --databases mydb > mydb.sql**

Within a single session, disable binary logging and disable **READ ONLY** for the database:

mysql> **SET sql\_log\_bin = OFF;**

mysql> **ALTER DATABASE mydb READ ONLY = 0;**

### 13.1.3 ALTER EVENT Statement

ALTER

[DEFINER = ***user***]

EVENT ***event\_name***

[ON SCHEDULE ***schedule***]

[ON COMPLETION [NOT] PRESERVE]

[RENAME TO ***new\_event\_name***]

[ENABLE | DISABLE | DISABLE ON SLAVE]

[COMMENT '***string***']

[DO ***event\_body***]

The [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement changes 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) clauses is exactly the same as when used with [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event). (See [Section 13.1.13, “CREATE EVENT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event).)

Any user can alter an event defined on a database for which that user has the [**EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_event) privilege. When a user executes a successful [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement, that user becomes the definer for the affected event.

[**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) works only with an existing event:

mysql> **ALTER EVENT no\_such\_event**

> **ON SCHEDULE**

> **EVERY '2:3' DAY\_HOUR;**

ERROR 1517 (HY000): Unknown event 'no\_such\_event'

In each of the following examples, assume that the event named **myevent** is defined as shown here:

CREATE EVENT myevent

ON SCHEDULE

EVERY 6 HOUR

COMMENT 'A sample comment.'

DO

UPDATE myschema.mytable SET mycol = mycol + 1;

The following statement changes the schedule for **myevent** from once every six hours starting immediately to once every twelve hours, starting four hours from the time the statement is run:

ALTER EVENT myevent

ON SCHEDULE

EVERY 12 HOUR

STARTS CURRENT\_TIMESTAMP + INTERVAL 4 HOUR;

It is possible to change multiple characteristics of an event in a single statement. This example changes the SQL statement executed by **myevent** to one that deletes all records from **mytable**; it also changes the schedule for the event such that it executes once, one day after this [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement is run.

ALTER EVENT myevent

ON SCHEDULE

AT CURRENT\_TIMESTAMP + INTERVAL 1 DAY

DO

TRUNCATE TABLE myschema.mytable;

Specify the options in an [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement only for those characteristics that you want to change; omitted options keep their existing values. This includes any default values for [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) such as **ENABLE**.

To disable **myevent**, use this [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement:

ALTER EVENT myevent

DISABLE;

The **ON SCHEDULE** clause may use expressions involving built-in MySQL functions and user variables to obtain any of the ***timestamp*** or ***interval*** values which it contains. You cannot use stored routines or user-defined functions in such expressions, and you cannot use any table references; however, you can use **SELECT FROM DUAL**. This is true for both [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) and [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statements. References to stored routines, user-defined functions, and tables in such cases are specifically not permitted, and fail with an error (see Bug #22830).

Although an [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement that contains another [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement in its [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) clause appears to succeed, when the server attempts to execute the resulting scheduled event, the execution fails with an error.

To rename an event, use the [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement's **RENAME TO** clause. This statement renames the event **myevent** to **yourevent**:

ALTER EVENT myevent

RENAME TO yourevent;

You can also move an event to a different database using **ALTER EVENT ... RENAME TO ...** and ***db\_name.event\_name*** notation, as shown here:

ALTER EVENT olddb.myevent

RENAME TO newdb.myevent;

To execute the previous statement, the user executing it must have the [**EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_event) privilege on both the **olddb** and **newdb** databases.

**Note**

There is no **RENAME EVENT** statement.

The value **DISABLE ON SLAVE** is used on a replica instead of **ENABLE** or **DISABLE** to indicate an event that was created on the replication source server and replicated to the replica, but that is not executed on the replica. Normally, **DISABLE ON SLAVE** is set automatically as required; however, there are some circumstances under which you may want or need to change it manually. See [Section 17.5.1.16, “Replication of Invoked Features”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-invoked), for more information.

### 13.1.4 ALTER FUNCTION Statement

ALTER FUNCTION ***func\_name*** [***characteristic*** ...]

***characteristic***: {

COMMENT '***string***'

| LANGUAGE SQL

| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

| SQL SECURITY { DEFINER | INVOKER }

}

This statement can be used to change the characteristics of a stored function. More than one change may be specified in an [**ALTER FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function) and [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function).

You must have the [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine) privilege for the function. (That privilege is granted automatically to the function creator.) If binary logging is enabled, the [**ALTER FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-function) statement might also require the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege, as described in [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging).

### 13.1.5 ALTER INSTANCE Statement

ALTER INSTANCE ***instance\_action***

***instance\_action***: {

| {ENABLE|DISABLE} INNODB REDO\_LOG

| ROTATE INNODB MASTER KEY

| ROTATE BINLOG MASTER KEY

| RELOAD TLS

[FOR CHANNEL {mysql\_main | mysql\_admin}]

[NO ROLLBACK ON ERROR]

| RELOAD KEYRING

}

**ALTER INSTANCE** defines actions applicable to a MySQL server instance. The statement supports these actions:

**ALTER INSTANCE {ENABLE | DISABLE} INNODB REDO\_LOG**

This action enables or disables **InnoDB** redo logging. Redo logging is enabled by default. This feature is intended only for loading data into a new MySQL instance. The statement is not written to the binary log. This action was introduced in MySQL 8.0.21.

**Warning**

Do not disable redo logging on a production system. While it is permitted to shutdown and restart the server while redo logging is disabled, an unexpected server stoppage while redo logging is disabled can cause data loss and instance corruption.

An [**ALTER INSTANCE [ENABLE|DISABLE] INNODB REDO\_LOG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance) operation requires an exclusive backup lock, which prevents other [**ALTER INSTANCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance) operations from executing concurrently. Other [**ALTER INSTANCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance) operations must wait for the lock to be released before executing.

For more information, see [Disabling Redo Logging](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-disable-redo-logging).

**ALTER INSTANCE ROTATE INNODB MASTER KEY**

This action rotates the master encryption key used for **InnoDB** tablespace encryption. Key rotation requires the [**ENCRYPTION\_KEY\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_encryption-key-admin) or [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege. To perform this action, a keyring plugin must be installed and configured. For instructions, see [Section 6.4.4, “The MySQL Keyring”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring).

**ALTER INSTANCE ROTATE INNODB MASTER KEY** supports concurrent DML. However, it cannot be run concurrently with [**CREATE TABLE ... ENCRYPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE ... ENCRYPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations, and locks are taken to prevent conflicts that could arise from concurrent execution of these statements. If one of the conflicting statements is running, it must complete before another can proceed.

**ALTER INSTANCE ROTATE INNODB MASTER KEY** statements are written to the binary log so that they can be executed on replicated servers.

For additional **ALTER INSTANCE ROTATE INNODB MASTER KEY** usage information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

**ALTER INSTANCE ROTATE BINLOG MASTER KEY**

This action rotates the binary log master key used for binary log encryption. Key rotation for the binary log master key requires the [**BINLOG\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_binlog-encryption-admin) or [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege. The statement cannot be used if the [**binlog\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption) system variable is set to **OFF**. To perform this action, a keyring plugin must be installed and configured. For instructions, see [Section 6.4.4, “The MySQL Keyring”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring).

**ALTER INSTANCE ROTATE BINLOG MASTER KEY** actions are not written to the binary log and are not executed on replicas. Binary log master key rotation can therefore be carried out in replication environments including a mix of MySQL versions. To schedule regular rotation of the binary log master key on all applicable source and replica servers, you can enable the MySQL Event Scheduler on each server and issue the **ALTER INSTANCE ROTATE BINLOG MASTER KEY** statement using a [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statement. If you rotate the binary log master key because you suspect that the current or any of the previous binary log master keys might have been compromised, issue the statement on every applicable source and replica server, which enables you to verify immediate compliance.

For additional **ALTER INSTANCE ROTATE BINLOG MASTER KEY** usage information, including what to do if the process does not complete correctly or is interrupted by an unexpected server halt, see [Section 17.3.2, “Encrypting Binary Log Files and Relay Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-binlog-encryption).

**ALTER INSTANCE RELOAD TLS**

This action reconfigures a TLS context from the current values of the system variables that define the context. It also updates the status variables that reflect the active context values. This action requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege. For additional information about reconfiguring the TLS context, including which system and status variables are context-related, see [Server-Side Runtime Configuration and Monitoring for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#using-encrypted-connections-server-side-runtime-configuration).

By default, the statement reloads the TLS context for the main connection interface. If the **FOR CHANNEL** clause (available as of MySQL 8.0.21) is given, the statement reloads the TLS context for the named channel: **mysql\_main** for the main connection interface, **mysql\_admin** for the administrative connection interface. For information about the different interfaces, see [Section 5.1.12.1, “Connection Interfaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#connection-interfaces). The updated TLS context properties are exposed in the Performance Schema [**tls\_channel\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-tls-channel-status-table) table. See [Section 27.12.21.7, “The tls\_channel\_status Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-tls-channel-status-table).

Updating the TLS context for the main interface may also affect the administrative interface because unless some nondefault TLS value is configured for that interface, it uses the same TLS context as the main interface.

By default, the **RELOAD TLS** action rolls back with an error and has no effect if the configuration values do not permit creation of the new TLS context. The previous context values continue to be used for new connections. If the optional **NO ROLLBACK ON ERROR** clause is given and the new context cannot be created, rollback does not occur. Instead, a warning is generated and encryption is disabled for new connections on the interface to which the statement applies.

**ALTER INSTANCE RELOAD TLS** statements are not written to the binary log (and thus are not replicated). TLS configuration is local and depends on local files not necessarily present on all servers involved.

**ALTER INSTANCE RELOAD KEYRING**

If a keyring component is installed, this action tells the component to re-read its configuration file and reinitialize any keyring in-memory data. If you modify the component configuration at runtime, the new configuration does not take effect until you perform this action. Keyring reloading requires the [**ENCRYPTION\_KEY\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_encryption-key-admin) privilege. This action was added in MySQL 8.0.24.

This action enables reconfiguring only the currently installed keyring component. It does not enable changing which component is installed. For example, if you change the configuration for the installed keyring component, [**ALTER INSTANCE RELOAD KEYRING**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-reload-keyring) causes the new configuration to take effect. On the other hand, if you change the keyring component named in the server manifest file, [**ALTER INSTANCE RELOAD KEYRING**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-instance-reload-keyring) has no effect and the current component remains installed.

**ALTER INSTANCE RELOAD KEYRING** statements are not written to the binary log (and thus are not replicated).

### 13.1.6 ALTER LOGFILE GROUP Statement

ALTER LOGFILE GROUP ***logfile\_group***

ADD UNDOFILE '***file\_name***'

[INITIAL\_SIZE [=] ***size***]

[WAIT]

ENGINE [=] ***engine\_name***

This statement adds an **UNDO** file named '***file\_name***' to an existing log file group ***logfile\_group***. An [**ALTER LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-logfile-group) statement has one and only one **ADD UNDOFILE** clause. No **DROP UNDOFILE** clause is currently supported.

**Note**

All NDB Cluster Disk Data objects share the same namespace. This means that each Disk Data object must be uniquely named (and not merely each Disk Data object of a given type). For example, you cannot have a tablespace and an undo log file with the same name, or an undo log file and a data file with the same name.

The optional **INITIAL\_SIZE** parameter sets the **UNDO** file's initial size in bytes; if not specified, the initial size defaults to 134217728 (128 MB). You may optionally follow ***size*** with a one-letter abbreviation for an order of magnitude, similar to those used in my.cnf. Generally, this is one of the letters **M** (megabytes) or **G** (gigabytes). (Bug #13116514, Bug #16104705, Bug #62858)

On 32-bit systems, the maximum supported value for **INITIAL\_SIZE** is 4294967296 (4 GB). (Bug #29186)

The minimum allowed value for **INITIAL\_SIZE** is 1048576 (1 MB). (Bug #29574)

**Note**

**WAIT** is parsed but otherwise ignored. This keyword currently has no effect, and is intended for future expansion.

The **ENGINE** parameter (required) determines the storage engine which is used by this log file group, with ***engine\_name*** being the name of the storage engine. Currently, the only accepted values for ***engine\_name*** are “[**NDBCLUSTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html)” and “[**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html)”. The two values are equivalent.

Here is an example, which assumes that the log file group **lg\_3** has already been created using [**CREATE LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group) (see [Section 13.1.16, “CREATE LOGFILE GROUP Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group)):

ALTER LOGFILE GROUP lg\_3

ADD UNDOFILE 'undo\_10.dat'

INITIAL\_SIZE=32M

ENGINE=NDBCLUSTER;

When [**ALTER LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-logfile-group) is used with **ENGINE = NDBCLUSTER** (alternatively, **ENGINE = NDB**), an **UNDO** log file is created on each NDB Cluster data node. You can verify that the **UNDO** files were created and obtain information about them by querying the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table. For example:

mysql> **SELECT FILE\_NAME, LOGFILE\_GROUP\_NUMBER, EXTRA**

-> **FROM INFORMATION\_SCHEMA.FILES**

-> **WHERE LOGFILE\_GROUP\_NAME = 'lg\_3';**

+-------------+----------------------+----------------+

| FILE\_NAME | LOGFILE\_GROUP\_NUMBER | EXTRA |

+-------------+----------------------+----------------+

| newdata.dat | 0 | CLUSTER\_NODE=3 |

| newdata.dat | 0 | CLUSTER\_NODE=4 |

| undo\_10.dat | 11 | CLUSTER\_NODE=3 |

| undo\_10.dat | 11 | CLUSTER\_NODE=4 |

+-------------+----------------------+----------------+

4 rows in set (0.01 sec)

(See [Section 26.3.15, “The INFORMATION\_SCHEMA FILES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table).)

Memory used for **UNDO\_BUFFER\_SIZE** comes from the global pool whose size is determined by the value of the [**SharedGlobalMemory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-sharedglobalmemory) data node configuration parameter. This includes any default value implied for this option by the setting of the [**InitialLogFileGroup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-initiallogfilegroup) data node configuration parameter.

[**ALTER LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-logfile-group) is useful only with Disk Data storage for NDB Cluster. For more information, see [Section 23.5.10, “NDB Cluster Disk Data Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data).

### 13.1.7 ALTER PROCEDURE Statement

ALTER PROCEDURE ***proc\_name*** [***characteristic*** ...]

***characteristic***: {

COMMENT '***string***'

| LANGUAGE SQL

| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

| SQL SECURITY { DEFINER | INVOKER }

}

This statement can be used to change the characteristics of a stored procedure. More than one change may be specified in an [**ALTER PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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 [**DROP PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-procedure) and [**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure).

You must have the [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine) privilege for the procedure. By default, that privilege is granted automatically to the procedure creator. This behavior can be changed by disabling the [**automatic\_sp\_privileges**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_automatic_sp_privileges) system variable. See [Section 25.2.2, “Stored Routines and MySQL Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-routines-privileges).

### 13.1.8 ALTER SERVER Statement

ALTER SERVER ***server\_name***

OPTIONS (***option*** [, ***option***] ...)

Alters the server information for ***server\_name***, adjusting any of the options permitted in the [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server) statement. The corresponding fields in the **mysql.servers** table are updated accordingly. This statement requires the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege.

For example, to update the **USER** option:

ALTER SERVER s OPTIONS (USER 'sally');

**ALTER SERVER** causes an implicit commit. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

**ALTER SERVER** is not written to the binary log, regardless of the logging format that is in use.

### 13.1.9 ALTER TABLE Statement

[13.1.9.1 ALTER TABLE Partition Operations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-partition-operations)

[13.1.9.2 ALTER TABLE and Generated Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-generated-columns)

[13.1.9.3 ALTER TABLE Examples](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-examples)

ALTER TABLE ***tbl\_name***

[***alter\_option*** [, ***alter\_option***] ...]

[***partition\_options***]

***alter\_option***: {

***table\_options***

| ADD [COLUMN] ***col\_name*** ***column\_definition***

[FIRST | AFTER ***col\_name***]

| ADD [COLUMN] (***col\_name*** ***column\_definition***,...)

| ADD {INDEX | KEY} [***index\_name***]

[***index\_type***] (***key\_part***,...) [***index\_option***] ...

| ADD {FULLTEXT | SPATIAL} [INDEX | KEY] [***index\_name***]

(***key\_part***,...) [***index\_option***] ...

| ADD [CONSTRAINT [***symbol***]] PRIMARY KEY

[***index\_type***] (***key\_part***,...)

[***index\_option***] ...

| ADD [CONSTRAINT [***symbol***]] UNIQUE [INDEX | KEY]

[***index\_name***] [***index\_type***] (***key\_part***,...)

[***index\_option***] ...

| ADD [CONSTRAINT [***symbol***]] FOREIGN KEY

[***index\_name***] (***col\_name***,...)

***reference\_definition***

| ADD [CONSTRAINT [***symbol***]] CHECK (***expr***) [[NOT] ENFORCED]

| DROP {CHECK | CONSTRAINT} ***symbol***

| ALTER {CHECK | CONSTRAINT} ***symbol*** [NOT] ENFORCED

| ALGORITHM [=] {DEFAULT | INSTANT | INPLACE | COPY}

| ALTER [COLUMN] ***col\_name*** {

SET DEFAULT {***literal*** | (***expr***)}

| SET {VISIBLE | INVISIBLE}

| DROP DEFAULT

}

| ALTER INDEX ***index\_name*** {VISIBLE | INVISIBLE}

| CHANGE [COLUMN] ***old\_col\_name*** ***new\_col\_name*** ***column\_definition***

[FIRST | AFTER ***col\_name***]

| [DEFAULT] CHARACTER SET [=] ***charset\_name*** [COLLATE [=] ***collation\_name***]

| CONVERT TO CHARACTER SET ***charset\_name*** [COLLATE ***collation\_name***]

| {DISABLE | ENABLE} KEYS

| {DISCARD | IMPORT} TABLESPACE

| DROP [COLUMN] ***col\_name***

| DROP {INDEX | KEY} ***index\_name***

| DROP PRIMARY KEY

| DROP FOREIGN KEY ***fk\_symbol***

| FORCE

| LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}

| MODIFY [COLUMN] ***col\_name*** ***column\_definition***

[FIRST | AFTER ***col\_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***

| RENAME [TO | AS] ***new\_tbl\_name***

| {WITHOUT | WITH} VALIDATION

}

***partition\_options***:

***partition\_option*** [***partition\_option***] ...

***partition\_option***: {

ADD PARTITION (***partition\_definition***)

| DROP PARTITION ***partition\_names***

| DISCARD PARTITION {***partition\_names*** | ALL} TABLESPACE

| IMPORT PARTITION {***partition\_names*** | ALL} TABLESPACE

| TRUNCATE PARTITION {***partition\_names*** | ALL}

| COALESCE PARTITION ***number***

| REORGANIZE PARTITION ***partition\_names*** INTO (***partition\_definitions***)

| EXCHANGE PARTITION ***partition\_name*** WITH TABLE ***tbl\_name*** [{WITH | WITHOUT} VALIDATION]

| ANALYZE PARTITION {***partition\_names*** | ALL}

| CHECK PARTITION {***partition\_names*** | ALL}

| OPTIMIZE PARTITION {***partition\_names*** | ALL}

| REBUILD PARTITION {***partition\_names*** | ALL}

| REPAIR PARTITION {***partition\_names*** | ALL}

| REMOVE PARTITIONING

}

***key\_part***: {***col\_name*** [(***length***)] | (***expr***)} [ASC | DESC]

***index\_type***:

USING {BTREE | HASH}

***index\_option***: {

KEY\_BLOCK\_SIZE [=] ***value***

| ***index\_type***

| WITH PARSER ***parser\_name***

| COMMENT '***string***'

| {VISIBLE | INVISIBLE}

}

***table\_options***:

***table\_option*** [[,] ***table\_option***] ...

***table\_option***: {

AUTOEXTEND\_SIZE [=] ***value***

| AUTO\_INCREMENT [=] ***value***

| AVG\_ROW\_LENGTH [=] ***value***

| [DEFAULT] CHARACTER SET [=] ***charset\_name***

| CHECKSUM [=] {0 | 1}

| [DEFAULT] COLLATE [=] ***collation\_name***

| COMMENT [=] '***string***'

| COMPRESSION [=] {'ZLIB' | 'LZ4' | 'NONE'}

| CONNECTION [=] '***connect\_string***'

| {DATA | INDEX} DIRECTORY [=] '***absolute path to directory***'

| DELAY\_KEY\_WRITE [=] {0 | 1}

| ENCRYPTION [=] {'Y' | 'N'}

| ENGINE [=] ***engine\_name***

| ENGINE\_ATTRIBUTE [=] '***string***'

| INSERT\_METHOD [=] { NO | FIRST | LAST }

| KEY\_BLOCK\_SIZE [=] ***value***

| MAX\_ROWS [=] ***value***

| MIN\_ROWS [=] ***value***

| PACK\_KEYS [=] {0 | 1 | DEFAULT}

| PASSWORD [=] '***string***'

| ROW\_FORMAT [=] {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}

| SECONDARY\_ENGINE\_ATTRIBUTE [=] '***string***'

| STATS\_AUTO\_RECALC [=] {DEFAULT | 0 | 1}

| STATS\_PERSISTENT [=] {DEFAULT | 0 | 1}

| STATS\_SAMPLE\_PAGES [=] ***value***

| TABLESPACE ***tablespace\_name*** [STORAGE {DISK | MEMORY}]

| UNION [=] (***tbl\_name***[,***tbl\_name***]...)

}

***partition\_options***:

(see [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) options)

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) changes the structure of a 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 characteristics such as the storage engine used for the table or the table comment.

To use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), you need [**ALTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter), [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create), and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privileges for the table. Renaming a table requires [**ALTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter) and [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) on the old table, [**ALTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter), [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create), and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) on the new table.

Following the table name, specify the alterations to be made. If none are given, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) does nothing.

The syntax for many of the permissible alterations is similar to clauses of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement. ***column\_definition*** clauses use the same syntax for **ADD** and **CHANGE** as for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). For more information, see [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table).

The word **COLUMN** is optional and can be omitted, except for **RENAME COLUMN** (to distinguish a column-renaming operation from the **RENAME** table-renaming operation).

Multiple **ADD**, **ALTER**, **DROP**, and **CHANGE** clauses are permitted in a single [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement, separated by commas. This is a MySQL extension to standard SQL, which permits only one of each clause per [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. For example, to drop multiple columns in a single statement, do this:

ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;

If a storage engine does not support an attempted [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation, a warning may result. Such warnings can be displayed with [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings). See [Section 13.7.7.42, “SHOW WARNINGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings). For information on troubleshooting [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), see [Section B.3.6.1, “Problems with ALTER TABLE”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#alter-table-problems).

For information about generated columns, see [Section 13.1.9.2, “ALTER TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-generated-columns).

For usage examples, see [Section 13.1.9.3, “ALTER TABLE Examples”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-examples).

**InnoDB** in MySQL 8.0.17 and later supports addition of multi-valued indexes on JSON columns using a ***key\_part*** specification can take the form **(CAST *json\_path* AS *type* ARRAY)**. See [Multi-Valued Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-multi-valued), for detailed information regarding multi-valued index creation and usage of, as well as restrictions and limitations on multi-valued indexes.

With the [**mysql\_info()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-info.html) C API function, you can find out how many rows were copied by [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). See [mysql\_info()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-info.html).

There are several additional aspects to the **ALTER TABLE** statement, described under the following topics in this section:

[Table Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-options)

[Performance and Space Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-performance)

[Concurrency Control](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-concurrency)

[Adding and Dropping Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-add-drop-column)

[Renaming, Redefining, and Reordering Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-redefine-column)

[Primary Keys and Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-index)

[Foreign Keys and Other Constraints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-foreign-key)

[Changing the Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-character-set)

[Importing InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-import)

[Row Order for MyISAM Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-row-order)

[Partitioning Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-partition-options)

#### Table Options

***table\_options*** signifies table options of the kind that can be used in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, such as **ENGINE**, **AUTO\_INCREMENT**, **AVG\_ROW\_LENGTH**, **MAX\_ROWS**, **ROW\_FORMAT**, or **TABLESPACE**.

For descriptions of all table options, see [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). However, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) ignores **DATA DIRECTORY** and **INDEX DIRECTORY** when given as table options. [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) permits them only as partitioning options, and requires that you have the **FILE** privilege.

Use of table options with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) provides a convenient way of altering single table characteristics. For example:

If **t1** is currently not an **InnoDB** table, this statement changes its storage engine to **InnoDB**:

ALTER TABLE t1 ENGINE = InnoDB;

See [Section 15.6.1.5, “Converting Tables from MyISAM to InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#converting-tables-to-innodb) for considerations when switching tables to the **InnoDB** storage engine.

When you specify an **ENGINE** clause, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) rebuilds the table. This is true even if the table already has the specified storage engine.

Running [**ALTER TABLE *tbl\_name* ENGINE=INNODB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on an existing **InnoDB** table performs a “null” [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation, which can be used to defragment an **InnoDB** table, as described in [Section 15.11.4, “Defragmenting a Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-file-defragmenting). Running [**ALTER TABLE *tbl\_name* FORCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on an **InnoDB** table performs the same function.

[**ALTER TABLE *tbl\_name* ENGINE=INNODB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**ALTER TABLE *tbl\_name* FORCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) use [online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl). For more information, see [Section 15.12, “InnoDB and Online DDL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl).

The outcome of attempting to change the storage engine of a table is affected by whether the desired storage engine is available and the setting of the [**NO\_ENGINE\_SUBSTITUTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_engine_substitution) SQL mode, as described in [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

To prevent inadvertent loss of data, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) cannot be used to change the storage engine of a table to **MERGE** or **BLACKHOLE**.

To change the **InnoDB** table to use compressed row-storage format:

ALTER TABLE t1 ROW\_FORMAT = COMPRESSED;

The **ENCRYPTION** clause enables or disables page-level data encryption for an **InnoDB** table. A keyring plugin must be installed and configured to enable encryption.

If the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) variable is enabled, the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege is required to use an **ENCRYPTION** clause with a setting that differs from the default schema encryption setting.

Prior to MySQL 8.0.16, the **ENCRYPTION** clause was only supported when altering tables residing in file-per-table tablespaces. As of MySQL 8.0.16, the **ENCRYPTION** clause is also supported for tables residing in general tablespaces.

For tables that reside in general tablespaces, table and tablespace encryption must match.

Altering table encryption by moving a table to a different tablespace or changing the storage engine is not permitted without explicitly specifying an **ENCRYPTION** clause.

As of MySQL 8.0.16, specifying an **ENCRYPTION** clause with a value other than **'N'** or **''** is not permitted if the table uses a storage engine that does not support encryption. Previously, the clause was accepted. Attempting to create a table without an **ENCRYPTION** clause in an encryption-enabled schema using a storage engine that does not support encryption is also not permitted.

For more information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

To reset the current auto-increment value:

ALTER TABLE t1 AUTO\_INCREMENT = 13;

You cannot reset the counter to a value less than or equal to the value that is currently in use. For both **InnoDB** and **MyISAM**, if the value is less than or equal to the maximum value currently in the **AUTO\_INCREMENT** column, the value is reset to the current maximum **AUTO\_INCREMENT** column value plus one.

To change the default table character set:

ALTER TABLE t1 CHARACTER SET = utf8;

See also [Changing the Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-character-set).

To add (or change) a table comment:

ALTER TABLE t1 COMMENT = 'New table comment';

Use **ALTER TABLE** with the **TABLESPACE** option to move **InnoDB** tables between existing [general tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_general_tablespace), [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces, and the [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace). See [Moving Tables Between Tablespaces Using ALTER TABLE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces-moving-non-partitioned-tables).

**ALTER TABLE ... TABLESPACE** operations always cause a full table rebuild, even if the **TABLESPACE** attribute has not changed from its previous value.

**ALTER TABLE ... TABLESPACE** syntax does not support moving a table from a temporary tablespace to a persistent tablespace.

The **DATA DIRECTORY** clause, which is supported with [**CREATE TABLE ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), is not supported with **ALTER TABLE ... TABLESPACE**, and is ignored if specified.

For more information about the capabilities and limitations of the **TABLESPACE** option, see [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table).

MySQL NDB Cluster 8.0 supports setting **NDB\_TABLE** options for controlling a table's partition balance (fragment count type), read-from-any-replica capability, full replication, or any combination of these, as part of the table comment for an **ALTER TABLE** statement in the same manner as for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), as shown in this example:

ALTER TABLE t1 COMMENT = "NDB\_TABLE=READ\_BACKUP=0,PARTITION\_BALANCE=FOR\_RA\_BY\_NODE";

Bear in mind that **ALTER TABLE ... COMMENT ...** discards any existing comment for the table. See [Setting NDB\_TABLE options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-comment-ndb-table-options), for additional information and examples.

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** options (available as of MySQL 8.0.21) are used to specify table, column, and index attributes for primary and secondary storage engines. The options are reserved for future use. Index attributes cannot be altered. An index must be dropped and added back with the desired change, which can be performed in a single [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement.

To verify that the table options were changed as intended, use [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table), or query the [**INFORMATION\_SCHEMA.TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table) table.

#### Performance and Space Requirements

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations are processed using one of the following algorithms:

**COPY**: Operations are performed on a copy of the original table, and table data is copied from the original table to the new table row by row. Concurrent DML is not permitted.

**INPLACE**: Operations avoid copying table data but may rebuild the table in place. An exclusive metadata lock on the table may be taken briefly during preparation and execution phases of the operation. Typically, concurrent DML is supported.

**INSTANT**: Operations only modify metadata in the data dictionary. No exclusive metadata locks are taken on the table during preparation and execution, and table data is unaffected, making operations instantaneous. Concurrent DML is permitted. (Introduced in MySQL 8.0.12)

The **ALGORITHM** clause is optional. If the **ALGORITHM** clause is omitted, MySQL uses **ALGORITHM=INSTANT** for storage engines and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) clauses that support it. Otherwise, **ALGORITHM=INPLACE** is used. If **ALGORITHM=INPLACE** is not supported, **ALGORITHM=COPY** is used.

Specifying an **ALGORITHM** clause requires the operation to use the specified algorithm for clauses and storage engines that support it, or fail with an error otherwise. Specifying **ALGORITHM=DEFAULT** is the same as omitting the **ALGORITHM** clause.

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that use the **COPY** algorithm wait for other operations that are modifying the table to complete. After alterations are applied to the table copy, data is copied over, the original table is deleted, and the table copy is renamed to the name of the original table. While the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation executes, the original table is readable by other sessions (with the exception noted shortly). Updates and writes to the table started after the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation begins are stalled until the new table is ready, then are automatically redirected to the new table. The temporary copy of the table is created in the database directory of the original table unless it is a **RENAME TO** operation that moves the table to a database that resides in a different directory.

The exception referred to earlier is that [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) blocks reads (not just writes) at the point where it is ready to clear outdated table structures from the table and table definition caches. At this point, it must acquire an exclusive lock. To do so, it waits for current readers to finish, and blocks new reads and writes.

An [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation that uses the **COPY** algorithm prevents concurrent DML operations. Concurrent queries are still allowed. That is, a table-copying operation always includes at least the concurrency restrictions of **LOCK=SHARED** (allow queries but not DML). You can further restrict concurrency for operations that support the **LOCK** clause by specifying **LOCK=EXCLUSIVE**, which prevents DML and queries. For more information, see [Concurrency Control](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-concurrency).

To force use of the **COPY** algorithm for an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation that would otherwise not use it, specify **ALGORITHM=COPY** or enable the [**old\_alter\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_old_alter_table) system variable. If there is a conflict between the **old\_alter\_table** setting and an **ALGORITHM** clause with a value other than **DEFAULT**, the **ALGORITHM** clause takes precedence.

For **InnoDB** tables, an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation that uses the **COPY** algorithm on a table that resides in a [shared tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_shared_tablespace) can increase the amount of space used by the tablespace. Such operations require as much additional space as the data in the table plus indexes. For a table residing in a shared tablespace, the additional space used during the operation is not released back to the operating system as it is for a table that resides in a [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace.

For information about space requirements for online DDL operations, see [Section 15.12.3, “Online DDL Space Requirements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-space-requirements).

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that support the **INPLACE** algorithm include:

**ALTER TABLE** operations supported by the **InnoDB** [online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_online_ddl) feature. See [Section 15.12.1, “Online DDL Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-operations).

Renaming a table. MySQL renames files that correspond to the table ***tbl\_name*** without making a copy. (You can also use the [**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table) statement to rename tables. See [Section 13.1.36, “RENAME TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table).) Privileges granted specifically for the renamed table are not migrated to the new name. They must be changed manually.

Operations that only modify table metadata. These operations are immediate because the server does not touch table contents. Metadata-only operations include:

Renaming a column. In NDB Cluster 8.0.18 and later, this operation can also be performed online.

Changing the default value of a column (except for [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables).

Modifying the definition of an [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) or [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) column by adding new enumeration or set members to the end of the list of valid member values, as long as the storage size of the data type does not change. For example, adding a member to a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) column that has 8 members changes the required storage per value from 1 byte to 2 bytes; this requires a table copy. Adding members in the middle of the list causes renumbering of existing members, which requires a table copy.

Changing the definition of a spatial column to remove the **SRID** attribute. (Adding or changing an **SRID** attribute does require a rebuild and cannot be done in place because the server must verify that all values have the specified SRID value.)

As of MySQL 8.0.14, changing a column character set, when these conditions apply:

The column data type is [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), a [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) type, or [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum).

The character set change is from **utf8mb3** to **utf8mb4**, or any character set to **binary**.

There is no index on the column.

As of MySQL 8.0.14, changing a generated column, when these conditions apply:

For **InnoDB** tables, statements that modify generated stored columns but do not change their type, expression, or nullability.

For non-**InnoDB** tables, statements that modify generated stored or virtual columns but do not change their type, expression, or nullability.

An example of such a change is a change to the column comment.

Renaming an index.

Adding or dropping a secondary index, for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables. See [Section 15.12.1, “Online DDL Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-operations).

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, operations that add and drop indexes on variable-width columns. These operations occur online, without table copying and without blocking concurrent DML actions for most of their duration. See [Section 23.5.11, “Online Operations with ALTER TABLE in NDB Cluster”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-online-operations).

Modifying index visibility with an **ALTER INDEX** operation.

Column modifications of tables containing generated columns that depend on columns with a **DEFAULT** value if the modified columns are not involved in the generated column expressions. For example, changing the **NULL** property of a separate column can be done in place without a table rebuild.

**ALTER TABLE** operations that support the **INSTANT** algorithm include:

Adding a column. This feature is referred to as “Instant **ADD COLUMN**”. Limitations apply. See [Section 15.12.1, “Online DDL Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-operations).

Adding or dropping a virtual column.

Adding or dropping a column default value.

Modifying the definition of an [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) or [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) column. The same restrictions apply as described above for **ALGORITHM=INSTANT**.

Changing the index type.

Renaming a table. The same restrictions apply as described above for **ALGORITHM=INSTANT**.

For more information about operations that support **ALGORITHM=INSTANT**, see [Section 15.12.1, “Online DDL Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl-operations).

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) upgrades MySQL 5.5 temporal columns to 5.6 format for **ADD COLUMN**, **CHANGE COLUMN**, **MODIFY COLUMN**, **ADD INDEX**, and **FORCE** operations. This conversion cannot be done using the **INPLACE** algorithm because the table must be rebuilt, so specifying **ALGORITHM=INPLACE** in these cases results in an error. Specify **ALGORITHM=COPY** if necessary.

If an **ALTER TABLE** operation on a multicolumn index used to partition a table by **KEY** changes the order of the columns, it can only be performed using **ALGORITHM=COPY**.

The **WITHOUT VALIDATION** and **WITH VALIDATION** clauses affect whether [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) performs an in-place operation for [virtual generated column](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_virtual_generated_column) modifications. See [Section 13.1.9.2, “ALTER TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-generated-columns).

NDB Cluster 8.0 supports online operations using the same **ALGORITHM=INPLACE** syntax used with the standard MySQL Server. **NDB** does not support changing a tablespace online; beginning with NDB 8.0.21, it is disallowed. See [Section 23.5.11, “Online Operations with ALTER TABLE in NDB Cluster”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-online-operations), for more information.

**ALTER TABLE** with **DISCARD ... PARTITION ... TABLESPACE** or **IMPORT ... PARTITION ... TABLESPACE** does not create any temporary tables or temporary partition files.

**ALTER TABLE** with **ADD PARTITION**, **DROP PARTITION**, **COALESCE PARTITION**, **REBUILD PARTITION**, or **REORGANIZE PARTITION** does not create temporary tables (except when used with [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables); however, these operations can and do create temporary partition files.

**ADD** or **DROP** operations for **RANGE** or **LIST** partitions are immediate operations or nearly so. **ADD** or **COALESCE** operations for **HASH** or **KEY** partitions copy data between all partitions, unless **LINEAR HASH** or **LINEAR KEY** was used; this is effectively the same as creating a new table, although the **ADD** or **COALESCE** operation is performed partition by partition. **REORGANIZE** operations copy only changed partitions and do not touch unchanged ones.

For **MyISAM** tables, you can speed up index re-creation (the slowest part of the alteration process) by setting the [**myisam\_sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_sort_buffer_size) system variable to a high value.

#### Concurrency Control

For [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that support it, you can use the **LOCK** clause to control the level of concurrent reads and writes on a table while it is being altered. Specifying a non-default value for this clause enables you to require a certain amount of concurrent access or exclusivity during the alter operation, and halts the operation if the requested degree of locking is not available.

Only **LOCK = DEFAULT** is permitted for operations that use **ALGORITHM=INSTANT**. The other **LOCK** clause parameters are not applicable.

The parameters for the **LOCK** clause are:

LOCK = DEFAULT

Maximum level of concurrency for the given **ALGORITHM** clause (if any) and **ALTER TABLE** operation: Permit concurrent reads and writes if supported. If not, permit concurrent reads if supported. If not, enforce exclusive access.

LOCK = NONE

If supported, permit concurrent reads and writes. Otherwise, an error occurs.

LOCK = SHARED

If supported, permit concurrent reads but block writes. Writes are blocked even if concurrent writes are supported by the storage engine for the given **ALGORITHM** clause (if any) and **ALTER TABLE** operation. If concurrent reads are not supported, an error occurs.

LOCK = EXCLUSIVE

Enforce exclusive access. This is done even if concurrent reads/writes are supported by the storage engine for the given **ALGORITHM** clause (if any) and **ALTER TABLE** operation.

#### Adding and Dropping Columns

Use **ADD** to add new columns to a table, and **DROP** to remove existing columns. **DROP *col\_name*** is a MySQL extension to standard SQL.

To add a column at a specific position within a table row, use **FIRST** or **AFTER *col\_name***. The default is to add the column last.

If a table contains only one column, the column cannot be dropped. If what you intend is to remove the table, use the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement instead.

If columns are dropped from a table, the columns are also removed from any index of which they are a part. If all columns that make up an index are dropped, the index is dropped as well. If you use **CHANGE** or **MODIFY** to shorten a column for which an index exists on the column, and the resulting column length is less than the index length, MySQL shortens the index automatically.

For **ALTER TABLE ... ADD**, if the column has an expression default value that uses a nondeterministic function, the statement may produce a warning or error. For further information, see [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults), and [Section 17.1.3.7, “Restrictions on Replication with GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-restrictions).

#### Renaming, Redefining, and Reordering Columns

The **CHANGE**, **MODIFY**, **RENAME COLUMN**, and **ALTER** clauses enable the names and definitions of existing columns to be altered. They have these comparative characteristics:

**CHANGE**:

Can rename a column and change its definition, or both.

Has more capability than **MODIFY** or **RENAME COLUMN**, but at the expense of convenience for some operations. **CHANGE** requires naming the column twice if not renaming it, and requires respecifying the column definition if only renaming it.

With **FIRST** or **AFTER**, can reorder columns.

**MODIFY**:

Can change a column definition but not its name.

More convenient than **CHANGE** to change a column definition without renaming it.

With **FIRST** or **AFTER**, can reorder columns.

**RENAME COLUMN**:

Can change a column name but not its definition.

More convenient than **CHANGE** to rename a column without changing its definition.

**ALTER**: Used only to change a column default value.

**CHANGE** is a MySQL extension to standard SQL. **MODIFY** and **RENAME COLUMN** are MySQL extensions for Oracle compatibility.

To alter a column to change both its name and definition, use **CHANGE**, specifying the old and new names and the new definition. For example, to rename an **INT NOT NULL** column from **a** to **b** and change its definition to use the **BIGINT** data type while retaining the **NOT NULL** attribute, do this:

ALTER TABLE t1 CHANGE a b BIGINT NOT NULL;

To change a column definition but not its name, use **CHANGE** or **MODIFY**. With **CHANGE**, the syntax requires two column names, so you must specify the same name twice to leave the name unchanged. For example, to change the definition of column **b**, do this:

ALTER TABLE t1 CHANGE b b INT NOT NULL;

**MODIFY** is more convenient to change the definition without changing the name because it requires the column name only once:

ALTER TABLE t1 MODIFY b INT NOT NULL;

To change a column name but not its definition, use **CHANGE** or **RENAME COLUMN**. With **CHANGE**, the syntax requires a column definition, so to leave the definition unchanged, you must respecify the definition the column currently has. For example, to rename an **INT NOT NULL** column from **b** to **a**, do this:

ALTER TABLE t1 CHANGE b a INT NOT NULL;

**RENAME COLUMN** is more convenient to change the name without changing the definition because it requires only the old and new names:

ALTER TABLE t1 RENAME COLUMN b TO a;

In general, you cannot rename a column to a name that already exists in the table. However, this is sometimes not the case, such as when you swap names or move them through a cycle. If a table has columns named **a**, **b**, and **c**, these are valid operations:

-- swap a and b

ALTER TABLE t1 RENAME COLUMN a TO b,

RENAME COLUMN b TO a;

-- "rotate" a, b, c through a cycle

ALTER TABLE t1 RENAME COLUMN a TO b,

RENAME COLUMN b TO c,

RENAME COLUMN c TO a;

For column definition changes using **CHANGE** or **MODIFY**, the definition must include the data type and all attributes that should apply to the new column, other than index attributes such as **PRIMARY KEY** or **UNIQUE**. Attributes present in the original definition but not specified for the new definition are not carried forward. Suppose that a column **col1** is defined as **INT UNSIGNED DEFAULT 1 COMMENT 'my column'** and you modify the column as follows, intending to change only **INT** to **BIGINT**:

ALTER TABLE t1 MODIFY col1 BIGINT;

That statement changes the data type from **INT** to **BIGINT**, but it also drops the **UNSIGNED**, **DEFAULT**, and **COMMENT** attributes. To retain them, the statement must include them explicitly:

ALTER TABLE t1 MODIFY col1 BIGINT UNSIGNED DEFAULT 1 COMMENT 'my column';

For data type changes using **CHANGE** or **MODIFY**, MySQL tries to convert existing column values to the new type as well as possible.

**Warning**

This conversion may result in alteration of data. For example, if you shorten a string column, values may be truncated. To prevent the operation from succeeding if conversions to the new data type would result in loss of data, enable strict SQL mode before using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) (see [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode)).

If you use **CHANGE** or **MODIFY** to shorten a column for which an index exists on the column, and the resulting column length is less than the index length, MySQL shortens the index automatically.

For columns renamed by **CHANGE** or **RENAME COLUMN**, MySQL automatically renames these references to the renamed column:

Indexes that refer to the old column, including invisible indexes and disabled **MyISAM** indexes.

Foreign keys that refer to the old column.

For columns renamed by **CHANGE** or **RENAME COLUMN**, MySQL does not automatically rename these references to the renamed column:

Generated column and partition expressions that refer to the renamed column. You must use **CHANGE** to redefine such expressions in the same [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement as the one that renames the column.

Views and stored programs that refer to the renamed column. You must manually alter the definition of these objects to refer to the new column name.

To reorder columns within a table, use **FIRST** and **AFTER** in **CHANGE** or **MODIFY** operations.

**ALTER ... SET DEFAULT** or **ALTER ... DROP DEFAULT** specify a new default value for a column or remove the old default value, respectively. If the old default is removed and the column can be **NULL**, the new default is **NULL**. If the column cannot be **NULL**, MySQL assigns a default value as described in [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults).

As of MySQL 8.0.23, **ALTER ... SET VISIBLE** and **ALTER ... SET INVISIBLE** enable column visibility to be changed. See [Section 13.1.20.10, “Invisible Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#invisible-columns).

#### Primary Keys and Indexes

**DROP PRIMARY KEY** drops the [primary key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_primary_key). If there is no primary key, an error occurs. For information about the performance characteristics of primary keys, especially for **InnoDB** tables, see [Section 8.3.2, “Primary Key Optimization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#primary-key-optimization).

If the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable is enabled, attempting to drop a primary key produces an error.

If you add a **UNIQUE INDEX** or **PRIMARY KEY** to a table, MySQL stores it before any nonunique index to permit detection of duplicate keys as early as possible.

**[DROP INDEX](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "drop-index" \o "13.1.27 DROP INDEX Statement)** removes an index. This is a MySQL extension to standard SQL. See [Section 13.1.27, “DROP INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-index). To determine index names, use **SHOW INDEX FROM *tbl\_name***.

Some storage engines permit you to specify an index type when creating an index. The syntax for the ***index\_type*** specifier is **USING *type\_name***. For details about **USING**, see [Section 13.1.15, “CREATE INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index). The preferred position is after the column list. Expect support for use of the option before the column list to be removed in a future MySQL release.

***index\_option*** values specify additional options for an index. **USING** is one such option. For details about permissible ***index\_option*** values, see [Section 13.1.15, “CREATE INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index).

**RENAME INDEX *old\_index\_name* TO *new\_index\_name*** renames an index. This is a MySQL extension to standard SQL. The content of the table remains unchanged. ***old\_index\_name*** must be the name of an existing index in the table that is not dropped by the same [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. ***new\_index\_name*** is the new index name, which cannot duplicate the name of an index in the resulting table after changes have been applied. Neither index name can be **PRIMARY**.

If you use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on a **MyISAM** table, all nonunique indexes are created in a separate batch (as for [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table)). This should make [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) much faster when you have many indexes.

For **MyISAM** tables, key updating can be controlled explicitly. Use **ALTER TABLE ... DISABLE KEYS** to tell MySQL to stop updating nonunique indexes. Then use **ALTER TABLE ... ENABLE KEYS** to re-create missing indexes. **MyISAM** does this with a special algorithm that is much faster than inserting keys one by one, so disabling keys before performing bulk insert operations should give a considerable speedup. Using **ALTER TABLE ... DISABLE KEYS** requires the [**INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_index) privilege in addition to the privileges mentioned earlier.

While the nonunique indexes are disabled, they are ignored for statements such as [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) that otherwise would use them.

After an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement, it may be necessary to run [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) to update index cardinality information. See [Section 13.7.7.22, “SHOW INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index).

The **ALTER INDEX** operation permits an index to be made visible or invisible. An invisible index is not used by the optimizer. Modification of index visibility applies to indexes other than primary keys (either explicit or implicit). This feature is storage engine neutral (supported for any engine). For more information, see [Section 8.3.12, “Invisible Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#invisible-indexes).

#### Foreign Keys and Other Constraints

The **FOREIGN KEY** and **REFERENCES** clauses are supported by the **InnoDB** and **NDB** storage engines, which implement **ADD [CONSTRAINT [*symbol*]] FOREIGN KEY [*index\_name*] (...) REFERENCES ... (...)**. See [Section 13.1.20.5, “FOREIGN KEY Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys). For other storage engines, the clauses are parsed but ignored.

For [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), unlike [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), **ADD FOREIGN KEY** ignores ***index\_name*** if given and uses an automatically generated foreign key name. As a workaround, include the **CONSTRAINT** clause to specify the foreign key name:

ADD CONSTRAINT ***name*** FOREIGN KEY (....) ...

**Important**

MySQL silently ignores inline **REFERENCES** specifications, where the references are defined as part of the column specification. MySQL accepts only **REFERENCES** clauses defined as part of a separate **FOREIGN KEY** specification.

**Note**

Partitioned **InnoDB** tables do not support foreign keys. This restriction does not apply to **NDB** tables, including those explicitly partitioned by **[LINEAR] KEY**. For more information, see [Section 24.6.2, “Partitioning Limitations Relating to Storage Engines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-limitations-storage-engines).

MySQL Server and NDB Cluster both support the use of [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to drop foreign keys:

ALTER TABLE ***tbl\_name*** DROP FOREIGN KEY ***fk\_symbol***;

Adding and dropping a foreign key in the same [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement is supported for [**ALTER TABLE ... ALGORITHM=INPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) but not for [**ALTER TABLE ... ALGORITHM=COPY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

The server prohibits changes to foreign key columns that have the potential to cause loss of referential integrity. A workaround is to use [**ALTER TABLE ... DROP FOREIGN KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) before changing the column definition and [**ALTER TABLE ... ADD FOREIGN KEY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) afterward. Examples of prohibited changes include:

Changes to the data type of foreign key columns that may be unsafe. For example, changing [**VARCHAR(20)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) to [**VARCHAR(30)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) is permitted, but changing it to [**VARCHAR(1024)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) is not because that alters the number of length bytes required to store individual values.

Changing a **NULL** column to **NOT NULL** in non-strict mode is prohibited to prevent converting **NULL** values to default non-**NULL** values, for which there are no corresponding values in the referenced table. The operation is permitted in strict mode, but an error is returned if any such conversion is required.

**ALTER TABLE *tbl\_name* RENAME *new\_tbl\_name*** changes internally generated foreign key constraint names and user-defined foreign key constraint names that begin with the string “***tbl\_name***\_ibfk\_” to reflect the new table name. **InnoDB** interprets foreign key constraint names that begin with the string “***tbl\_name***\_ibfk\_” as internally generated names.

Prior to MySQL 8.0.16, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) permits only the following limited version of **CHECK** constraint-adding syntax, which is parsed and ignored:

ADD CHECK (***expr***)

As of MySQL 8.0.16, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) permits **CHECK** constraints for existing tables to be added, dropped, or altered:

Add a new **CHECK** constraint:

ALTER TABLE ***tbl\_name***

ADD CONSTRAINT [***symbol***] CHECK (***expr***) [[NOT] ENFORCED];

The meaning of constraint syntax elements is the same as for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). See [Section 13.1.20.6, “CHECK Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-check-constraints).

Drop an existing **CHECK** constraint named ***symbol***:

ALTER TABLE ***tbl\_name***

DROP CHECK ***symbol***;

Alter whether an existing **CHECK** constraint named ***symbol*** is enforced:

ALTER TABLE ***tbl\_name***

ALTER CHECK ***symbol*** [NOT] ENFORCED;

The **DROP CHECK** and **ALTER CHECK** clauses are MySQL extensions to standard SQL.

As of MySQL 8.0.19, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) permits more general (and SQL standard) syntax for dropping and altering existing constraints of any type, where the constraint type is determined from the constraint name:

Drop an existing constraint named ***symbol***:

ALTER TABLE ***tbl\_name***

DROP CONSTRAINT ***symbol***;

If the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable is enabled, attempting to drop a primary key produces an error.

Alter whether an existing constraint named ***symbol*** is enforced:

ALTER TABLE ***tbl\_name***

ALTER CONSTRAINT ***symbol*** [NOT] ENFORCED;

Only **CHECK** constraints can be altered to be unenforced. All other constraint types are always enforced.

The SQL standard specifies that all types of constraints (primary key, unique index, foreign key, check) belong to the same namespace. In MySQL, each constraint type has its own namespace per schema. Consequently, names for each type of constraint must be unique per schema, but constraints of different types can have the same name. When multiple constraints have the same name, **DROP CONSTRAINT** and **ADD CONSTRAINT** are ambiguous and an error occurs. In such cases, constraint-specific syntax must be used to modify the constraint. For example, use **DROP PRIMARY KEY** or DROP FOREIGN KEY to drop a primary key or foreign key.

If a table alteration causes a violation of an enforced **CHECK** constraint, an error occurs and the table is not modified. Examples of operations for which an error occurs:

Attempts to add the **AUTO\_INCREMENT** attribute to a column that is used in a **CHECK** constraint.

Attempts to add an enforced **CHECK** constraint or enforce a nonenforced **CHECK** constraint for which existing rows violate the constraint condition.

Attempts to modify, rename, or drop a column that is used in a **CHECK** constraint, unless that constraint is also dropped in the same statement. Exception: If a **CHECK** constraint refers only to a single column, dropping the column automatically drops the constraint.

**ALTER TABLE *tbl\_name* RENAME *new\_tbl\_name*** changes internally generated and user-defined **CHECK** constraint names that begin with the string “***tbl\_name***\_chk\_” to reflect the new table name. MySQL interprets **CHECK** constraint names that begin with the string “***tbl\_name***\_chk\_” as internally generated names.

#### Changing the Character Set

To change the table default character set and all character columns ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)) to a new character set, use a statement like this:

ALTER TABLE ***tbl\_name*** CONVERT TO CHARACTER SET ***charset\_name***;

The statement also changes the collation of all character columns. If you specify no **COLLATE** clause to indicate which collation to use, the statement uses default collation for the character set. If this collation is inappropriate for the intended table use (for example, if it would change from a case-sensitive collation to a case-insensitive collation), specify a collation explicitly.

For a column that has a data type of [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) or one of the [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types, **CONVERT TO CHARACTER SET** changes the data type as necessary to ensure that the new column is long enough to store as many characters as the original column. For example, a [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column has two length bytes, which store the byte-length of values in the column, up to a maximum of 65,535. For a **latin1** [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column, each character requires a single byte, so the column can store up to 65,535 characters. If the column is converted to **utf8**, each character might require up to three bytes, for a maximum possible length of 3 × 65,535 = 196,605 bytes. That length does not fit in a [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column's length bytes, so MySQL converts the data type to [**MEDIUMTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), which is the smallest string type for which the length bytes can record a value of 196,605. Similarly, a [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column might be converted to [**MEDIUMTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob).

To avoid data type changes of the type just described, do not use **CONVERT TO CHARACTER SET**. Instead, use **MODIFY** to change individual columns. For example:

ALTER TABLE t MODIFY latin1\_text\_col TEXT CHARACTER SET utf8;

ALTER TABLE t MODIFY latin1\_varchar\_col VARCHAR(***M***) CHARACTER SET utf8;

If you specify **CONVERT TO CHARACTER SET binary**, the [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns are converted to their corresponding binary string types ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)). This means that the columns no longer have a character set and a subsequent **CONVERT TO** operation does not apply to them.

If ***charset\_name*** is **DEFAULT** in a **CONVERT TO CHARACTER SET** operation, the character set named by the [**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database) system variable is used.

**Warning**

The **CONVERT TO** operation converts column values between the original and named character sets. This is not what you want if you have a column in one character set (like **latin1**) but the stored values actually use some other, incompatible character set (like **utf8**). In this case, you have to do the following for each such column:

ALTER TABLE t1 CHANGE c1 c1 BLOB;

ALTER TABLE t1 CHANGE c1 c1 TEXT CHARACTER SET utf8;

The reason this works is that there is no conversion when you convert to or from [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns.

To change only the default character set for a table, use this statement:

ALTER TABLE ***tbl\_name*** DEFAULT CHARACTER SET ***charset\_name***;

The word **DEFAULT** is optional. The default character set is the character set that is used if you do not specify the character set for columns that you add to a table later (for example, with **ALTER TABLE ... ADD column**).

When the [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) system variable is enabled, which is the default setting, character set conversion is not permitted on tables that include a character string column used in a foreign key constraint. The workaround is to disable [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) before performing the character set conversion. You must perform the conversion on both tables involved in the foreign key constraint before re-enabling [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks). If you re-enable [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) after converting only one of the tables, an **ON DELETE CASCADE** or **ON UPDATE CASCADE** operation could corrupt data in the referencing table due to implicit conversion that occurs during these operations (Bug #45290, Bug #74816).

#### Importing InnoDB Tables

An **InnoDB** table created in its own [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespace can be imported from a backup or from another MySQL server instance using **DISCARD TABLEPACE** and **IMPORT TABLESPACE** clauses. See [Section 15.6.1.3, “Importing InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import).

#### Row Order for MyISAM Tables

**ORDER BY** enables you to create the new table with the rows in a specific order. This option is useful primarily when you know that you query the rows in a certain order most of the time. By using this option after major changes to the table, you might be able to get higher performance. In some cases, it might make sorting easier for MySQL if the table is in order by the column that you want to order it by later.

**Note**

The table does not remain in the specified order after inserts and deletes.

**ORDER BY** syntax permits one or more column names to be specified for sorting, each of which optionally can be followed by **ASC** or **DESC** to indicate ascending or descending sort order, respectively. The default is ascending order. Only column names are permitted as sort criteria; arbitrary expressions are not permitted. This clause should be given last after any other clauses.

**ORDER BY** does not make sense for **InnoDB** tables because **InnoDB** always orders table rows according to the [clustered index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_clustered_index).

When used on a partitioned table, **ALTER TABLE ... ORDER BY** orders rows within each partition only.

#### Partitioning Options

***partition\_options*** signifies options that can be used with partitioned tables for repartitioning, to add, drop, discard, import, merge, and split partitions, and to perform partitioning maintenance.

It is possible for an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement to contain a **PARTITION BY** or **REMOVE PARTITIONING** clause in an addition to other alter specifications, but the **PARTITION BY** or **REMOVE PARTITIONING** clause must be specified last after any other specifications. The **ADD PARTITION**, **DROP PARTITION**, **DISCARD PARTITION**, **IMPORT PARTITION**, **COALESCE PARTITION**, **REORGANIZE PARTITION**, **EXCHANGE PARTITION**, **ANALYZE PARTITION**, **CHECK PARTITION**, and **REPAIR PARTITION** options cannot be combined with other alter specifications in a single **ALTER TABLE**, since the options just listed act on individual partitions.

For more information about partition options, see [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), and [Section 13.1.9.1, “ALTER TABLE Partition Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-partition-operations). For information about and examples of **ALTER TABLE ... EXCHANGE PARTITION** statements, see [Section 24.3.3, “Exchanging Partitions and Subpartitions with Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-management-exchange).

#### 13.1.9.1 ALTER TABLE Partition Operations

Partitioning-related clauses for [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) can be used with partitioned tables for repartitioning, to add, drop, discard, import, merge, and split partitions, and to perform partitioning maintenance.

Simply using a ***partition\_options*** clause with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on a partitioned table repartitions the table according to the partitioning scheme defined by the ***partition\_options***. This clause always begins with **PARTITION BY**, and follows the same syntax and other rules as apply to the ***partition\_options*** clause for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) (for more detailed information, see [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table)), and can also be used to partition an existing table that is not already partitioned. For example, consider a (nonpartitioned) table defined as shown here:

CREATE TABLE t1 (

id INT,

year\_col INT

);

This table can be partitioned by **HASH**, using the **id** column as the partitioning key, into 8 partitions by means of this statement:

ALTER TABLE t1

PARTITION BY HASH(id)

PARTITIONS 8;

MySQL supports an **ALGORITHM** option with **[SUB]PARTITION BY [LINEAR] KEY**. **ALGORITHM=1** causes the server to use the same key-hashing functions as MySQL 5.1 when computing the placement of rows in partitions; **ALGORITHM=2** means that the server employs the key-hashing functions implemented and used by default for new **KEY** partitioned tables in MySQL 5.5 and later. (Partitioned tables created with the key-hashing functions employed in MySQL 5.5 and later cannot be used by a MySQL 5.1 server.) Not specifying the option has the same effect as using **ALGORITHM=2**. This option is intended for use chiefly when upgrading or downgrading **[LINEAR] KEY** partitioned tables between MySQL 5.1 and later MySQL versions, or for creating tables partitioned by **KEY** or **LINEAR KEY** on a MySQL 5.5 or later server which can be used on a MySQL 5.1 server.

The table that results from using an **ALTER TABLE ... PARTITION BY** statement must follow the same rules as one created using **CREATE TABLE ... PARTITION BY**. This includes the rules governing the relationship between any unique keys (including any primary key) that the table might have, and the column or columns used in the partitioning expression, as discussed in [Section 24.6.1, “Partitioning Keys, Primary Keys, and Unique Keys”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-limitations-partitioning-keys-unique-keys). The **CREATE TABLE ... PARTITION BY** rules for specifying the number of partitions also apply to **ALTER TABLE ... PARTITION BY**.

The ***partition\_definition*** clause for **ALTER TABLE ADD PARTITION** supports the same options as the clause of the same name for the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement. (See [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), for the syntax and description.) Suppose that you have the partitioned table created as shown here:

CREATE TABLE t1 (

id INT,

year\_col INT

)

PARTITION BY RANGE (year\_col) (

PARTITION p0 VALUES LESS THAN (1991),

PARTITION p1 VALUES LESS THAN (1995),

PARTITION p2 VALUES LESS THAN (1999)

);

You can add a new partition **p3** to this table for storing values less than **2002** as follows:

ALTER TABLE t1 ADD PARTITION (PARTITION p3 VALUES LESS THAN (2002));

**DROP PARTITION** can be used to drop one or more **RANGE** or **LIST** partitions. This statement cannot be used with **HASH** or **KEY** partitions; instead, use **COALESCE PARTITION** (see later in this section). Any data that was stored in the dropped partitions named in the ***partition\_names*** list is discarded. For example, given the table **t1** defined previously, you can drop the partitions named **p0** and **p1** as shown here:

ALTER TABLE t1 DROP PARTITION p0, p1;

**Note**

**DROP PARTITION** does not work with tables that use the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine. See [Section 24.3.1, “Management of RANGE and LIST Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-management-range-list), and [Section 23.1.7, “Known Limitations of NDB Cluster”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-limitations).

**ADD PARTITION** and **DROP PARTITION** do not currently support **IF [NOT] EXISTS**.

The [**DISCARD PARTITION ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**IMPORT PARTITION ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) options extend the [Transportable Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transportable_tablespace) feature to individual **InnoDB** table partitions. Each **InnoDB** table partition has its own tablespace file (.ibd file). The [Transportable Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transportable_tablespace) feature makes it easy to copy the tablespaces from a running MySQL server instance to another running instance, or to perform a restore on the same instance. Both options take a comma-separated list of one or more partition names. For example:

ALTER TABLE t1 DISCARD PARTITION p2, p3 TABLESPACE;

ALTER TABLE t1 IMPORT PARTITION p2, p3 TABLESPACE;

When running [**DISCARD PARTITION ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**IMPORT PARTITION ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on subpartitioned tables, both partition and subpartition names are allowed. When a partition name is specified, subpartitions of that partition are included.

The [Transportable Tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transportable_tablespace) feature also supports copying or restoring partitioned **InnoDB** tables. For more information, see [Section 15.6.1.3, “Importing InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import).

Renames of partitioned tables are supported. You can rename individual partitions indirectly using **ALTER TABLE ... REORGANIZE PARTITION**; however, this operation copies the partition's data.

To delete rows from selected partitions, use the **TRUNCATE PARTITION** option. This option takes a list of one or more comma-separated partition names. Consider the table **t1** created by this statement:

CREATE TABLE t1 (

id INT,

year\_col INT

)

PARTITION BY RANGE (year\_col) (

PARTITION p0 VALUES LESS THAN (1991),

PARTITION p1 VALUES LESS THAN (1995),

PARTITION p2 VALUES LESS THAN (1999),

PARTITION p3 VALUES LESS THAN (2003),

PARTITION p4 VALUES LESS THAN (2007)

);

To delete all rows from partition **p0**, use the following statement:

ALTER TABLE t1 TRUNCATE PARTITION p0;

The statement just shown has the same effect as the following [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement:

DELETE FROM t1 WHERE year\_col < 1991;

When truncating multiple partitions, the partitions do not have to be contiguous: This can greatly simplify delete operations on partitioned tables that would otherwise require very complex **WHERE** conditions if done with [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements. For example, this statement deletes all rows from partitions **p1** and **p3**:

ALTER TABLE t1 TRUNCATE PARTITION p1, p3;

An equivalent [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement is shown here:

DELETE FROM t1 WHERE

(year\_col >= 1991 AND year\_col < 1995)

OR

(year\_col >= 2003 AND year\_col < 2007);

If you use the **ALL** keyword in place of the list of partition names, the statement acts on all table partitions.

**TRUNCATE PARTITION** merely deletes rows; it does not alter the definition of the table itself, or of any of its partitions.

To verify that the rows were dropped, check the **INFORMATION\_SCHEMA.PARTITIONS** table, using a query such as this one:

SELECT PARTITION\_NAME, TABLE\_ROWS

FROM INFORMATION\_SCHEMA.PARTITIONS

WHERE TABLE\_NAME = 't1';

**COALESCE PARTITION** can be used with a table that is partitioned by **HASH** or **KEY** to reduce the number of partitions by ***number***. Suppose that you have created table **t2** as follows:

CREATE TABLE t2 (

name VARCHAR (30),

started DATE

)

PARTITION BY HASH( YEAR(started) )

PARTITIONS 6;

To reduce the number of partitions used by **t2** from 6 to 4, use the following statement:

ALTER TABLE t2 COALESCE PARTITION 2;

The data contained in the last ***number*** partitions is merged into the remaining partitions. In this case, partitions 4 and 5 are merged into the first 4 partitions (the partitions numbered 0, 1, 2, and 3).

To change some but not all the partitions used by a partitioned table, you can use **REORGANIZE PARTITION**. This statement can be used in several ways:

To merge a set of partitions into a single partition. This is done by naming several partitions in the ***partition\_names*** list and supplying a single definition for ***partition\_definition***.

To split an existing partition into several partitions. Accomplish this by naming a single partition for ***partition\_names*** and providing multiple ***partition\_definitions***.

To change the ranges for a subset of partitions defined using **VALUES LESS THAN** or the value lists for a subset of partitions defined using **VALUES IN**.

**Note**

For partitions that have not been explicitly named, MySQL automatically provides the default names **p0**, **p1**, **p2**, and so on. The same is true with regard to subpartitions.

For more detailed information about and examples of **ALTER TABLE ... REORGANIZE PARTITION** statements, see [Section 24.3.1, “Management of RANGE and LIST Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-management-range-list).

To exchange a table partition or subpartition with a table, use the [**ALTER TABLE ... EXCHANGE PARTITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement—that is, to move any existing rows in the partition or subpartition to the nonpartitioned table, and any existing rows in the nonpartitioned table to the table partition or subpartition.

For usage information and examples, see [Section 24.3.3, “Exchanging Partitions and Subpartitions with Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-management-exchange).

Several options provide partition maintenance and repair functionality analogous to that implemented for nonpartitioned tables by statements such as [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) and [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) (which are also supported for partitioned tables; for more information, see [Section 13.7.3, “Table Maintenance Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table-maintenance-statements)). These include **ANALYZE PARTITION**, **CHECK PARTITION**, **OPTIMIZE PARTITION**, **REBUILD PARTITION**, and **REPAIR PARTITION**. Each of these options takes a ***partition\_names*** clause consisting of one or more names of partitions, separated by commas. The partitions must already exist in the target table. You can also use the **ALL** keyword in place of ***partition\_names***, in which case the statement acts on all table partitions. For more information and examples, see [Section 24.3.4, “Maintenance of Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-maintenance).

[**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) does not currently support per-partition optimization; **ALTER TABLE ... OPTIMIZE PARTITION** causes the entire table to rebuilt and analyzed, and an appropriate warning to be issued. (Bug #11751825, Bug #42822) To work around this problem, use **ALTER TABLE ... REBUILD PARTITION** and **ALTER TABLE ... ANALYZE PARTITION** instead.

The **ANALYZE PARTITION**, **CHECK PARTITION**, **OPTIMIZE PARTITION**, and **REPAIR PARTITION** options are not supported for tables which are not partitioned.

**REMOVE PARTITIONING** enables you to remove a table's partitioning without otherwise affecting the table or its data. This option can be combined with other [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) options such as those used to add, drop, or rename columns or indexes.

Using the **ENGINE** option with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) changes the storage engine used by the table without affecting the partitioning. The target storage engine must provide its own partitioning handler. Only the **InnoDB** and **NDB** storage engines have native partitioning handlers; **NDB** is not currently supported in MySQL 8.0.

It is possible for an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement to contain a **PARTITION BY** or **REMOVE PARTITIONING** clause in an addition to other alter specifications, but the **PARTITION BY** or **REMOVE PARTITIONING** clause must be specified last after any other specifications.

The **ADD PARTITION**, **DROP PARTITION**, **COALESCE PARTITION**, **REORGANIZE PARTITION**, **ANALYZE PARTITION**, **CHECK PARTITION**, and **REPAIR PARTITION** options cannot be combined with other alter specifications in a single **ALTER TABLE**, since the options just listed act on individual partitions. For more information, see [Section 13.1.9.1, “ALTER TABLE Partition Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-partition-operations).

Only a single instance of any one of the following options can be used in a given [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement: **PARTITION BY**, **ADD PARTITION**, **DROP PARTITION**, **TRUNCATE PARTITION**, **EXCHANGE PARTITION**, **REORGANIZE PARTITION**, or **COALESCE PARTITION**, **ANALYZE PARTITION**, **CHECK PARTITION**, **OPTIMIZE PARTITION**, **REBUILD PARTITION**, **REMOVE PARTITIONING**.

For example, the following two statements are invalid:

ALTER TABLE t1 ANALYZE PARTITION p1, ANALYZE PARTITION p2;

ALTER TABLE t1 ANALYZE PARTITION p1, CHECK PARTITION p2;

In the first case, you can analyze partitions **p1** and **p2** of table **t1** concurrently using a single statement with a single **ANALYZE PARTITION** option that lists both of the partitions to be analyzed, like this:

ALTER TABLE t1 ANALYZE PARTITION p1, p2;

In the second case, it is not possible to perform **ANALYZE** and **CHECK** operations on different partitions of the same table concurrently. Instead, you must issue two separate statements, like this:

ALTER TABLE t1 ANALYZE PARTITION p1;

ALTER TABLE t1 CHECK PARTITION p2;

**REBUILD** operations are currently unsupported for subpartitions. The **REBUILD** keyword is expressly disallowed with subpartitions, and causes **ALTER TABLE** to fail with an error if so used.

**CHECK PARTITION**and**REPAIR PARTITION** operations fail when the partition to be checked or repaired contains any duplicate key errors.

For more information about these statements, see [Section 24.3.4, “Maintenance of Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-maintenance).

#### 13.1.9.2 ALTER TABLE and Generated Columns

**ALTER TABLE** operations permitted for generated columns are **ADD**, **MODIFY**, and **CHANGE**.

Generated columns can be added.

CREATE TABLE t1 (c1 INT);

ALTER TABLE t1 ADD COLUMN c2 INT GENERATED ALWAYS AS (c1 + 1) STORED;

The data type and expression of generated columns can be modified.

CREATE TABLE t1 (c1 INT, c2 INT GENERATED ALWAYS AS (c1 + 1) STORED);

ALTER TABLE t1 MODIFY COLUMN c2 TINYINT GENERATED ALWAYS AS (c1 + 5) STORED;

Generated columns can be renamed or dropped, if no other column refers to them.

CREATE TABLE t1 (c1 INT, c2 INT GENERATED ALWAYS AS (c1 + 1) STORED);

ALTER TABLE t1 CHANGE c2 c3 INT GENERATED ALWAYS AS (c1 + 1) STORED;

ALTER TABLE t1 DROP COLUMN c3;

Virtual generated columns cannot be altered to stored generated columns, or vice versa. To work around this, drop the column, then add it with the new definition.

CREATE TABLE t1 (c1 INT, c2 INT GENERATED ALWAYS AS (c1 + 1) VIRTUAL);

ALTER TABLE t1 DROP COLUMN c2;

ALTER TABLE t1 ADD COLUMN c2 INT GENERATED ALWAYS AS (c1 + 1) STORED;

Nongenerated columns can be altered to stored but not virtual generated columns.

CREATE TABLE t1 (c1 INT, c2 INT);

ALTER TABLE t1 MODIFY COLUMN c2 INT GENERATED ALWAYS AS (c1 + 1) STORED;

Stored but not virtual generated columns can be altered to nongenerated columns. The stored generated values become the values of the nongenerated column.

CREATE TABLE t1 (c1 INT, c2 INT GENERATED ALWAYS AS (c1 + 1) STORED);

ALTER TABLE t1 MODIFY COLUMN c2 INT;

**ADD COLUMN** is not an in-place operation for stored columns (done without using a temporary table) because the expression must be evaluated by the server. For stored columns, indexing changes are done in place, and expression changes are not done in place. Changes to column comments are done in place.

For non-partitioned tables, **ADD COLUMN** and **DROP COLUMN** are in-place operations for virtual columns. However, adding or dropping a virtual column cannot be performed in place in combination with other [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations.

For partitioned tables, **ADD COLUMN** and **DROP COLUMN** are not in-place operations for virtual columns.

**InnoDB** supports secondary indexes on virtual generated columns. Adding or dropping a secondary index on a virtual generated column is an in-place operation. For more information, see [Section 13.1.20.9, “Secondary Indexes and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-secondary-indexes).

When a **VIRTUAL** generated column is added to a table or modified, it is not ensured that data being calculated by the generated column expression is be out of range for the column. This can lead to inconsistent data being returned and unexpectedly failed statements. To permit control over whether validation occurs for such columns, **ALTER TABLE** supports **WITHOUT VALIDATION** and **WITH VALIDATION** clauses:

With **WITHOUT VALIDATION** (the default if neither clause is specified), an in-place operation is performed (if possible), data integrity is not checked, and the statement finishes more quickly. However, later reads from the table might report warnings or errors for the column if values are out of range.

With **WITH VALIDATION**, **ALTER TABLE** copies the table. If an out-of-range or any other error occurs, the statement fails. Because a table copy is performed, the statement takes longer.

**WITHOUT VALIDATION** and **WITH VALIDATION** are permitted only with **ADD COLUMN**, **CHANGE COLUMN**, and **MODIFY COLUMN** operations. Otherwise, an [**ER\_WRONG\_USAGE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_usage) error occurs.

If expression evaluation causes truncation or provides incorrect input to a function, the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement terminates with an error and the DDL operation is rejected.

An [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement that changes the default value of a column ***col\_name*** may also change the value of a generated column expression that refers to the column using ***col\_name***, which may change the value of a generated column expression that refers to the column using [**DEFAULT(*col\_name*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_default). For this reason, [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that change the definition of a column cause a table rebuild if any generated column expression uses [**DEFAULT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_default).

#### 13.1.9.3 ALTER TABLE Examples

Begin with a table **t1** created as shown here:

CREATE TABLE t1 (a INTEGER, b CHAR(10));

To rename the table from **t1** to **t2**:

ALTER TABLE t1 RENAME t2;

To change column **a** from [**INTEGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) to **TINYINT NOT NULL** (leaving the name the same), and to change column **b** from **CHAR(10)** to **CHAR(20)** as well as renaming it from **b** to **c**:

ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);

To add a new [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) column named **d**:

ALTER TABLE t2 ADD d TIMESTAMP;

To add an index on column **d** and a **UNIQUE** index on column **a**:

ALTER TABLE t2 ADD INDEX (d), ADD UNIQUE (a);

To remove column **c**:

ALTER TABLE t2 DROP COLUMN c;

To add a new **AUTO\_INCREMENT** integer column named **c**:

ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO\_INCREMENT,

ADD PRIMARY KEY (c);

We indexed **c** (as a **PRIMARY KEY**) because **AUTO\_INCREMENT** columns must be indexed, and we declare **c** as **NOT NULL** because primary key columns cannot be **NULL**.

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, it is also possible to change the storage type used for a table or column. For example, consider an [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) table created as shown here:

mysql> **CREATE TABLE t1 (c1 INT) TABLESPACE ts\_1 ENGINE NDB;**

Query OK, 0 rows affected (1.27 sec)

To convert this table to disk-based storage, you can use the following [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement:

mysql> **ALTER TABLE t1 TABLESPACE ts\_1 STORAGE DISK;**

Query OK, 0 rows affected (2.99 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> **SHOW CREATE TABLE t1\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t1

Create Table: CREATE TABLE `t1` (

`c1` int(11) DEFAULT NULL

) /\*!50100 TABLESPACE ts\_1 STORAGE DISK \*/

ENGINE=ndbcluster DEFAULT CHARSET=latin1

1 row in set (0.01 sec)

It is not necessary that the tablespace was referenced when the table was originally created; however, the tablespace must be referenced by the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table):

mysql> **CREATE TABLE t2 (c1 INT) ts\_1 ENGINE NDB;**

Query OK, 0 rows affected (1.00 sec)

mysql> **ALTER TABLE t2 STORAGE DISK;**

ERROR 1005 (HY000): Can't create table 'c.#sql-1750\_3' (errno: 140)

mysql> **ALTER TABLE t2 TABLESPACE ts\_1 STORAGE DISK;**

Query OK, 0 rows affected (3.42 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> **SHOW CREATE TABLE t2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t1

Create Table: CREATE TABLE `t2` (

`c1` int(11) DEFAULT NULL

) /\*!50100 TABLESPACE ts\_1 STORAGE DISK \*/

ENGINE=ndbcluster DEFAULT CHARSET=latin1

1 row in set (0.01 sec)

To change the storage type of an individual column, you can use **ALTER TABLE ... MODIFY [COLUMN]**. For example, suppose you create an NDB Cluster Disk Data table with two columns, using this [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement:

mysql> **CREATE TABLE t3 (c1 INT, c2 INT)**

-> **TABLESPACE ts\_1 STORAGE DISK ENGINE NDB;**

Query OK, 0 rows affected (1.34 sec)

To change column **c2** from disk-based to in-memory storage, include a STORAGE MEMORY clause in the column definition used by the ALTER TABLE statement, as shown here:

mysql> **ALTER TABLE t3 MODIFY c2 INT STORAGE MEMORY;**

Query OK, 0 rows affected (3.14 sec)

Records: 0 Duplicates: 0 Warnings: 0

You can make an in-memory column into a disk-based column by using **STORAGE DISK** in a similar fashion.

Column **c1** uses disk-based storage, since this is the default for the table (determined by the table-level **STORAGE DISK** clause in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement). However, column **c2** uses in-memory storage, as can be seen here in the output of SHOW [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table):

mysql> **SHOW CREATE TABLE t3\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t3

Create Table: CREATE TABLE `t3` (

`c1` int(11) DEFAULT NULL,

`c2` int(11) /\*!50120 STORAGE MEMORY \*/ DEFAULT NULL

) /\*!50100 TABLESPACE ts\_1 STORAGE DISK \*/ ENGINE=ndbcluster DEFAULT CHARSET=latin1

1 row in set (0.02 sec)

When you add an **AUTO\_INCREMENT** column, column values are filled in with sequence numbers automatically. For **MyISAM** tables, you can set the first sequence number by executing **SET INSERT\_ID=*value*** before [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) or by using the **AUTO\_INCREMENT=*value*** table option.

With **MyISAM** tables, if you do not change the **AUTO\_INCREMENT** column, the sequence number is not affected. If you drop an **AUTO\_INCREMENT** column and then add another **AUTO\_INCREMENT** column, the numbers are resequenced beginning with 1.

When replication is used, adding an **AUTO\_INCREMENT** column to a table might not produce the same ordering of the rows on the replica and the source. This occurs because the order in which the rows are numbered depends on the specific storage engine used for the table and the order in which the rows were inserted. If it is important to have the same order on the source and replica, the rows must be ordered before assigning an **AUTO\_INCREMENT** number. Assuming that you want to add an **AUTO\_INCREMENT** column to the table **t1**, the following statements produce a new table **t2** identical to **t1** but with an **AUTO\_INCREMENT** column:

CREATE TABLE t2 (id INT AUTO\_INCREMENT PRIMARY KEY)

SELECT \* FROM t1 ORDER BY col1, col2;

This assumes that the table **t1** has columns **col1** and **col2**.

This set of statements also produces a new table **t2** identical to **t1**, with the addition of an **AUTO\_INCREMENT** column:

CREATE TABLE t2 LIKE t1;

ALTER TABLE t2 ADD id INT AUTO\_INCREMENT PRIMARY KEY;

INSERT INTO t2 SELECT \* FROM t1 ORDER BY col1, col2;

**Important**

To guarantee the same ordering on both source and replica, all columns of **t1** must be referenced in the **ORDER BY** clause.

Regardless of the method used to create and populate the copy having the **AUTO\_INCREMENT** column, the final step is to drop the original table and then rename the copy:

DROP TABLE t1;

ALTER TABLE t2 RENAME t1;

### 13.1.10 ALTER TABLESPACE Statement

ALTER [UNDO] TABLESPACE ***tablespace\_name***

NDB only:

{ADD | DROP} DATAFILE '***file\_name***'

[INITIAL\_SIZE [=] size]

[WAIT]

InnoDB and NDB:

[RENAME TO ***tablespace\_name***]

InnoDB only:

[AUTOEXTEND\_SIZE [=] '***value***']

[SET {ACTIVE | INACTIVE}]

[ENCRYPTION [=] {'Y' | 'N'}]

InnoDB and NDB:

[ENGINE [=] ***engine\_name***]

Reserved for future use:

[ENGINE\_ATTRIBUTE [=] '***string***']

This statement is used with **NDB** and **InnoDB** tablespaces. It can be used to add a new data file to, or to drop a data file from an **NDB** tablespace. It can also be used to rename an NDB Cluster Disk Data tablespace, rename an **InnoDB** general tablespace, encrypt an **InnoDB** general tablespace, or mark an **InnoDB** undo tablespace as active or inactive.

The **UNDO** keyword, introduced in MySQL 8.0.14, is used with the **SET {ACTIVE | INACTIVE}** clause to mark an **InnoDB** undo tablespace as active or inactive. For more information, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

The **ADD DATAFILE** variant enables you to specify an initial size for an **NDB** Disk Data tablespace using an **INITIAL\_SIZE** clause, where ***size*** is measured in bytes; the default value is 134217728 (128 MB). You may optionally follow ***size*** with a one-letter abbreviation for an order of magnitude, similar to those used in my.cnf. Generally, this is one of the letters **M** (megabytes) or **G** (gigabytes).

On 32-bit systems, the maximum supported value for **INITIAL\_SIZE** is 4294967296 (4 GB). (Bug #29186)

**INITIAL\_SIZE** is rounded, explicitly, as for [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace).

Once a data file has been created, its size cannot be changed; however, you can add more data files to an NDB tablespace using additional **ALTER TABLESPACE ... ADD DATAFILE** statements.

When **ALTER TABLESPACE ... ADD DATAFILE** is used with **ENGINE = NDB**, a data file is created on each Cluster data node, but only one row is generated in the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table. See the description of this table, as well as [Section 23.5.10.1, “NDB Cluster Disk Data Objects”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data-objects), for more information. **ADD DATAFILE** is not supported with **InnoDB** tablespaces.

Using **DROP DATAFILE** with [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) drops the data file '***file\_name***' from an NDB tablespace. You cannot drop a data file from a tablespace which is in use by any table; in other words, the data file must be empty (no extents used). See [Section 23.5.10.1, “NDB Cluster Disk Data Objects”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data-objects). In addition, any data file to be dropped must previously have been added to the tablespace with [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) or [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace). **DROP DATAFILE** is not supported with **InnoDB** tablespaces.

**WAIT** is parsed but otherwise ignored. It is intended for future expansion.

The **ENGINE** clause, which specifies the storage engine used by the tablespace, is deprecated; expect it to be removed in a future release. The tablespace storage engine is known by the data dictionary, making the **ENGINE** clause obsolete. If the storage engine is specified, it must match the tablespace storage engine defined in the data dictionary. The only values for ***engine\_name*** compatible with [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tablespaces are **NDB** and **NDBCLUSTER**.

**RENAME TO** operations are implicitly performed in [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) mode, regardless of the [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) setting.

A **RENAME TO** operation cannot be performed while [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) or [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) is in effect for tables that reside in the tablespace.

Exclusive [metadata locks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_metadata_lock) are taken on tables that reside in a general tablespace while the tablespace is renamed, which prevents concurrent DDL. Concurrent DML is supported.

The [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-tablespace) privilege is required to rename an **InnoDB** general tablespace.

The **AUTOEXTEND\_SIZE** option defines the amount by which **InnoDB** extends the size of a tablespace when it becomes full. Introduced in MySQL 8.0.23. The setting must be a multiple of 4MB. The default setting is 0, which causes the tablespace to be extended according to the implicit default behavior. For more information, see [Section 15.6.3.9, “Tablespace AUTOEXTEND\_SIZE Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tablespace-autoextend-size).

The **ENCRYPTION** clause enables or disables page-level data encryption for an **InnoDB** general tablespace or the **mysql** system tablespace. Encryption support for general tablespaces was introduced in MySQL 8.0.13. Encryption support for the **mysql** system tablespace was introduced in MySQL 8.0.16.

A keyring plugin must be installed and configured before encryption can be enabled.

As of MySQL 8.0.16, if the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) variable is enabled, the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege is required to alter a general tablespace with an **ENCRYPTION** clause setting that differs from the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting.

Enabling encryption for a general tablespace fails if any table in the tablespace belongs to a schema defined with [**DEFAULT ENCRYPTION='N'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database). Similarly, disabling encryption fails if any table in the general tablespace belongs to a schema defined with [**DEFAULT ENCRYPTION='Y'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database). The [**DEFAULT ENCRYPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) schema option was introduced in MySQL 8.0.16.

If an [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement executed on a general tablespace does not include an **ENCRYPTION** clause, the tablespace retains its current encryption status, regardless of the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting.

When a general tablespace or the **mysql** system tablespace is encrypted, all tables residing in the tablespace are encrypted. Likewise, a table created in an encrypted tablespace is encrypted.

The **INPLACE** algorithm is used when altering the **ENCRYPTION** attribute of a general tablespace or the **mysql** system tablespace. The **INPLACE** algorithm permits concurrent DML on tables that reside in the tablespace. Concurrent DDL is blocked.

For more information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

The **ENGINE\_ATTRIBUTE** option (available as of MySQL 8.0.21) is used to specify tablespace attributes for primary storage engines. The option is reserved for future use.

Permitted values are a string literal containing a valid **JSON** document or an empty string (''). Invalid **JSON** is rejected.

ALTER TABLESPACE ts1 ENGINE\_ATTRIBUTE='{"***key***":"***value***"}';

**ENGINE\_ATTRIBUTE** values can be repeated without error. In this case, the last specified value is used.

**ENGINE\_ATTRIBUTE** values are not checked by the server, nor are they cleared when the table's storage engine is changed.

It is not permitted to alter an individual element of a JSON attribute value. You can only add or replace an attribute.

### 13.1.11 ALTER VIEW Statement

ALTER

[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]

[DEFINER = ***user***]

[SQL SECURITY { DEFINER | INVOKER }]

VIEW ***view\_name*** [(***column\_list***)]

AS ***select\_statement***

[WITH [CASCADED | LOCAL] CHECK OPTION]

This statement changes the definition of a view, which must exist. The syntax is similar to that for [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view) see [Section 13.1.23, “CREATE VIEW Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view)). This statement requires the [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-view) and [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) privileges for the view, and some privilege for each column referred to in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. [**ALTER VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-view) is permitted only to the definer or users with the [**SET\_USER\_ID**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_set-user-id) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

### 13.1.12 CREATE DATABASE Statement

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] ***db\_name***

[***create\_option***] ...

***create\_option***: [DEFAULT] {

CHARACTER SET [=] ***charset\_name***

| COLLATE [=] ***collation\_name***

| ENCRYPTION [=] {'Y' | 'N'}

}

[**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) creates a database with the given name. To use this statement, you need the [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create) privilege for the database. [**CREATE SCHEMA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) is a synonym for [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database).

An error occurs if the database exists and you did not specify **IF NOT EXISTS**.

[**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) is not permitted within a session that has an active [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement.

Each ***create\_option*** specifies a database characteristic. Database characteristics are stored in the data dictionary.

The **CHARACTER SET** option specifies the default database character set. The **COLLATE** option specifies the default database collation. For information about character set and collation names, see [Chapter 10, *Character Sets, Collations, Unicode*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html).

To see the available character sets and collations, use the the [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) and [**SHOW COLLATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) statements, respectively. See [Section 13.7.7.3, “SHOW CHARACTER SET Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set), and [Section 13.7.7.4, “SHOW COLLATION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation).

The **ENCRYPTION** option, introduced in MySQL 8.0.16, defines the default database encryption, which is inherited by tables created in the database. The permitted values are **'Y'** (encryption enabled) and **'N'** (encryption disabled). If the **ENCRYPTION** option is not specified, the value of the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) system variable defines the default database encryption. If the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) system variable is enabled, the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege is required to specify a default encryption setting that differs from the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting. For more information, see [Defining an Encryption Default for Schemas and General Tablespaces](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-schema-tablespace-encryption-default).

A database in MySQL is implemented as a directory containing files that correspond to tables in the database. Because there are no tables in a database when it is initially created, the [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) statement creates only a directory under the MySQL data directory. Rules for permissible database names are given in [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers). If a database name contains special characters, the name for the database directory contains encoded versions of those characters as described in [Section 9.2.4, “Mapping of Identifiers to File Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifier-mapping).

Creating a database directory by manually creating a directory under the data directory (for example, with **mkdir**) is unsupported in MySQL 8.0.

When you create a database, let the server manage the directory and the files in it. Manipulating database directories and files directly can cause inconsistencies and unexpected results.

MySQL has no limit on the number of databases. The underlying file system may have a limit on the number of directories.

You can also use the [**mysqladmin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) program to create databases. See [Section 4.5.2, “mysqladmin — A MySQL Server Administration Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin).

### 13.1.13 CREATE EVENT Statement

CREATE

[DEFINER = ***user***]

EVENT

[IF NOT EXISTS]

***event\_name***

ON SCHEDULE ***schedule***

[ON COMPLETION [NOT] PRESERVE]

[ENABLE | DISABLE | DISABLE ON SLAVE]

[COMMENT '***string***']

DO ***event\_body***;

***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}

This statement creates and schedules a new event. The event does not run unless the Event Scheduler is enabled. For information about checking Event Scheduler status and enabling it if necessary, see [Section 25.4.2, “Event Scheduler Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#events-configuration).

[**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) requires the [**EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_event) privilege for the schema in which the event is to be created. If the **DEFINER** clause is present, the privileges required depend on the ***user*** value, as discussed in [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security).

The minimum requirements for a valid [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statement are as follows:

The keywords [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) plus an event name, which uniquely identifies the event in a database schema.

An **ON SCHEDULE** clause, which determines when and how often the event executes.

A [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) clause, which contains the SQL statement to be executed by an event.

This is an example of a minimal [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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 MySQL identifier with a maximum length of 64 characters. Event names are not case-sensitive, so you cannot have two events named **myevent** and **MyEvent** in the same schema. In general, the rules governing event names are the same as those for names of stored routines. See [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers).

An event is associated with a schema. If no schema is indicated as part of ***event\_name***, the default (current) schema is assumed. To create an event in a specific schema, qualify the event name with a schema using ***schema\_name*.*event\_name*** syntax.

The **DEFINER** clause specifies the MySQL account to be used when checking access privileges at event execution time. If the **DEFINER** clause is present, the ***user*** value should be a MySQL account specified as **'*user\_name*'@'*host\_name*'**, [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user), or [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user). The permitted ***user*** values depend on the privileges you hold, as discussed in [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security). Also see that section for additional information about event security.

If the **DEFINER** clause is omitted, the default definer is the user who executes the [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statement. This is the same as specifying **DEFINER = CURRENT\_USER** explicitly.

Within an event body, the [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) function returns the account used to check privileges at event execution time, which is the **DEFINER** user. For information about user auditing within events, see [Section 6.2.22, “SQL-Based Account Activity Auditing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-activity-auditing).

**IF NOT EXISTS** has the same meaning for [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) as for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table): If an event named ***event\_name*** already exists in the same schema, no action is taken, and no error results. (However, a warning is generated in such cases.)

The **ON SCHEDULE** clause determines when, how often, and for how long the ***event\_body*** defined for the event repeats. This clause takes one of two forms:

**AT *timestamp*** is used for a one-time event. It specifies that the event executes one time only at the date and time given by ***timestamp***, which must include both the date and time, or must be an expression that resolves to a datetime value. You may use a value of either the [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) or [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) type for this purpose. If the date is in the past, a warning occurs, as shown here:

mysql> **SELECT NOW();**

+---------------------+

| NOW() |

+---------------------+

| 2006-02-10 23:59:01 |

+---------------------+

1 row in set (0.04 sec)

mysql> **CREATE EVENT e\_totals**

-> **ON SCHEDULE AT '2006-02-10 23:59:00'**

-> **DO INSERT INTO test.totals VALUES (NOW());**

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Note

Code: 1588

Message: Event execution time is in the past and ON COMPLETION NOT

PRESERVE is set. The event was dropped immediately after

creation.

[**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statements which are themselves invalid—for whatever reason—fail with an error.

You may use [**CURRENT\_TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-timestamp) to specify the current date and time. In such a case, the event acts as soon as it is created.

To create an event which occurs at some point in the future relative to the current date and time—such as that expressed by the phrase “three weeks from now”—you can use the optional clause **+ INTERVAL *interval***. The ***interval*** portion consists of two parts, a quantity and a unit of time, and follows the syntax rules described in [Temporal Intervals](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#temporal-intervals), except that you cannot use any units keywords that involving microseconds when defining an event. With some interval types, complex time units may be used. For example, “two minutes and ten seconds” can be expressed as **+ INTERVAL '2:10' MINUTE\_SECOND**.

You can also combine intervals. For example, **AT CURRENT\_TIMESTAMP + INTERVAL 3 WEEK + INTERVAL 2 DAY** is equivalent to “three weeks and two days from now”. Each portion of such a clause must begin with **+ INTERVAL**.

To repeat actions at a regular interval, use an **EVERY** clause. The **EVERY** keyword is followed by an ***interval*** as described in the previous discussion of the **AT** keyword. (**+ INTERVAL** is not used with **EVERY**.) For example, **EVERY 6 WEEK** means “every six weeks”.

Although **+ INTERVAL** clauses are not permitted in an **EVERY** clause, you can use the same complex time units permitted in a **+ INTERVAL**.

An **EVERY** clause may contain an optional **STARTS** clause. **STARTS** is followed by a ***timestamp*** value that indicates when the action should begin repeating, and may also use **+ INTERVAL *interval*** to specify an amount of time “from now”. For example, **EVERY 3 MONTH STARTS CURRENT\_TIMESTAMP + INTERVAL 1 WEEK** means “every three months, beginning one week from now”. Similarly, you can express “every two weeks, beginning six hours and fifteen minutes from now” as **EVERY 2 WEEK STARTS CURRENT\_TIMESTAMP + INTERVAL '6:15' HOUR\_MINUTE**. Not specifying **STARTS** is the same as using **STARTS CURRENT\_TIMESTAMP**—that is, the action specified for the event begins repeating immediately upon creation of the event.

An **EVERY** clause may contain an optional **ENDS** clause. The **ENDS** keyword is followed by a ***timestamp*** value that tells MySQL when the event should stop repeating. You may also use **+ INTERVAL *interval*** with **ENDS**; for instance, **EVERY 12 HOUR STARTS CURRENT\_TIMESTAMP + INTERVAL 30 MINUTE ENDS CURRENT\_TIMESTAMP + INTERVAL 4 WEEK** is equivalent to “every twelve hours, beginning thirty minutes from now, and ending four weeks from now”. Not using **ENDS** means that the event continues executing indefinitely.

**ENDS** supports the same syntax for complex time units as **STARTS** does.

You may use **STARTS**, **ENDS**, both, or neither in an **EVERY** clause.

If a repeating event does not terminate within its scheduling interval, the result may be multiple instances of the event executing simultaneously. If this is undesirable, you should institute a mechanism to prevent simultaneous instances. For example, you could use the [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) function, or row or table locking.

The **ON SCHEDULE** clause may use expressions involving built-in MySQL functions and user variables to obtain any of the ***timestamp*** or ***interval*** values which it contains. You may not use stored functions or user-defined functions in such expressions, nor may you use any table references; however, you may use **SELECT FROM DUAL**. This is true for both [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) and [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statements. References to stored functions, user-defined functions, and tables in such cases are specifically not permitted, and fail with an error (see Bug #22830).

Times in the **ON SCHEDULE** clause are interpreted using the current session [**time\_zone**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_time_zone) value. This becomes the event time zone; that is, the time zone that is used for event scheduling and is in effect within the event as it executes. These times are converted to UTC and stored along with the event time zone internally. This enables event execution to proceed as defined regardless of any subsequent changes to the server time zone or daylight saving time effects. For additional information about representation of event times, see [Section 25.4.4, “Event Metadata”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#events-metadata). See also [Section 13.7.7.18, “SHOW EVENTS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events), and [Section 26.3.14, “The INFORMATION\_SCHEMA EVENTS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table).

Normally, once an event has expired, it is immediately dropped. You can override this behavior by specifying **ON COMPLETION PRESERVE**. Using **ON COMPLETION NOT PRESERVE** merely makes the default nonpersistent behavior explicit.

You can create an event but prevent it from being active using the **DISABLE** keyword. Alternatively, you can use **ENABLE** to make explicit the default status, which is active. This is most useful in conjunction with [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) (see [Section 13.1.3, “ALTER EVENT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event)).

A third value may also appear in place of **ENABLE** or **DISABLE**; **DISABLE ON SLAVE** is set for the status of an event on a replica to indicate that the event was created on the replication source server and replicated to the replica, but is not executed on the replica. See [Section 17.5.1.16, “Replication of Invoked Features”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-invoked).

You may supply a comment for an event using a **COMMENT** clause. ***comment*** may be any string of up to 64 characters that you wish to use for describing the event. The comment text, being a string literal, must be surrounded by quotation marks.

The [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) clause specifies an action carried by the event, and consists of an SQL statement. Nearly any valid MySQL statement that can be used in a stored routine can also be used as the action statement for a scheduled event. (See [Section 25.8, “Restrictions on Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-program-restrictions).) For example, the following event **e\_hourly** deletes all rows from the **sessions** table once per hour, where this table is part of the **site\_activity** schema:

CREATE EVENT e\_hourly

ON SCHEDULE

EVERY 1 HOUR

COMMENT 'Clears out sessions table each hour.'

DO

DELETE FROM site\_activity.sessions;

MySQL stores the [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) system variable setting in effect when an event is created or altered, and always executes the event with this setting in force, regardless of the current server SQL mode when the event begins executing.

A [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statement that contains an [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement in its [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) clause appears to succeed; however, when the server attempts to execute the resulting scheduled event, the execution fails with an error.

**Note**

Statements such as [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) that merely return a result set have no effect when used in an event; the output from these is not sent to the MySQL Monitor, nor is it stored anywhere. However, you can use statements such as [**SELECT ... INTO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and [**INSERT INTO ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) that store a result. (See the next example in this section for an instance of the latter.)

The schema to which an event belongs is the default schema for table references in the [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) clause. Any references to tables in other schemas must be qualified with the proper schema name.

As with stored routines, you can use compound-statement syntax in the [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) clause by using the **BEGIN** and **END** keywords, as shown here:

delimiter |

CREATE EVENT e\_daily

ON SCHEDULE

EVERY 1 DAY

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 |

delimiter ;

This example uses the **delimiter** command to change the statement delimiter. See [Section 25.1, “Defining Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-defining).

More complex compound statements, such as those used in stored routines, are possible in an event. This example uses local variables, an error handler, and a flow control construct:

delimiter |

CREATE EVENT e

ON SCHEDULE

EVERY 5 SECOND

DO

BEGIN

DECLARE v INTEGER;

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION BEGIN END;

SET v = 0;

WHILE v < 5 DO

INSERT INTO t1 VALUES (0);

UPDATE t2 SET s1 = s1 + 1;

SET v = v + 1;

END WHILE;

END |

delimiter ;

There is no way to pass parameters directly to or from events; however, it is possible to invoke a stored routine with parameters within an event:

CREATE EVENT e\_call\_myproc

ON SCHEDULE

AT CURRENT\_TIMESTAMP + INTERVAL 1 DAY

DO CALL myproc(5, 27);

If an event's definer has privileges sufficient to set global system variables (see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges)), the event can read and write global variables. As granting such privileges entails a potential for abuse, extreme care must be taken in doing so.

Generally, any statements that are valid in stored routines may be used for action statements executed by events. For more information about statements permissible within stored routines, see [Section 25.2.1, “Stored Routine Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-routines-syntax). You can create an event as part of a stored routine, but an event cannot be created by another event.

### 13.1.14 CREATE FUNCTION Statement

The [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function) statement is used to create stored functions and user-defined functions (UDFs):

For information about creating stored functions, see [Section 13.1.17, “CREATE PROCEDURE and CREATE FUNCTION Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure).

For information about creating user-defined functions, see [Section 13.7.4.1, “CREATE FUNCTION Statement for User-Defined Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf).

### 13.1.15 CREATE INDEX Statement

CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX ***index\_name***

[***index\_type***]

ON ***tbl\_name*** (***key\_part***,...)

[***index\_option***]

[***algorithm\_option*** | ***lock\_option***] ...

***key\_part***: {***col\_name*** [(***length***)] | (***expr***)} [ASC | DESC]

***index\_option***: {

KEY\_BLOCK\_SIZE [=] ***value***

| ***index\_type***

| WITH PARSER ***parser\_name***

| COMMENT '***string***'

| {VISIBLE | INVISIBLE}

| ENGINE\_ATTRIBUTE [=] '***string***'

| SECONDARY\_ENGINE\_ATTRIBUTE [=] '***string***'

}

***index\_type***:

USING {BTREE | HASH}

***algorithm\_option***:

ALGORITHM [=] {DEFAULT | INPLACE | COPY}

***lock\_option***:

LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}

Normally, you create all indexes on a table at the time the table itself is created with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). See [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). This guideline is especially important for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, where the primary key determines the physical layout of rows in the data file. [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) enables you to add indexes to existing tables.

[**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) is mapped to an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement to create indexes. See [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) cannot be used to create a **PRIMARY KEY**; use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) instead. For more information about indexes, see [Section 8.3.1, “How MySQL Uses Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#mysql-indexes).

[**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) supports secondary indexes on virtual columns. For more information, see [Section 13.1.20.9, “Secondary Indexes and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-secondary-indexes).

When the [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) setting is enabled, run the [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) statement for an [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) table after creating an index on that table.

Beginning with MySQL 8.0.17, the ***expr*** for a ***key\_part*** specification can take the form **(CAST *json\_expression* AS *type* ARRAY)** to create a multi-valued index on a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column. See [Multi-Valued Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-multi-valued).

An index specification of the form **(*key\_part1*, *key\_part2*, ...)** creates an index with multiple key parts. Index key values are formed by concatenating the values of the given key parts. For example **(col1, col2, col3)** specifies a multiple-column index with index keys consisting of values from **col1**, **col2**, and **col3**.

A ***key\_part*** specification can end with **ASC** or **DESC** to specify whether index values are stored in ascending or descending order. The default is ascending if no order specifier is given. **ASC** and **DESC** are not permitted for **HASH** indexes. **ASC** and **DESC** are also not supported for multi-valued indexes. As of MySQL 8.0.12, **ASC** and **DESC** are not permitted for **SPATIAL** indexes.

The following sections describe different aspects of the [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statement:

[Column Prefix Key Parts](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-column-prefixes)

[Functional Key Parts](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-functional-key-parts)

[Unique Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-unique)

[Full-Text Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-fulltext)

[Multi-Valued Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-multi-valued)

[Spatial Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-spatial)

[Index Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-options)

[Table Copying and Locking Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-copying)

#### Column Prefix Key Parts

For string columns, indexes can be created that use only the leading part of column values, using ***col\_name*(*length*)** syntax to specify an index prefix length:

Prefixes can be specified for [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) key parts.

Prefixes must be specified for [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) key parts. Additionally, [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns can be indexed only for **InnoDB**, **MyISAM**, and **BLACKHOLE** tables.

Prefix limits are measured in bytes. However, prefix lengths for index specifications in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), and [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statements are interpreted as number of characters for nonbinary string types ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)) and number of bytes for binary string types ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)). Take this into account when specifying a prefix length for a nonbinary string column that uses a multibyte character set.

Prefix support and lengths of prefixes (where supported) are storage engine dependent. For example, a prefix can be up to 767 bytes long for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables that use the [**REDUNDANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redundant_row_format) or [**COMPACT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compact_row_format) row format. The prefix length limit is 3072 bytes for **InnoDB** tables that use the [**DYNAMIC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dynamic_row_format) or [**COMPRESSED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compressed_row_format) row format. For [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables, the prefix length limit is 1000 bytes. The [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine does not support prefixes (see [Section 23.1.7.6, “Unsupported or Missing Features in NDB Cluster”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-limitations-unsupported)).

If a specified index prefix exceeds the maximum column data type size, [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) handles the index as follows:

For a nonunique index, either an error occurs (if strict SQL mode is enabled), or the index length is reduced to lie within the maximum column data type size and a warning is produced (if strict SQL mode is not enabled).

For a unique index, an error occurs regardless of SQL mode because reducing the index length might enable insertion of nonunique entries that do not meet the specified uniqueness requirement.

The statement shown here creates an index using the first 10 characters of the **name** column (assuming that **name** has a nonbinary string type):

CREATE INDEX part\_of\_name ON customer (name(10));

If names in the column usually differ in the first 10 characters, lookups performed using this index should not be much slower than using an index created from the entire **name** column. Also, using column prefixes for indexes can make the index file much smaller, which could save a lot of disk space and might also speed up [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations.

#### Functional Key Parts

A “normal” index indexes column values or prefixes of column values. For example, in the following table, the index entry for a given **t1** row includes the full **col1** value and a prefix of the **col2** value consisting of its first 10 characters:

CREATE TABLE t1 (

col1 VARCHAR(10),

col2 VARCHAR(20),

INDEX (col1, col2(10))

);

MySQL 8.0.13 and higher supports functional key parts that index expression values rather than column or column prefix values. Use of functional key parts enables indexing of values not stored directly in the table. Examples:

CREATE TABLE t1 (col1 INT, col2 INT, INDEX func\_index ((ABS(col1))));

CREATE INDEX idx1 ON t1 ((col1 + col2));

CREATE INDEX idx2 ON t1 ((col1 + col2), (col1 - col2), col1);

ALTER TABLE t1 ADD INDEX ((col1 \* 40) DESC);

An index with multiple key parts can mix nonfunctional and functional key parts.

**ASC** and **DESC** are supported for functional key parts.

Functional key parts must adhere to the following rules. An error occurs if a key part definition contains disallowed constructs.

In index definitions, enclose expressions within parentheses to distinguish them from columns or column prefixes. For example, this is permitted; the expressions are enclosed within parentheses:

INDEX ((col1 + col2), (col3 - col4))

This produces an error; the expressions are not enclosed within parentheses:

INDEX (col1 + col2, col3 - col4)

A functional key part cannot consist solely of a column name. For example, this is not permitted:

INDEX ((col1), (col2))

Instead, write the key parts as nonfunctional key parts, without parentheses:

INDEX (col1, col2)

A functional key part expression cannot refer to column prefixes. For a workaround, see the discussion of [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring) and [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) later in this section.

Functional key parts are not permitted in foreign key specifications.

For [**CREATE TABLE ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like), the destination table preserves functional key parts from the original table.

Functional indexes are implemented as hidden virtual generated columns, which has these implications:

Each functional key part counts against the limit on total number of table columns; see [Section 8.4.7, “Limits on Table Column Count and Row Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#column-count-limit).

Functional key parts inherit all restrictions that apply to generated columns. Examples:

Only functions permitted for generated columns are permitted for functional key parts.

Subqueries, parameters, variables, stored functions, and user-defined functions are not permitted.

For more information about applicable restrictions, see [Section 13.1.20.8, “CREATE TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-generated-columns), and [Section 13.1.9.2, “ALTER TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-generated-columns).

The virtual generated column itself requires no storage. The index itself takes up storage space as any other index.

**UNIQUE** is supported for indexes that include functional key parts. However, primary keys cannot include functional key parts. A primary key requires the generated column to be stored, but functional key parts are implemented as virtual generated columns, not stored generated columns.

**SPATIAL** and **FULLTEXT** indexes cannot have functional key parts.

If a table contains no primary key, **InnoDB** automatically promotes the first **UNIQUE NOT NULL** index to the primary key. This is not supported for **UNIQUE NOT NULL** indexes that have functional key parts.

Nonfunctional indexes raise a warning if there are duplicate indexes. Indexes that contain functional key parts do not have this feature.

To remove a column that is referenced by a functional key part, the index must be removed first. Otherwise, an error occurs.

Although nonfunctional key parts support a prefix length specification, this is not possible for functional key parts. The solution is to use [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring) (or [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast), as described later in this section). For a functional key part containing the [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring) function to be used in a query, the **WHERE** clause must contain [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring) with the same arguments. In the following example, only the second [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) is able to use the index because that is the only query in which the arguments to [**SUBSTRING()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_substring) match the index specification:

CREATE TABLE tbl (

col1 LONGTEXT,

INDEX idx1 ((SUBSTRING(col1, 1, 10)))

);

SELECT \* FROM tbl WHERE SUBSTRING(col1, 1, 9) = '123456789';

SELECT \* FROM tbl WHERE SUBSTRING(col1, 1, 10) = '1234567890';

Functional key parts enable indexing of values that cannot be indexed otherwise, such as [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) values. However, this must be done correctly to achieve the desired effect. For example, this syntax does not work:

CREATE TABLE employees (

data JSON,

INDEX ((data->>'$.name'))

);

The syntax fails because:

The [**->>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-inline-path) operator translates into [**JSON\_UNQUOTE(JSON\_EXTRACT(...))**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-unquote).

[**JSON\_UNQUOTE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-unquote) returns a value with a data type of [**LONGTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), and the hidden generated column thus is assigned the same data type.

MySQL cannot index [**LONGTEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns specified without a prefix length on the key part, and prefix lengths are not permitted in functional key parts.

To index the **JSON** column, you could try using the [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) function as follows:

CREATE TABLE employees (

data JSON,

INDEX ((CAST(data->>'$.name' AS CHAR(30))))

);

The hidden generated column is assigned the [**VARCHAR(30)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) data type, which can be indexed. But this approach produces a new issue when trying to use the index:

[**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) returns a string with the collation **utf8mb4\_0900\_ai\_ci** (the server default collation).

[**JSON\_UNQUOTE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-unquote) returns a string with the collation **utf8mb4\_bin** (hard coded).

As a result, there is a collation mismatch between the indexed expression in the preceding table definition and the **WHERE** clause expression in the following query:

SELECT \* FROM employees WHERE data->>'$.name' = 'James';

The index is not used because the expressions in the query and the index differ. To support this kind of scenario for functional key parts, the optimizer automatically strips [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) when looking for an index to use, but only if the collation of the indexed expression matches that of the query expression. For an index with a functional key part to be used, either of the following two solutions work (although they differ somewhat in effect):

Solution 1. Assign the indexed expression the same collation as [**JSON\_UNQUOTE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-unquote):

CREATE TABLE employees (

data JSON,

INDEX idx ((CAST(data->>"$.name" AS CHAR(30)) COLLATE utf8mb4\_bin))

);

INSERT INTO employees VALUES

('{ "name": "james", "salary": 9000 }'),

('{ "name": "James", "salary": 10000 }'),

('{ "name": "Mary", "salary": 12000 }'),

('{ "name": "Peter", "salary": 8000 }');

SELECT \* FROM employees WHERE data->>'$.name' = 'James';

The **->>** operator is the same as **JSON\_UNQUOTE(JSON\_EXTRACT(...))**, and **JSON\_UNQUOTE()** returns a string with collation **utf8mb4\_bin**. The comparison is thus case-sensitive, and only one row matches:

+------------------------------------+

| data |

+------------------------------------+

| {"name": "James", "salary": 10000} |

+------------------------------------+

Solution 2. Specify the full expression in the query:

CREATE TABLE employees (

data JSON,

INDEX idx ((CAST(data->>"$.name" AS CHAR(30))))

);

INSERT INTO employees VALUES

('{ "name": "james", "salary": 9000 }'),

('{ "name": "James", "salary": 10000 }'),

('{ "name": "Mary", "salary": 12000 }'),

('{ "name": "Peter", "salary": 8000 }');

SELECT \* FROM employees WHERE CAST(data->>'$.name' AS CHAR(30)) = 'James';

**CAST()** returns a string with collation **utf8mb4\_0900\_ai\_ci**, so the comparison case-insensitive and two rows match:

+------------------------------------+

| data |

+------------------------------------+

| {"name": "james", "salary": 9000} |

| {"name": "James", "salary": 10000} |

+------------------------------------+

Be aware that although the optimizer supports automatically stripping [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) with indexed generated columns, the following approach does not work because it produces a different result with and without an index (Bug#27337092):

mysql> **CREATE TABLE employees (**

**data JSON,**

**generated\_col VARCHAR(30) AS (CAST(data->>'$.name' AS CHAR(30)))**

**);**

Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> **INSERT INTO employees (data)**

**VALUES ('{"name": "james"}'), ('{"name": "James"}');**

Query OK, 2 rows affected, 1 warning (0.01 sec)

Records: 2 Duplicates: 0 Warnings: 1

mysql> **SELECT \* FROM employees WHERE data->>'$.name' = 'James';**

+-------------------+---------------+

| data | generated\_col |

+-------------------+---------------+

| {"name": "James"} | James |

+-------------------+---------------+

1 row in set (0.00 sec)

mysql> **ALTER TABLE employees ADD INDEX idx (generated\_col);**

Query OK, 0 rows affected, 1 warning (0.03 sec)

Records: 0 Duplicates: 0 Warnings: 1

mysql> **SELECT \* FROM employees WHERE data->>'$.name' = 'James';**

+-------------------+---------------+

| data | generated\_col |

+-------------------+---------------+

| {"name": "james"} | james |

| {"name": "James"} | James |

+-------------------+---------------+

2 rows in set (0.01 sec)

#### Unique Indexes

A **UNIQUE** index creates a constraint such that all values in the index must be distinct. An error occurs if you try to add a new row with a key value that matches an existing row. If you specify a prefix value for a column in a **UNIQUE** index, the column values must be unique within the prefix length. A **UNIQUE** index permits multiple **NULL** values for columns that can contain **NULL**.

If a table has a **PRIMARY KEY** or **UNIQUE NOT NULL** index that consists of a single column that has an integer type, you can use **\_rowid** to refer to the indexed column in [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, as follows:

**\_rowid** refers to the **PRIMARY KEY** column if there is a **PRIMARY KEY** consisting of a single integer column. If there is a **PRIMARY KEY** but it does not consist of a single integer column, **\_rowid** cannot be used.

Otherwise, **\_rowid** refers to the column in the first **UNIQUE NOT NULL** index if that index consists of a single integer column. If the first **UNIQUE NOT NULL** index does not consist of a single integer column, **\_rowid** cannot be used.

#### Full-Text Indexes

**FULLTEXT** indexes are supported only for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables and can include only [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns. Indexing always happens over the entire column; column prefix indexing is not supported and any prefix length is ignored if specified. See [Section 12.10, “Full-Text Search Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search), for details of operation.

#### Multi-Valued Indexes

As of MySQL 8.0.17, **InnoDB** supports multi-valued indexes. A multi-valued index is a secondary index defined on a column that stores an array of values. A “normal” index has one index record for each data record (1:1). A multi-valued index can have multiple index records for a single data record (N:1). Multi-valued indexes are intended for indexing **JSON** arrays. For example, a multi-valued index defined on the array of zip codes in the following JSON document creates an index record for each zip code, with each index record referencing the same data record.

{

"user":"Bob",

"user\_id":31,

"zipcode":[94477,94536]

}

##### Creating multi-valued Indexes

You can create a multi-valued index in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), or [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statement. This requires using [**CAST(... AS ... ARRAY)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) in the index definition, which casts same-typed scalar values in a **JSON** array to an SQL data type array. A virtual column is then generated transparently with the values in the SQL data type array; finally, a functional index (also referred to as a virtual index) is created on the virtual column. It is the functional index defined on the virtual column of values from the SQL data type array that forms the multi-valued index.

The examples in the following list show the three different ways in which a multi-valued index **zips** can be created on an array **$.zipcode** on a **JSON** column **custinfo** in a table named **customers**. In each case, the JSON array is cast to an SQL data type array of **UNSIGNED** integer values.

**CREATE TABLE** only:

CREATE TABLE customers (

id BIGINT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

modified DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

custinfo JSON,

INDEX zips( (CAST(custinfo->'$.zip' AS UNSIGNED ARRAY)) )

);

**CREATE TABLE** plus **ALTER TABLE**:

CREATE TABLE customers (

id BIGINT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

modified DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

custinfo JSON

);

ALTER TABLE customers ADD INDEX zips( (CAST(custinfo->'$.zip' AS UNSIGNED ARRAY)) );

**CREATE TABLE** plus **CREATE INDEX**:

CREATE TABLE customers (

id BIGINT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

modified DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

custinfo JSON

);

CREATE INDEX zips ON customers ( (CAST(custinfo->'$.zip' AS UNSIGNED ARRAY)) );

A multi-valued index can also be defined as part of a composite index. This example shows a composite index that includes two single-valued parts (for the **id** and **modified** columns), and one multi-valued part (for the **custinfo** column):

CREATE TABLE customers (

id BIGINT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

modified DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

custinfo JSON

);

ALTER TABLE customers ADD INDEX comp(id, modified,

(CAST(custinfo->'$.zipcode' AS UNSIGNED ARRAY)) );

Only one multi-valued key part can be used in a composite index. The multi-valued key part may be used in any order relative to the other parts of the key. In other words, the **ALTER TABLE** statement just shown could have used **comp(id, (CAST(custinfo->'$.zipcode' AS UNSIGNED ARRAY), modified))** (or any other ordering) and still have been valid.

##### Using multi-valued Indexes

The optimizer uses a multi-valued index to fetch records when the following functions are specified in a **WHERE** clause:

[**MEMBER OF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_member-of)

[**JSON\_CONTAINS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-contains)

[**JSON\_OVERLAPS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-overlaps)

We can demonstrate this by creating and populating the **customers** table using the following **CREATE TABLE** and **INSERT** statements:

mysql> **CREATE TABLE customers (**

-> **id BIGINT NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

-> **modified DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,**

-> **custinfo JSON**

-> **);**

Query OK, 0 rows affected (0.51 sec)

mysql> **INSERT INTO customers VALUES**

-> **(NULL, NOW(), '{"user":"Jack","user\_id":37,"zipcode":[94582,94536]}'),**

-> **(NULL, NOW(), '{"user":"Jill","user\_id":22,"zipcode":[94568,94507,94582]}'),**

-> **(NULL, NOW(), '{"user":"Bob","user\_id":31,"zipcode":[94477,94507]}'),**

-> **(NULL, NOW(), '{"user":"Mary","user\_id":72,"zipcode":[94536]}'),**

-> **(NULL, NOW(), '{"user":"Ted","user\_id":56,"zipcode":[94507,94582]}');**

Query OK, 5 rows affected (0.07 sec)

Records: 5 Duplicates: 0 Warnings: 0

First we execute three queries on the **customers** table, one each using **MEMBER OF()**, **JSON\_CONTAINS()**, and **JSON\_OVERLAPS()**, with the result from each query shown here:

mysql> **SELECT \* FROM customers**

-> **WHERE 94507 MEMBER OF(custinfo->'$.zipcode');**

+----+---------------------+-------------------------------------------------------------------+

| id | modified | custinfo |

+----+---------------------+-------------------------------------------------------------------+

| 2 | 2019-06-29 22:23:12 | {"user": "Jill", "user\_id": 22, "zipcode": [94568, 94507, 94582]} |

| 3 | 2019-06-29 22:23:12 | {"user": "Bob", "user\_id": 31, "zipcode": [94477, 94507]} |

| 5 | 2019-06-29 22:23:12 | {"user": "Ted", "user\_id": 56, "zipcode": [94507, 94582]} |

+----+---------------------+-------------------------------------------------------------------+

3 rows in set (0.00 sec)

mysql> **SELECT \* FROM customers**

-> **WHERE JSON\_CONTAINS(custinfo->'$.zipcode', CAST('[94507,94582]' AS JSON));**

+----+---------------------+-------------------------------------------------------------------+

| id | modified | custinfo |

+----+---------------------+-------------------------------------------------------------------+

| 2 | 2019-06-29 22:23:12 | {"user": "Jill", "user\_id": 22, "zipcode": [94568, 94507, 94582]} |

| 5 | 2019-06-29 22:23:12 | {"user": "Ted", "user\_id": 56, "zipcode": [94507, 94582]} |

+----+---------------------+-------------------------------------------------------------------+

2 rows in set (0.00 sec)

mysql> **SELECT \* FROM customers**

-> **WHERE JSON\_OVERLAPS(custinfo->'$.zipcode', CAST('[94507,94582]' AS JSON));**

+----+---------------------+-------------------------------------------------------------------+

| id | modified | custinfo |

+----+---------------------+-------------------------------------------------------------------+

| 1 | 2019-06-29 22:23:12 | {"user": "Jack", "user\_id": 37, "zipcode": [94582, 94536]} |

| 2 | 2019-06-29 22:23:12 | {"user": "Jill", "user\_id": 22, "zipcode": [94568, 94507, 94582]} |

| 3 | 2019-06-29 22:23:12 | {"user": "Bob", "user\_id": 31, "zipcode": [94477, 94507]} |

| 5 | 2019-06-29 22:23:12 | {"user": "Ted", "user\_id": 56, "zipcode": [94507, 94582]} |

+----+---------------------+-------------------------------------------------------------------+

4 rows in set (0.00 sec)

Next, we run [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) on each of the previous three queries:

mysql> **EXPLAIN SELECT \* FROM customers**

-> **WHERE 94507 MEMBER OF(custinfo->'$.zipcode');**

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

| 1 | SIMPLE | customers | NULL | ALL | NULL | NULL | NULL | NULL | 5 | 100.00 | Using where |

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

1 row in set, 1 warning (0.00 sec)

mysql> **EXPLAIN SELECT \* FROM customers**

-> **WHERE JSON\_CONTAINS(custinfo->'$.zipcode', CAST('[94507,94582]' AS JSON));**

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

| 1 | SIMPLE | customers | NULL | ALL | NULL | NULL | NULL | NULL | 5 | 100.00 | Using where |

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

1 row in set, 1 warning (0.00 sec)

mysql> **EXPLAIN SELECT \* FROM customers**

-> **WHERE JSON\_OVERLAPS(custinfo->'$.zipcode', CAST('[94507,94582]' AS JSON));**

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

| 1 | SIMPLE | customers | NULL | ALL | NULL | NULL | NULL | NULL | 5 | 100.00 | Using where |

+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+

1 row in set, 1 warning (0.01 sec)

None of the three queries just shown are able to use any keys. To solve this problem, we can add a multi-valued index on the **zipcode** array in the **JSON** column (**custinfo**), like this:

mysql> **ALTER TABLE customers**

-> **ADD INDEX zips( (CAST(custinfo->'$.zipcode' AS UNSIGNED ARRAY)) );**

Query OK, 0 rows affected (0.47 sec)

Records: 0 Duplicates: 0 Warnings: 0

When we run the previous **EXPLAIN** statements again, we can now observe that the queries can (and do) use the index **zips** that was just created:

mysql> **EXPLAIN SELECT \* FROM customers**

-> **WHERE 94507 MEMBER OF(custinfo->'$.zipcode');**

+----+-------------+-----------+------------+------+---------------+------+---------+-------+------+----------+-------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-----------+------------+------+---------------+------+---------+-------+------+----------+-------------+

| 1 | SIMPLE | customers | NULL | ref | zips | zips | 9 | const | 1 | 100.00 | Using where |

+----+-------------+-----------+------------+------+---------------+------+---------+-------+------+----------+-------------+

1 row in set, 1 warning (0.00 sec)

mysql> **EXPLAIN SELECT \* FROM customers**

-> **WHERE JSON\_CONTAINS(custinfo->'$.zipcode', CAST('[94507,94582]' AS JSON));**

+----+-------------+-----------+------------+-------+---------------+------+---------+------+------+----------+-------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-----------+------------+-------+---------------+------+---------+------+------+----------+-------------+

| 1 | SIMPLE | customers | NULL | range | zips | zips | 9 | NULL | 6 | 100.00 | Using where |

+----+-------------+-----------+------------+-------+---------------+------+---------+------+------+----------+-------------+

1 row in set, 1 warning (0.00 sec)

mysql> **EXPLAIN SELECT \* FROM customers**

-> **WHERE JSON\_OVERLAPS(custinfo->'$.zipcode', CAST('[94507,94582]' AS JSON));**

+----+-------------+-----------+------------+-------+---------------+------+---------+------+------+----------+-------------+

| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-----------+------------+-------+---------------+------+---------+------+------+----------+-------------+

| 1 | SIMPLE | customers | NULL | range | zips | zips | 9 | NULL | 6 | 100.00 | Using where |

+----+-------------+-----------+------------+-------+---------------+------+---------+------+------+----------+-------------+

1 row in set, 1 warning (0.01 sec)

A multi-valued index can be defined as a unique key. If defined as a unique key, attempting to insert a value already present in the multi-valued index returns a duplicate key error. If duplicate values are already present, attempting to add a unique multi-valued index fails, as shown here:

mysql> **ALTER TABLE customers DROP INDEX zips;**

Query OK, 0 rows affected (0.55 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> **ALTER TABLE customers**

-> **ADD UNIQUE INDEX zips((CAST(custinfo->'$.zipcode' AS UNSIGNED ARRAY)));**

ERROR 1062 (23000): Duplicate entry '[94507, ' for key 'customers.zips'

mysql> **ALTER TABLE customers**

-> **ADD INDEX zips((CAST(custinfo->'$.zipcode' AS UNSIGNED ARRAY)));**

Query OK, 0 rows affected (0.36 sec)

Records: 0 Duplicates: 0 Warnings: 0

##### Characteristics of Multi-Valued Indexes

Multi-valued indexes have the additional characteristics listed here:

DML operations that affect multi-valued indexes are handled in the same way as DML operations that affect a normal index, with the only difference being that there may be more than one insert or update for a single clustered index record.

Nullability and multi-valued indexes:

If multi-valued key part has an empty array, no entries are added to the index, and the data record is not accessible by an index scan.

If multi-valued key part generation returns a **NULL** value, a single entry containing **NULL** is added to the multi-valued index. If the key part is defined as **NOT NULL**, an error is reported.

If the typed array column is set to **NULL**, the storage engine stores single record containing **NULL** that points to the data record.

**JSON** null values are not permitted in indexed arrays. If any returned value is **NULL**, it is treated as a JSON null and an Invalid JSON value error is reported.

Because multi-valued indexes are virtual indexes on virtual columns, they must adhere to the same rules as secondary indexes on virtual generated columns.

Index records are not added for empty arrays.

##### Limitations and Restrictions on Multi-valued Indexes

Multi-valued indexes are subject to the limitations and restrictions listed here:

Only one multi-valued key part is permitted per multi-valued index. However, the [**CAST(... AS ... ARRAY)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) expression can refer to multiple arrays within a **JSON** document, as shown here:

CAST(data->'$.arr[\*][\*]' AS UNSIGNED ARRAY)

In this case, all values matching the JSON expression are stored in the index as a single flat array.

An index with a multi-valued key part does not support ordering and therefore cannot be used as a primary key. For the same reason, a multi-valued index cannot be defined using the **ASC** or **DESC** keyword.

A multi-valued index cannot be a covering index.

The maximum number of values per record for a multi-valued index is determined by the amount of data than can be stored on a single undo log page, which is 65221 bytes (64K minus 315 bytes for overhead), which means that the maximum total length of key values is also 65221 bytes. The maximum number of keys depends on various factors, which prevents defining a specific limit. Tests have shown a multi-valued index to permit as many as 1604 integer keys per record, for example. When the limit is reached, an error similar to the following is reported: ERROR 3905 (HY000): Exceeded max number of values per record for multi-valued index 'idx' by 1 value(s).

The only type of expression that is permitted in a multi-valued key part is a **JSON** expression. The expression need not reference an existing element in a JSON document inserted into the indexed column, but must itself be syntactically valid.

Because index records for the same clustered index record are dispersed throughout a multi-valued index, a multi-valued index does not support range scans or index-only scans.

Multi-valued indexes are not permitted in foreign key specifications.

Index prefixes cannot be defined for multi-valued indexes.

Multi-valued indexes cannot be defined on data cast as [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) (see the description of the [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) function).

Online creation of a multi-value index is not supported, which means the operation uses **ALGORITHM=COPY**. See [Performance and Space Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-performance).

Character sets and collations other than the following two combinations of character set and collation are not supported for multi-valued indexes:

The **binary** character set with the default **binary** collation

The **utf8mb4** character set with the default **utf8mb4\_0900\_as\_cs** collation.

As with other indexes on columns of **InnoDB** tables, a multi-valued index cannot be created with **USING HASH**; attempting to do so results in a warning: This storage engine does not support the HASH index algorithm, storage engine default was used instead. (**USING BTREE** is supported as usual.)

#### Spatial Indexes

The [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine), [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html), and [**ARCHIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#archive-storage-engine) storage engines support spatial columns such as [**POINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-type-overview) and [**GEOMETRY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-type-overview). ([Section 11.4, “Spatial Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-types), describes the spatial data types.) However, support for spatial column indexing varies among engines. Spatial and nonspatial indexes on spatial columns are available according to the following rules.

Spatial indexes on spatial columns have these characteristics:

Available only for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables. Specifying **SPATIAL INDEX** for other storage engines results in an error.

As of MySQL 8.0.12, an index on a spatial column must be a **SPATIAL** index. The **SPATIAL** keyword is thus optional but implicit for creating an index on a spatial column.

Available for single spatial columns only. A spatial index cannot be created over multiple spatial columns.

Indexed columns must be **NOT NULL**.

Column prefix lengths are prohibited. The full width of each column is indexed.

Not permitted for a primary key or unique index.

Nonspatial indexes on spatial columns (created with **INDEX**, **UNIQUE**, or **PRIMARY KEY**) have these characteristics:

Permitted for any storage engine that supports spatial columns except [**ARCHIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#archive-storage-engine).

Columns can be **NULL** unless the index is a primary key.

The index type for a non-**SPATIAL** index depends on the storage engine. Currently, B-tree is used.

Permitted for a column that can have **NULL** values only for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine), and [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) tables.

#### Index Options

Following the key part list, index options can be given. An ***index\_option*** value can be any of the following:

**KEY\_BLOCK\_SIZE [=] *value***

For [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables, **KEY\_BLOCK\_SIZE** optionally specifies the size in bytes to use for index key blocks. The value is treated as a hint; a different size could be used if necessary. A **KEY\_BLOCK\_SIZE** value specified for an individual index definition overrides a table-level **KEY\_BLOCK\_SIZE** value.

**KEY\_BLOCK\_SIZE** is not supported at the index level for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables. See [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table).

***index\_type***

Some storage engines permit you to specify an index type when creating an index. For example:

CREATE TABLE lookup (id INT) ENGINE = MEMORY;

CREATE INDEX id\_index ON lookup (id) USING BTREE;

[Table 13.1, “Index Types Per Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-storage-engine-index-types) shows the permissible index type values supported by different storage engines. Where multiple index types are listed, the first one is the default when no index type specifier is given. Storage engines not listed in the table do not support an ***index\_type*** clause in index definitions.

**Table 13.1 Index Types Per Storage Engine**

| **Storage Engine** | **Permissible Index Types** |
| --- | --- |
| [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) | **BTREE** |
| [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) | **BTREE** |
| [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine)/**HEAP** | **HASH**, **BTREE** |
| [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) | **HASH**, **BTREE** (see note in text) |

The ***index\_type*** clause cannot be used for **FULLTEXT INDEX** or (prior to MySQL 8.0.12) **SPATIAL INDEX** specifications. Full-text index implementation is storage engine dependent. Spatial indexes are implemented as R-tree indexes.

If you specify an index type that is not valid for a given storage engine, but another index type is available that the engine can use without affecting query results, the engine uses the available type. The parser recognizes **RTREE** as a type name. As of MySQL 8.0.12, this is permitted only for **SPATIAL** indexes. Prior to 8.0.12, **RTREE** cannot be specified for any storage engine.

**BTREE** indexes are implemented by the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine as T-tree indexes.

**Note**

For indexes on [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) table columns, the **USING** option can be specified only for a unique index or primary key. **USING HASH** prevents the creation of an ordered index; otherwise, creating a unique index or primary key on an [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) table automatically results in the creation of both an ordered index and a hash index, each of which indexes the same set of columns.

For unique indexes that include one or more **NULL** columns of an [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) table, the hash index can be used only to look up literal values, which means that **IS [NOT] NULL** conditions require a full scan of the table. One workaround is to make sure that a unique index using one or more **NULL** columns on such a table is always created in such a way that it includes the ordered index; that is, avoid employing **USING HASH** when creating the index.

If you specify an index type that is not valid for a given storage engine, but another index type is available that the engine can use without affecting query results, the engine uses the available type. The parser recognizes **RTREE** as a type name, but currently this cannot be specified for any storage engine.

**Note**

Use of the ***index\_type*** option before the **ON *tbl\_name*** clause is deprecated; expect support for use of the option in this position to be removed in a future MySQL release. If an ***index\_type*** option is given in both the earlier and later positions, the final option applies.

**TYPE *type\_name*** is recognized as a synonym for **USING *type\_name***. However, **USING** is the preferred form.

The following tables show index characteristics for the storage engines that support the ***index\_type*** option.

**Table 13.2 InnoDB Storage Engine Index Characteristics**

| **Index Class** | **Index Type** | **Stores NULL VALUES** | **Permits Multiple NULL Values** | **IS NULL Scan Type** | **IS NOT NULL Scan Type** |
| --- | --- | --- | --- | --- | --- |
| **Primary key** | **BTREE** | No | No | N/A | N/A |
| **Unique** | **BTREE** | Yes | Yes | Index | Index |
| **Key** | **BTREE** | Yes | Yes | Index | Index |
| **FULLTEXT** | N/A | Yes | Yes | Table | Table |
| **SPATIAL** | N/A | No | No | N/A | N/A |

**Table 13.3 MyISAM Storage Engine Index Characteristics**

| **Index Class** | **Index Type** | **Stores NULL VALUES** | **Permits Multiple NULL Values** | **IS NULL Scan Type** | **IS NOT NULL Scan Type** |
| --- | --- | --- | --- | --- | --- |
| **Primary key** | **BTREE** | No | No | N/A | N/A |
| **Unique** | **BTREE** | Yes | Yes | Index | Index |
| **Key** | **BTREE** | Yes | Yes | Index | Index |
| **FULLTEXT** | N/A | Yes | Yes | Table | Table |
| **SPATIAL** | N/A | No | No | N/A | N/A |

**Table 13.4 MEMORY Storage Engine Index Characteristics**

| **Index Class** | **Index Type** | **Stores NULL VALUES** | **Permits Multiple NULL Values** | **IS NULL Scan Type** | **IS NOT NULL Scan Type** |
| --- | --- | --- | --- | --- | --- |
| **Primary key** | **BTREE** | No | No | N/A | N/A |
| **Unique** | **BTREE** | Yes | Yes | Index | Index |
| **Key** | **BTREE** | Yes | Yes | Index | Index |
| **Primary key** | **HASH** | No | No | N/A | N/A |
| **Unique** | **HASH** | Yes | Yes | Index | Index |
| **Key** | **HASH** | Yes | Yes | Index | Index |

**Table 13.5 NDB Storage Engine Index Characteristics**

| **Index Class** | **Index Type** | **Stores NULL VALUES** | **Permits Multiple NULL Values** | **IS NULL Scan Type** | **IS NOT NULL Scan Type** |
| --- | --- | --- | --- | --- | --- |
| **Primary key** | **BTREE** | No | No | Index | Index |
| **Unique** | **BTREE** | Yes | Yes | Index | Index |
| **Key** | **BTREE** | Yes | Yes | Index | Index |
| **Primary key** | **HASH** | No | No | Table (see note 1) | Table (see note 1) |
| **Unique** | **HASH** | Yes | Yes | Table (see note 1) | Table (see note 1) |
| **Key** | **HASH** | Yes | Yes | Table (see note 1) | Table (see note 1) |

Table note:

1. **USING HASH** prevents creation of an implicit ordered index.

**WITH PARSER *parser\_name***

This option can be used only with **FULLTEXT** indexes. It associates a parser plugin with the index if full-text indexing and searching operations need special handling. [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) support full-text parser plugins. If you have a [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) table with an associated full-text parser plugin, you can convert the table to **InnoDB** using **ALTER TABLE**. See [Full-Text Parser Plugins](https://dev.mysql.com/doc/extending-mysql/8.0/en/plugin-types.html#full-text-plugin-type) and [Writing Full-Text Parser Plugins](https://dev.mysql.com/doc/extending-mysql/8.0/en/writing-full-text-plugins.html) for more information.

**COMMENT '*string*'**

Index definitions can include an optional comment of up to 1024 characters.

The [**MERGE\_THRESHOLD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#index-page-merge-threshold) for index pages can be configured for individual indexes using the ***index\_option*** **COMMENT** clause of the [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statement. For example:

CREATE TABLE t1 (id INT);

CREATE INDEX id\_index ON t1 (id) COMMENT 'MERGE\_THRESHOLD=40';

If the page-full percentage for an index page falls below the **MERGE\_THRESHOLD** value when a row is deleted or when a row is shortened by an update operation, [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) attempts to merge the index page with a neighboring index page. The default **MERGE\_THRESHOLD** value is 50, which is the previously hardcoded value.

**MERGE\_THRESHOLD** can also be defined at the index level and table level using [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements. For more information, see [Section 15.8.11, “Configuring the Merge Threshold for Index Pages”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#index-page-merge-threshold).

**VISIBLE**, **INVISIBLE**

Specify index visibility. Indexes are visible by default. An invisible index is not used by the optimizer. Specification of index visibility applies to indexes other than primary keys (either explicit or implicit). For more information, see [Section 8.3.12, “Invisible Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#invisible-indexes).

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** options (available as of MySQL 8.0.21) are used to specify index attributes for primary and secondary storage engines. The options are reserved for future use.

Permitted values are a string literal containing a valid **JSON** document or an empty string (''). Invalid **JSON** is rejected.

CREATE INDEX i1 ON t1 (c1) ENGINE\_ATTRIBUTE='{"***key***":"***value***"}';

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values can be repeated without error. In this case, the last specified value is used.

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values are not checked by the server, nor are they cleared when the table's storage engine is changed.

#### Table Copying and Locking Options

**ALGORITHM** and **LOCK** clauses may be given to influence the table copying method and level of concurrency for reading and writing the table while its indexes are being modified. They have the same meaning as for the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. For more information, see [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table)

NDB Cluster supports online operations using the same **ALGORITHM=INPLACE** syntax used with the standard MySQL Server. See [Section 23.5.11, “Online Operations with ALTER TABLE in NDB Cluster”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-online-operations), for more information.

### 13.1.16 CREATE LOGFILE GROUP Statement

CREATE LOGFILE GROUP ***logfile\_group***

ADD UNDOFILE '***undo\_file***'

[INITIAL\_SIZE [=] ***initial\_size***]

[UNDO\_BUFFER\_SIZE [=] ***undo\_buffer\_size***]

[REDO\_BUFFER\_SIZE [=] ***redo\_buffer\_size***]

[NODEGROUP [=] ***nodegroup\_id***]

[WAIT]

[COMMENT [=] '***string***']

ENGINE [=] ***engine\_name***

This statement creates a new log file group named ***logfile\_group*** having a single **UNDO** file named '***undo\_file***'. A [**CREATE LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group) statement has one and only one **ADD UNDOFILE** clause. For rules covering the naming of log file groups, see [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers).

**Note**

All NDB Cluster Disk Data objects share the same namespace. This means that each Disk Data object must be uniquely named (and not merely each Disk Data object of a given type). For example, you cannot have a tablespace and a log file group with the same name, or a tablespace and a data file with the same name.

There can be only one log file group per NDB Cluster instance at any given time.

The optional **INITIAL\_SIZE** parameter sets the **UNDO** file's initial size; if not specified, it defaults to **128M** (128 megabytes). The optional **UNDO\_BUFFER\_SIZE** parameter sets the size used by the **UNDO** buffer for the log file group; The default value for **UNDO\_BUFFER\_SIZE** is **8M** (eight megabytes); this value cannot exceed the amount of system memory available. Both of these parameters are specified in bytes. You may optionally follow either or both of these with a one-letter abbreviation for an order of magnitude, similar to those used in my.cnf. Generally, this is one of the letters **M** (for megabytes) or **G** (for gigabytes).

Memory used for **UNDO\_BUFFER\_SIZE** comes from the global pool whose size is determined by the value of the [**SharedGlobalMemory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-sharedglobalmemory) data node configuration parameter. This includes any default value implied for this option by the setting of the [**InitialLogFileGroup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-initiallogfilegroup) data node configuration parameter.

The maximum permitted for **UNDO\_BUFFER\_SIZE** is 629145600 (600 MB).

On 32-bit systems, the maximum supported value for **INITIAL\_SIZE** is 4294967296 (4 GB). (Bug #29186)

The minimum allowed value for **INITIAL\_SIZE** is 1048576 (1 MB).

The **ENGINE** option determines the storage engine to be used by this log file group, with ***engine\_name*** being the name of the storage engine. In MySQL 8.0, this must be [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) (or [**NDBCLUSTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html)). If **ENGINE** is not set, MySQL tries to use the engine specified by the [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) server system variable (formerly [**storage\_engine**](https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_storage_engine)). In any case, if the engine is not specified as [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) or [**NDBCLUSTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html), the **CREATE LOGFILE GROUP** statement appears to succeed but actually fails to create the log file group, as shown here:

mysql> **CREATE LOGFILE GROUP lg1**

-> **ADD UNDOFILE 'undo.dat' INITIAL\_SIZE = 10M;**

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS;**

+-------+------+------------------------------------------------------------------------------------------------+

| Level | Code | Message |

+-------+------+------------------------------------------------------------------------------------------------+

| Error | 1478 | Table storage engine 'InnoDB' does not support the create option 'TABLESPACE or LOGFILE GROUP' |

+-------+------+------------------------------------------------------------------------------------------------+

1 row in set (0.00 sec)

mysql> **DROP LOGFILE GROUP lg1 ENGINE = NDB;**

ERROR 1529 (HY000): Failed to drop LOGFILE GROUP

mysql> **CREATE LOGFILE GROUP lg1**

-> **ADD UNDOFILE 'undo.dat' INITIAL\_SIZE = 10M**

-> **ENGINE = NDB;**

Query OK, 0 rows affected (2.97 sec)

The fact that the **CREATE LOGFILE GROUP** statement does not actually return an error when a non-**NDB** storage engine is named, but rather appears to succeed, is a known issue which we hope to address in a future release of NDB Cluster.

***REDO\_BUFFER\_SIZE***, **NODEGROUP**, **WAIT**, and **COMMENT** are parsed but ignored, and so have no effect in MySQL 8.0. These options are intended for future expansion.

When used with **ENGINE [=] NDB**, a log file group and associated **UNDO** log file are created on each Cluster data node. You can verify that the **UNDO** files were created and obtain information about them by querying the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table. For example:

mysql> **SELECT LOGFILE\_GROUP\_NAME, LOGFILE\_GROUP\_NUMBER, EXTRA**

-> **FROM INFORMATION\_SCHEMA.FILES**

-> **WHERE FILE\_NAME = 'undo\_10.dat';**

+--------------------+----------------------+----------------+

| LOGFILE\_GROUP\_NAME | LOGFILE\_GROUP\_NUMBER | EXTRA |

+--------------------+----------------------+----------------+

| lg\_3 | 11 | CLUSTER\_NODE=3 |

| lg\_3 | 11 | CLUSTER\_NODE=4 |

+--------------------+----------------------+----------------+

2 rows in set (0.06 sec)

[**CREATE LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group) is useful only with Disk Data storage for NDB Cluster. See [Section 23.5.10, “NDB Cluster Disk Data Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data).

### 13.1.17 CREATE PROCEDURE and CREATE FUNCTION Statements

CREATE

[DEFINER = ***user***]

PROCEDURE ***sp\_name*** ([***proc\_parameter***[,...]])

[***characteristic*** ...] ***routine\_body***

CREATE

[DEFINER = ***user***]

FUNCTION ***sp\_name*** ([***func\_parameter***[,...]])

RETURNS ***type***

[***characteristic*** ...] ***routine\_body***

***proc\_parameter***:

[ IN | OUT | INOUT ] ***param\_name*** ***type***

***func\_parameter***:

***param\_name*** ***type***

***type***:

***Any valid MySQL data type***

***characteristic***: {

COMMENT '***string***'

| LANGUAGE SQL

| [NOT] DETERMINISTIC

| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

| SQL SECURITY { DEFINER | INVOKER }

}

***routine\_body***:

***Valid SQL routine statement***

These statements are used to create a stored routine (a stored procedure or function). That is, the specified routine becomes known to the server. By default, a stored 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.

The **CREATE FUNCTION** statement is also used in MySQL to support UDFs (user-defined functions). See [Section 13.7.4.1, “CREATE FUNCTION Statement for User-Defined Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf). A UDF can be regarded as an external stored function. Stored functions share their namespace with UDFs. See [Section 9.2.5, “Function Name Parsing and Resolution”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#function-resolution), for the rules describing how the server interprets references to different kinds of functions.

To invoke a stored procedure, use the [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statement (see [Section 13.2.1, “CALL Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call)). To invoke a stored function, refer to it in an expression. The function returns a value during expression evaluation.

[**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure) and [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function) require the [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine) privilege. If the **DEFINER** clause is present, the privileges required depend on the ***user*** value, as discussed in [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security). If binary logging is enabled, [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function) might require the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege, as discussed in [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging).

By default, MySQL automatically grants the [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine) and [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute) privileges to the routine creator. This behavior can be changed by disabling the [**automatic\_sp\_privileges**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_automatic_sp_privileges) system variable. See [Section 25.2.2, “Stored Routines and MySQL Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-routines-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 in this section.

If the routine name is the same as the name of a built-in SQL function, a syntax error occurs unless you use a space between the name and the following parenthesis when defining the routine or invoking it later. For this reason, avoid using the names of existing SQL functions for your own stored routines.

The [**IGNORE\_SPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_ignore_space) SQL mode applies to built-in functions, not to stored routines. It is always permissible to have spaces after a stored routine name, regardless of whether [**IGNORE\_SPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_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 is an **IN** parameter by default. To specify otherwise for a parameter, use the keyword **OUT** or **INOUT** before the parameter name.

**Note**

Specifying a parameter as **IN**, **OUT**, or **INOUT** is valid only for a **PROCEDURE**. For a **FUNCTION**, parameters are always regarded as **IN** parameters.

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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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 **OUT** or **INOUT** parameter. If you are calling the procedure from within a trigger, you can also pass **NEW.*col\_name*** as an **OUT** or **INOUT** parameter.

For information about the effect of unhandled conditions on procedure parameters, see [Section 13.6.7.8, “Condition Handling and OUT or INOUT Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#conditions-and-parameters).

Routine parameters cannot be referenced in statements prepared within the routine; see [Section 25.8, “Restrictions on Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-program-restrictions).

The following example shows a simple stored procedure that, given a country code, counts the number of cities for that country that appear in the **city** table of the **world** database. The country code is passed using an **IN** parameter, and the city count is returned using an **OUT** parameter:

mysql> **delimiter //**

mysql> **CREATE PROCEDURE citycount (IN country CHAR(3), OUT cities INT)**

**BEGIN**

**SELECT COUNT(\*) INTO cities FROM world.city**

**WHERE CountryCode = country;**

**END//**

Query OK, 0 rows affected (0.01 sec)

mysql> **delimiter ;**

mysql> **CALL citycount('JPN', @cities); -- cities in Japan**

Query OK, 1 row affected (0.00 sec)

mysql> **SELECT @cities;**

+---------+

| @cities |

+---------+

| 248 |

+---------+

1 row in set (0.00 sec)

mysql> **CALL citycount('FRA', @cities); -- cities in France**

Query OK, 1 row affected (0.00 sec)

mysql> **SELECT @cities;**

+---------+

| @cities |

+---------+

| 40 |

+---------+

1 row in set (0.00 sec)

The example uses the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client **delimiter** command to change the statement delimiter from **;** to **//** while the procedure is being defined. This enables the **;** delimiter used in the procedure body to be passed through to the server rather than being interpreted by [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) itself. See [Section 25.1, “Defining Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-defining).

The **RETURNS** clause may be specified only for a **FUNCTION**, for which it is mandatory. It indicates the return type of the function, and the function body must contain a **RETURN *value*** statement. If the [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) statement returns a value of a different type, the value is coerced to the proper type. For example, if a function specifies an [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) or [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) value in the **RETURNS** clause, but the [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) statement returns an integer, the value returned from the function is the string for the corresponding [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) member of set of [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) members.

The following example function takes a parameter, performs an operation using an SQL function, and returns the result. In this case, it is unnecessary to use **delimiter** because the function definition contains no internal **;** statement delimiters:

mysql> **CREATE FUNCTION hello (s CHAR(20))**

mysql> **RETURNS CHAR(50) DETERMINISTIC**

**RETURN CONCAT('Hello, ',s,'!');**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT hello('world');**

+----------------+

| hello('world') |

+----------------+

| Hello, world! |

+----------------+

1 row in set (0.00 sec)

Parameter types and function return types can be declared to use any valid data type. The **COLLATE** attribute can be used if preceded by a **CHARACTER SET** specification.

The ***routine\_body*** consists of a valid SQL routine statement. This can be a simple statement such as [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), or a compound statement written using **BEGIN** and **END**. Compound statements can contain declarations, loops, and other control structure statements. The syntax for these statements is described in [Section 13.6, “Compound Statement Syntax”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#sql-compound-statements). In practice, stored functions tend to use compound statements, unless the body consists of a single [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) statement.

MySQL permits routines to contain DDL statements, such as **CREATE** and **DROP**. MySQL also permits stored procedures (but not stored functions) to contain SQL transaction statements such as [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit). Stored functions may not contain statements that perform explicit or implicit commit or rollback. Support for these statements is not required by the SQL standard, which states that each DBMS vendor may decide whether to permit them.

Statements that return a result set can be used within a stored procedure but not within a stored function. This prohibition includes [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements that do not have an **INTO *var\_list*** clause and other statements such as [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show), [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain), and [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table). For statements that can be determined at function definition time to return a result set, a **Not allowed to return a result set from a function** error occurs ([**ER\_SP\_NO\_RETSET**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_sp_no_retset)). For statements that can be determined only at runtime to return a result set, a **PROCEDURE %s can't return a result set in the given context** error occurs ([**ER\_SP\_BADSELECT**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_sp_badselect)).

[**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statements within stored routines are not permitted. When a routine is invoked, an implicit **USE *db\_name*** is performed (and undone when the routine terminates). The causes the routine to have the given default database while it executes. References to objects in databases other than the routine default database should be qualified with the appropriate database name.

For additional information about statements that are not permitted in stored routines, see [Section 25.8, “Restrictions on Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-program-restrictions).

For information about invoking stored procedures from within programs written in a language that has a MySQL interface, see [Section 13.2.1, “CALL Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call).

MySQL stores the [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) system variable setting in effect when a routine is created or altered, and always executes the routine with this setting in force, regardless of the current server SQL mode when the routine begins executing.

The switch from the SQL mode of the invoker to that of the routine occurs after evaluation of arguments and assignment of the resulting values to routine parameters. If you define a routine in strict SQL mode but invoke it in nonstrict mode, assignment of arguments to routine parameters does not take place in strict mode. If you require that expressions passed to a routine be assigned in strict SQL mode, you should invoke the routine with strict mode in effect.

The **COMMENT** characteristic is a MySQL extension, and may be used to describe the stored routine. This information is displayed by the [**SHOW CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-procedure) and [**SHOW CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-function) statements.

The **LANGUAGE** characteristic indicates the language in which the routine is written. The server ignores this characteristic; only SQL routines are supported.

A routine is considered “deterministic” if it always produces the same result for the same input parameters, and “not deterministic” otherwise. If neither **DETERMINISTIC** nor **NOT DETERMINISTIC** is given in the routine definition, the default is **NOT DETERMINISTIC**. To declare that a function is deterministic, you must specify **DETERMINISTIC** explicitly.

Assessment of the nature of a routine is based on the “honesty” of the creator: MySQL does not check that a routine declared **DETERMINISTIC** is free of statements that produce nondeterministic results. However, misdeclaring a routine might affect results or affect performance. Declaring a nondeterministic routine as **DETERMINISTIC** might lead to unexpected results by causing the optimizer to make incorrect execution plan choices. Declaring a deterministic routine as **NONDETERMINISTIC** might diminish performance by causing available optimizations not to be used.

If binary logging is enabled, the **DETERMINISTIC** characteristic affects which routine definitions MySQL accepts. See [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging).

A routine that contains the [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now) function (or its synonyms) or [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) is nondeterministic, but it might still be replication-safe. For [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now), the binary log includes the timestamp and replicates correctly. [**RAND()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_rand) also replicates correctly as long as it is called only a single time during the execution of a routine. (You can consider the routine execution timestamp and random number seed as implicit inputs that are identical on the source and replica.)

Several characteristics provide information about the nature of data use by the routine. In MySQL, these characteristics are advisory only. The server does not use them to constrain what kinds of statements a routine is permitted to execute.

**CONTAINS SQL** indicates that the routine does not contain statements that read or write data. This is the default if none of these characteristics is given explicitly. Examples of such statements are **SET @x = 1** or **DO RELEASE\_LOCK('abc')**, which execute but neither read nor write data.

**NO SQL** indicates that the routine contains no SQL statements.

**READS SQL DATA** indicates that the routine contains statements that read data (for example, [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)), but not statements that write data.

**MODIFIES SQL DATA** indicates that the routine contains statements that may write data (for example, [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete)).

The **SQL SECURITY** characteristic can be **DEFINER** or **INVOKER** to specify the security context; that is, whether the routine executes using the privileges of the account named in the routine **DEFINER** clause or the user who invokes it. This account must have permission to access the database with which the routine is associated. The default value is **DEFINER**. The user who invokes the routine must have the [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute) privilege for it, as must the **DEFINER** account if the routine executes in definer security context.

The **DEFINER** clause specifies the MySQL account to be used when checking access privileges at routine execution time for routines that have the **SQL SECURITY DEFINER** characteristic.

If the **DEFINER** clause is present, the ***user*** value should be a MySQL account specified as **'*user\_name*'@'*host\_name*'**, [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user), or [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user). The permitted ***user*** values depend on the privileges you hold, as discussed in [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security). Also see that section for additional information about stored routine security.

If the **DEFINER** clause is omitted, the default definer is the user who executes the [**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure) or [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function) statement. This is the same as specifying **DEFINER = CURRENT\_USER** explicitly.

Within the body of a stored routine that is defined with the **SQL SECURITY DEFINER** characteristic, the [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) function returns the routine's **DEFINER** value. For information about user auditing within stored routines, see [Section 6.2.22, “SQL-Based Account Activity Auditing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-activity-auditing).

Consider the following procedure, which displays a count of the number of MySQL accounts listed in the **mysql.user** system table:

CREATE DEFINER = 'admin'@'localhost' PROCEDURE account\_count()

BEGIN

SELECT 'Number of accounts:', COUNT(\*) FROM mysql.user;

END;

The procedure is assigned a **DEFINER** account of **'admin'@'localhost'** no matter which user defines it. It executes with the privileges of that account no matter which user invokes it (because the default security characteristic is **DEFINER**). The procedure succeeds or fails depending on whether invoker has the [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute) privilege for it and **'admin'@'localhost'** has the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for the **mysql.user** table.

Now suppose that the procedure is defined with the **SQL SECURITY INVOKER** characteristic:

CREATE DEFINER = 'admin'@'localhost' PROCEDURE account\_count()

SQL SECURITY INVOKER

BEGIN

SELECT 'Number of accounts:', COUNT(\*) FROM mysql.user;

END;

The procedure still has a **DEFINER** of **'admin'@'localhost'**, but in this case, it executes with the privileges of the invoking user. Thus, the procedure succeeds or fails depending on whether the invoker has the [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute) privilege for it and the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for the **mysql.user** table.

The server handles the data type of a routine parameter, local routine variable created with [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare), or function return value as follows:

Assignments are checked for data type mismatches and overflow. Conversion and overflow problems result in warnings, or errors in strict SQL mode.

Only scalar values can be assigned. For example, a statement such as **SET x = (SELECT 1, 2)** is invalid.

For character data types, if **CHARACTER SET** is includedd in the declaration, the specified character set and its default collation is used. If the **COLLATE** attribute is also present, that collation is used rather than the default collation.

If **CHARACTER SET** and **COLLATE** are not present, the database character set and collation in effect at routine creation time are used. To avoid having the server use the database character set and collation, provide an explicit **CHARACTER SET** and a **COLLATE** attribute for character data parameters.

If you alter the database default character set or collation, stored routines that are to use the new database defaults must be dropped and recreated.

The database character set and collation are given by the value of the [**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database) and [**collation\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_database) system variables. For more information, see [Section 10.3.3, “Database Character Set and Collation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-database).

### 13.1.18 CREATE SERVER Statement

CREATE SERVER ***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***

}

This statement creates the definition of a server for use with the **FEDERATED** storage engine. The **CREATE SERVER** statement creates a new row in the **servers** table in the **mysql** database. This statement requires the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) 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*** is an identifier and may be quoted with single quotation marks.

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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement when creating a **FEDERATED** table. The options that you specify are used to populate the columns in the **mysql.servers** table. The table columns are **Server\_name**, **Host**, **Db**, **Username**, **Password**, **Port** and **Socket**.

For example:

CREATE SERVER s

FOREIGN DATA WRAPPER mysql

OPTIONS (USER 'Remote', HOST '198.51.100.106', DATABASE 'test');

Be sure to specify all options necessary to establish a connection to the server. The user name, host name, and database name are mandatory. Other options might be required as well, such as password.

The data stored in the table can be used when creating a connection to a **FEDERATED** table:

CREATE TABLE t (s1 INT) ENGINE=FEDERATED CONNECTION='s';

For more information, see [Section 16.8, “The FEDERATED Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-storage-engine).

**CREATE SERVER** causes an implicit commit. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

**CREATE SERVER** is not written to the binary log, regardless of the logging format that is in use.

### 13.1.19 CREATE SPATIAL REFERENCE SYSTEM Statement

CREATE OR REPLACE SPATIAL REFERENCE SYSTEM

***srid*** ***srs\_attribute*** ...

CREATE SPATIAL REFERENCE SYSTEM

[IF NOT EXISTS]

***srid*** ***srs\_attribute*** ...

***srs\_attribute***: {

NAME '***srs\_name***'

| DEFINITION '***definition***'

| ORGANIZATION '***org\_name***' IDENTIFIED BY ***org\_id***

| DESCRIPTION '***description***'

}

***srid***, ***org\_id***: ***32-bit unsigned integer***

This statement creates a [spatial reference system](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-reference-systems) (SRS) definition and stores it in the data dictionary. It requires the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege. The resulting data dictionary entry can be inspected using the **INFORMATION\_SCHEMA** [**ST\_SPATIAL\_REFERENCE\_SYSTEMS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-st-spatial-reference-systems-table) table.

SRID values must be unique, so if neither **OR REPLACE** nor **IF NOT EXISTS** is specified, an error occurs if an SRS definition with the given ***srid*** value already exists.

With **CREATE OR REPLACE** syntax, any existing SRS definition with the same SRID value is replaced, unless the SRID value is used by some column in an existing table. In that case, an error occurs. For example:

mysql> **CREATE OR REPLACE SPATIAL REFERENCE SYSTEM 4326 ...;**

ERROR 3716 (SR005): Can't modify SRID 4326. There is at

least one column depending on it.

To identify which column or columns use the SRID, use this query, replacing 4326 with the SRID of the definition you are trying to create:

SELECT \* FROM INFORMATION\_SCHEMA.ST\_GEOMETRY\_COLUMNS WHERE SRS\_ID=4326;

With **CREATE ... IF NOT EXISTS** syntax, any existing SRS definition with the same SRID value causes the new definition to be ignored and a warning occurs.

SRID values must be in the range of 32-bit unsigned integers, with these restrictions:

SRID 0 is a valid SRID but cannot be used with [**CREATE SPATIAL REFERENCE SYSTEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-spatial-reference-system).

If the value is in a reserved SRID range, a warning occurs. Reserved ranges are [0, 32767] (reserved by EPSG), [60,000,000, 69,999,999] (reserved by EPSG), and [2,000,000,000, 2,147,483,647] (reserved by MySQL). EPSG stands for the [European Petroleum Survey Group](http://epsg.org).

Users should not create SRSs with SRIDs in the reserved ranges. Doing so runs the risk of the SRIDs conflicting with future SRS definitions distributed with MySQL, with the result that the new system-provided SRSs are not installed for MySQL upgrades or that the user-defined SRSs are overwritten.

Attributes for the statement must satisfy these conditions:

Attributes can be given in any order, but no attribute can be given more than once.

The **NAME** and **DEFINITION** attributes are mandatory.

The **NAME** ***srs\_name*** attribute value must be unique. The combination of the **ORGANIZATION** ***org\_name*** and ***org\_id*** attribute values must be unique.

The **NAME** ***srs\_name*** attribute value and **ORGANIZATION** ***org\_name*** attribute value cannot be empty or begin or end with whitespace.

String values in attribute specifications cannot contain control characters, including newline.

The following table shows the maximum lengths for string attribute values.

**Table 13.6 CREATE SPATIAL REFERENCE SYSTEM Attribute Lengths**

| **Attribute** | **Maximum Length (characters)** |
| --- | --- |
| **NAME** | 80 |
| **DEFINITION** | 4096 |
| **ORGANIZATION** | 256 |
| **DESCRIPTION** | 2048 |

Here is an example [**CREATE SPATIAL REFERENCE SYSTEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-spatial-reference-system) statement. The **DEFINITION** value is reformatted across multiple lines for readability. (For the statement to be legal, the value actually must be given on a single line.)

CREATE SPATIAL REFERENCE SYSTEM 4120

NAME 'Greek'

ORGANIZATION 'EPSG' IDENTIFIED BY 4120

DEFINITION

'GEOGCS["Greek",DATUM["Greek",SPHEROID["Bessel 1841",

6377397.155,299.1528128,AUTHORITY["EPSG","7004"]],

AUTHORITY["EPSG","6120"]],PRIMEM["Greenwich",0,

AUTHORITY["EPSG","8901"]],UNIT["degree",0.017453292519943278,

AUTHORITY["EPSG","9122"]],AXIS["Lat",NORTH],AXIS["Lon",EAST],

AUTHORITY["EPSG","4120"]]';

The grammar for SRS definitions is based on the grammar defined in OpenGIS Implementation Specification: Coordinate Transformation Services, Revision 1.00, OGC 01-009, January 12, 2001, Section 7.2. This specification is available at <http://www.opengeospatial.org/standards/ct>.

MySQL incorporates these changes to the specification:

Only the **<horz cs>** production rule is implemented (that is, geographic and projected SRSs).

There is an optional, nonstandard **<authority>** clause for **<parameter>**. This makes it possible to recognize projection parameters by authority instead of name.

The specification does not make **AXIS** clauses mandatory in **GEOGCS** spatial reference system definitions. However, if there are no **AXIS** clauses, MySQL cannot determine whether a definition has axes in latitude-longitude order or longitude-latitude order. MySQL enforces the nonstandard requirement that each **GEOGCS** definition must include two **AXIS** clauses. One must be **NORTH** or **SOUTH**, and the other **EAST** or **WEST**. The **AXIS** clause order determines whether the definition has axes in latitude-longitude order or longitude-latitude order.

SRS definitions may not contain newlines.

If an SRS definition specifies an authority code for the projection (which is recommended), an error occurs if the definition is missing mandatory parameters. In this case, the error message indicates what the problem is. The projection methods and mandatory parameters that MySQL supports are shown in [Table 13.7, “Supported Spatial Reference System Projection Methods”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#supported-srs-projections-table) and [Table 13.8, “Spatial Reference System Projection Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#srs-projection-parameters-table).

For additional information about writing SRS definitions for MySQL, see [Geographic Spatial Reference Systems in MySQL 8.0](https://mysqlserverteam.com/geographic-spatial-reference-systems-in-mysql-8-0/) and [Projected Spatial Reference Systems in MySQL 8.0](https://mysqlserverteam.com/projected-spatial-reference-systems-in-mysql-8-0/)

The following table shows the projection methods that MySQL supports. MySQL permits unknown projection methods but cannot check the defintion for mandatory paramters and cannot convert spatial data to or from an unknown projection. For detailed explanations of how each projection works, including formulas, see [EPSG Guidance Note 7-2](http://www.epsg.org/Portals/0/373-07-2.pdf).

**Table 13.7 Supported Spatial Reference System Projection Methods**

| **EPSG Code** | **Projection Name** | **Mandatory Parameters (EPSG Codes)** |
| --- | --- | --- |
| **1024** | Popular Visualisation Pseudo Mercator | 8801, 8802, 8806, 8807 |
| **1027** | Lambert Azimuthal Equal Area (Spherical) | 8801, 8802, 8806, 8807 |
| **1028** | Equidistant Cylindrical | 8823, 8802, 8806, 8807 |
| **1029** | **Equidistant Cylindrical (Spherical)** | **8823, 8802, 8806, 8807** |
| **1041** | **Krovak (North Orientated)** | **8811, 8833, 1036, 8818, 8819, 8806, 8807** |
| **1042** | **Krovak Modified** | **8811, 8833, 1036, 8818, 8819, 8806, 8807, 8617, 8618, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035** |
| **1043** | **Krovak Modified (North Orientated)** | **8811, 8833, 1036, 8818, 8819, 8806, 8807, 8617, 8618, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035** |
| **1051** | **Lambert Conic Conformal (2SP Michigan)** | **8821, 8822, 8823, 8824, 8826, 8827, 1038** |
| **1052** | **Colombia Urban** | **8801, 8802, 8806, 8807, 1039** |
| **9801** | **Lambert Conic Conformal (1SP)** | **8801, 8802, 8805, 8806, 8807** |
| **9802** | **Lambert Conic Conformal (2SP)** | **8821, 8822, 8823, 8824, 8826, 8827** |
| **9803** | **Lambert Conic Conformal (2SP Belgium)** | **8821, 8822, 8823, 8824, 8826, 8827** |
| **9804** | **Mercator (variant A)** | **8801, 8802, 8805, 8806, 8807** |
| **9805** | **Mercator (variant B)** | **8823, 8802, 8806, 8807** |
| **9806** | **Cassini-Soldner** | **8801, 8802, 8806, 8807** |
| **9807** | **Transverse Mercator** | **8801, 8802, 8805, 8806, 8807** |
| **9808** | **Transverse Mercator (South Orientated)** | **8801, 8802, 8805, 8806, 8807** |
| **9809** | **Oblique Stereographic** | **8801, 8802, 8805, 8806, 8807** |
| **9810** | **Polar Stereographic (variant A)** | **8801, 8802, 8805, 8806, 8807** |
| **9811** | **New Zealand Map Grid** | **8801, 8802, 8806, 8807** |
| **9812** | **Hotine Oblique Mercator (variant A)** | **8811, 8812, 8813, 8814, 8815, 8806, 8807** |
| **9813** | **Laborde Oblique Mercator** | **8811, 8812, 8813, 8815, 8806, 8807** |
| **9815** | **Hotine Oblique Mercator (variant B)** | **8811, 8812, 8813, 8814, 8815, 8816, 8817** |
| **9816** | **Tunisia Mining Grid** | **8821, 8822, 8826, 8827** |
| **9817** | **Lambert Conic Near-Conformal** | **8801, 8802, 8805, 8806, 8807** |
| **9818** | **American Polyconic** | **8801, 8802, 8806, 8807** |
| **9819** | **Krovak** | **8811, 8833, 1036, 8818, 8819, 8806, 8807** |
| **9820** | **Lambert Azimuthal Equal Area** | **8801, 8802, 8806, 8807** |
| **9822** | **Albers Equal Area** | **8821, 8822, 8823, 8824, 8826, 8827** |
| **9824** | **Transverse Mercator Zoned Grid System** | **8801, 8830, 8831, 8805, 8806, 8807** |
| **9826** | **Lambert Conic Conformal (West Orientated)** | **8801, 8802, 8805, 8806, 8807** |
| **9828** | **Bonne (South Orientated)** | **8801, 8802, 8806, 8807** |
| **9829** | **Polar Stereographic (variant B)** | **8832, 8833, 8806, 8807** |
| **9830** | **Polar Stereographic (variant C)** | **8832, 8833, 8826, 8827** |
| **9831** | **Guam Projection** | **8801, 8802, 8806, 8807** |
| **9832** | **Modified Azimuthal Equidistant** | **8801, 8802, 8806, 8807** |
| **9833** | **Hyperbolic Cassini-Soldner** | **8801, 8802, 8806, 8807** |
| **9834** | **Lambert Cylindrical Equal Area (Spherical)** | **8823, 8802, 8806, 8807** |
| **9835** | Lambert Cylindrical Equal Area | 8823, 8802, 8806, 8807 |

The following table shows the projection parameters that MySQL recognizes. Recognition occurs primarily by authority code. If there is no authority code, MySQL falls back to case-insensitive string matching on the parameter name. For details about each parameter, look it up by code in the [EPSG Online Registry](https://www.epsg-registry.org).

**Table 13.8 Spatial Reference System Projection Parameters**

| **EPSG Code** | **Fallback Name (Recognized by MySQL)** | **EPSG Name** |
| --- | --- | --- |
| **1026** | **c1** | **C1** |
| **1027** | **c2** | **C2** |
| **1028** | **c3** | **C3** |
| **1029** | **c4** | **C4** |
| **1030** | **c5** | **C5** |
| **1031** | **c6** | **C6** |
| **1032** | **c7** | **C7** |
| **1033** | **c8** | **C8** |
| **1034** | **c9** | **C9** |
| **1035** | **c10** | **C10** |
| **1036** | **azimuth** | **Co-latitude of cone axis** |
| **1038** | **ellipsoid\_scale\_factor** | **Ellipsoid scaling factor** |
| **1039** | **projection\_plane\_height\_at\_origin** | **Projection plane origin height** |
| **8617** | **evaluation\_point\_ordinate\_1** | **Ordinate 1 of evaluation point** |
| **8618** | **evaluation\_point\_ordinate\_2** | **Ordinate 2 of evaluation point** |
| **8801** | **latitude\_of\_origin** | **Latitude of natural origin** |
| **8802** | **central\_meridian** | **Longitude of natural origin** |
| **8805** | **scale\_factor** | **Scale factor at natural origin** |
| **8806** | **false\_easting** | **False easting** |
| **8807** | **false\_northing** | **False northing** |
| **8811** | **latitude\_of\_center** | **Latitude of projection centre** |
| **8812** | **longitude\_of\_center** | **Longitude of projection centre** |
| **8813** | **azimuth** | **Azimuth of initial line** |
| **8814** | **rectified\_grid\_angle** | **Angle from Rectified to Skew Grid** |
| **8815** | **scale\_factor** | **Scale factor on initial line** |
| **8816** | **false\_easting** | **Easting at projection centre** |
| **8817** | **false\_northing** | **Northing at projection centre** |
| **8818** | **pseudo\_standard\_parallel\_1** | **Latitude of pseudo standard parallel** |
| **8819** | **scale\_factor** | **Scale factor on pseudo standard parallel** |
| **8821** | **latitude\_of\_origin** | **Latitude of false origin** |
| **8822** | **central\_meridian** | **Longitude of false origin** |
| **8823** | **standard\_parallel\_1, standard\_parallel1** | **Latitude of 1st standard parallel** |
| **8824** | **standard\_parallel\_2, standard\_parallel2** | **Latitude of 2nd standard parallel** |
| **8826** | **false\_easting** | **Easting at false origin** |
| **8827** | **false\_northing** | **Northing at false origin** |
| **8830** | **initial\_longitude** | **Initial longitude** |
| **8831** | **zone\_width** | **Zone width** |
| **8832** | **standard\_parallel** | **Latitude of standard parallel** |
| **8833** | longitude\_of\_center | Longitude of origin |

### 13.1.20 CREATE TABLE Statement

[13.1.20.1 Files Created by CREATE TABLE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-files)

[13.1.20.2 CREATE TEMPORARY TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-temporary-table)

[13.1.20.3 CREATE TABLE ... LIKE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like)

[13.1.20.4 CREATE TABLE ... SELECT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select)

[13.1.20.5 FOREIGN KEY Constraints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys)

[13.1.20.6 CHECK Constraints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-check-constraints)

[13.1.20.7 Silent Column Specification Changes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#silent-column-changes)

[13.1.20.8 CREATE TABLE and Generated Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-generated-columns)

[13.1.20.9 Secondary Indexes and Generated Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-secondary-indexes)

[13.1.20.10 Invisible Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#invisible-columns)

[13.1.20.11 Setting NDB\_TABLE Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-ndb-table-comment-options)

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] ***tbl\_name***

(***create\_definition***,...)

[***table\_options***]

[***partition\_options***]

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] ***tbl\_name***

[(***create\_definition***,...)]

[***table\_options***]

[***partition\_options***]

[IGNORE | REPLACE]

[AS] ***query\_expression***

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] ***tbl\_name***

{ LIKE ***old\_tbl\_name*** | (LIKE ***old\_tbl\_name***) }

***create\_definition***: {

***col\_name*** ***column\_definition***

| {INDEX | KEY} [***index\_name***] [***index\_type***] (***key\_part***,...)

[***index\_option***] ...

| {FULLTEXT | SPATIAL} [INDEX | KEY] [***index\_name***] (***key\_part***,...)

[***index\_option***] ...

| [CONSTRAINT [***symbol***]] PRIMARY KEY

[***index\_type***] (***key\_part***,...)

[***index\_option***] ...

| [CONSTRAINT [***symbol***]] UNIQUE [INDEX | KEY]

[***index\_name***] [***index\_type***] (***key\_part***,...)

[***index\_option***] ...

| [CONSTRAINT [***symbol***]] FOREIGN KEY

[***index\_name***] (***col\_name***,...)

***reference\_definition***

| ***check\_constraint\_definition***

}

***column\_definition***: {

***data\_type*** [NOT NULL | NULL] [DEFAULT {***literal*** | (***expr***)} ]

[VISIBLE | INVISIBLE]

[AUTO\_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]

[COMMENT '***string***']

[COLLATE ***collation\_name***]

[COLUMN\_FORMAT {FIXED | DYNAMIC | DEFAULT}]

[ENGINE\_ATTRIBUTE [=] '***string***']

[SECONDARY\_ENGINE\_ATTRIBUTE [=] '***string***']

[STORAGE {DISK | MEMORY}]

[***reference\_definition***]

[***check\_constraint\_definition***]

| ***data\_type***

[COLLATE ***collation\_name***]

[GENERATED ALWAYS] AS (***expr***)

[VIRTUAL | STORED] [NOT NULL | NULL]

[VISIBLE | INVISIBLE]

[UNIQUE [KEY]] [[PRIMARY] KEY]

[COMMENT '***string***']

[***reference\_definition***]

[***check\_constraint\_definition***]

}

***data\_type***:

(see [Chapter 11, *Data Types*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html))

***key\_part***: {***col\_name*** [(***length***)] | (***expr***)} [ASC | DESC]

***index\_type***:

USING {BTREE | HASH}

***index\_option***: {

KEY\_BLOCK\_SIZE [=] ***value***

| ***index\_type***

| WITH PARSER ***parser\_name***

| COMMENT '***string***'

| {VISIBLE | INVISIBLE}

|**ENGINE\_ATTRIBUTE** [=] '***string***'

|**SECONDARY\_ENGINE\_ATTRIBUTE** [=] '***string***'

}

***check\_constraint\_definition***:

[CONSTRAINT [***symbol***]] CHECK (***expr***) [[NOT] ENFORCED]

***reference\_definition***:

REFERENCES ***tbl\_name*** (***key\_part***,...)

[MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]

[ON DELETE ***reference\_option***]

[ON UPDATE ***reference\_option***]

***reference\_option***:

RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

***table\_options***:

***table\_option*** [[,] ***table\_option***] ...

***table\_option***: {

AUTOEXTEND\_SIZE [=] ***value***

| AUTO\_INCREMENT [=] ***value***

| AVG\_ROW\_LENGTH [=] ***value***

| [DEFAULT] CHARACTER SET [=] ***charset\_name***

| CHECKSUM [=] {0 | 1}

| [DEFAULT] COLLATE [=] ***collation\_name***

| COMMENT [=] '***string***'

| COMPRESSION [=] {'ZLIB' | 'LZ4' | 'NONE'}

| CONNECTION [=] '***connect\_string***'

| {DATA | INDEX} DIRECTORY [=] '***absolute path to directory***'

| DELAY\_KEY\_WRITE [=] {0 | 1}

| ENCRYPTION [=] {'Y' | 'N'}

| ENGINE [=] ***engine\_name***

| ENGINE\_ATTRIBUTE [=] '***string***'

| INSERT\_METHOD [=] { NO | FIRST | LAST }

| KEY\_BLOCK\_SIZE [=] ***value***

| MAX\_ROWS [=] ***value***

| MIN\_ROWS [=] ***value***

| PACK\_KEYS [=] {0 | 1 | DEFAULT}

| PASSWORD [=] '***string***'

| ROW\_FORMAT [=] {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}

| SECONDARY\_ENGINE\_ATTRIBUTE [=] '***string***'

| STATS\_AUTO\_RECALC [=] {DEFAULT | 0 | 1}

| STATS\_PERSISTENT [=] {DEFAULT | 0 | 1}

| STATS\_SAMPLE\_PAGES [=] ***value***

| TABLESPACE ***tablespace\_name*** [STORAGE {DISK | MEMORY}]

| UNION [=] (***tbl\_name***[,***tbl\_name***]...)

}

***partition\_options***:

PARTITION BY

{ [LINEAR] HASH(***expr***)

| [LINEAR] KEY [ALGORITHM={1 | 2}] (***column\_list***)

| RANGE{(***expr***) | COLUMNS(***column\_list***)}

| LIST{(***expr***) | COLUMNS(***column\_list***)} }

[PARTITIONS ***num***]

[SUBPARTITION BY

{ [LINEAR] HASH(***expr***)

| [LINEAR] KEY [ALGORITHM={1 | 2}] (***column\_list***) }

[SUBPARTITIONS ***num***]

]

[(***partition\_definition*** [, ***partition\_definition***] ...)]

***partition\_definition***:

PARTITION ***partition\_name***

[VALUES

{LESS THAN {(***expr*** | ***value\_list***) | MAXVALUE}

|

IN (***value\_list***)}]

[[STORAGE] ENGINE [=] ***engine\_name***]

[COMMENT [=] '***string***' ]

[DATA DIRECTORY [=] '***data\_dir***']

[INDEX DIRECTORY [=] '***index\_dir***']

[MAX\_ROWS [=] ***max\_number\_of\_rows***]

[MIN\_ROWS [=] ***min\_number\_of\_rows***]

[TABLESPACE [=] tablespace\_name]

[(***subpartition\_definition*** [, ***subpartition\_definition***] ...)]

***subpartition\_definition***:

SUBPARTITION ***logical\_name***

[[STORAGE] ENGINE [=] ***engine\_name***]

[COMMENT [=] '***string***' ]

[DATA DIRECTORY [=] '***data\_dir***']

[INDEX DIRECTORY [=] '***index\_dir***']

[MAX\_ROWS [=] ***max\_number\_of\_rows***]

[MIN\_ROWS [=] ***min\_number\_of\_rows***]

[TABLESPACE [=] tablespace\_name]

***query\_expression:***

SELECT ... (***Some valid select or union statement***)

[**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) creates a table with the given name. You must have the [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create) privilege for the table.

By default, tables are created in the default database, using the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) storage engine. An error occurs if the table exists, if there is no default database, or if the database does not exist.

MySQL has no limit on the number of tables. The underlying file system may have a limit on the number of files that represent tables. Individual storage engines may impose engine-specific constraints. **InnoDB** permits up to 4 billion tables.

For information about the physical representation of a table, see [Section 13.1.20.1, “Files Created by CREATE TABLE”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-files).

There are several aspects to the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, described under the following topics in this section:

[Table Name](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-name)

[Temporary Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-temporary-tables)

[Table Cloning and Copying](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-clone-copy)

[Column Data Types and Attributes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-types-attributes)

[Indexes, Foreign Keys, and CHECK Constraints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-indexes-keys)

[Table Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-options)

[Table Partitioning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-partitioning)

#### Table Name

***tbl\_name***

The table name can be specified as ***db\_name.tbl\_name*** to create the table in a specific database. This works regardless of whether there is a default database, assuming that the database exists. If you use quoted identifiers, quote the database and table names separately. For example, write **`mydb`.`mytbl`**, not **`mydb.mytbl`**.

Rules for permissible table names are given in [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers).

**IF NOT EXISTS**

Prevents an error from occurring if the table exists. However, there is no verification that the existing table has a structure identical to that indicated by the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement.

#### Temporary Tables

You can use the **TEMPORARY** keyword when creating a table. A **TEMPORARY** table is visible only within the current session, and is dropped automatically when the session is closed. For more information, see [Section 13.1.20.2, “CREATE TEMPORARY TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-temporary-table).

#### Table Cloning and Copying

**LIKE**

Use **CREATE TABLE ... LIKE** to create an empty table based on the definition of another table, including any column attributes and indexes defined in the original table:

CREATE TABLE ***new\_tbl*** LIKE ***orig\_tbl***;

For more information, see [Section 13.1.20.3, “CREATE TABLE ... LIKE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like).

**[AS] *query\_expression***

To create one table from another, add a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement at the end of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement:

CREATE TABLE ***new\_tbl*** AS SELECT \* FROM ***orig\_tbl***;

For more information, see [Section 13.1.20.4, “CREATE TABLE ... SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select).

**IGNORE | REPLACE**

The **IGNORE** and **REPLACE** options indicate how to handle rows that duplicate unique key values when copying a table using a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement.

For more information, see [Section 13.1.20.4, “CREATE TABLE ... SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select).

#### Column Data Types and Attributes

There is a hard limit of 4096 columns per table, but the effective maximum may be less for a given table and depends on the factors discussed in [Section 8.4.7, “Limits on Table Column Count and Row Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#column-count-limit).

***data\_type***

***data\_type*** represents the data type in a column definition. For a full description of the syntax available for specifying column data types, as well as information about the properties of each type, see [Chapter 11, *Data Types*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html).

Some attributes do not apply to all data types. **AUTO\_INCREMENT** applies only to integer and floating-point types. Prior to MySQL 8.0.13, **DEFAULT** does not apply to the [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), **GEOMETRY**, and [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) types.

Character data types ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), the [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types, [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum), [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set), and any synonyms) can include **CHARACTER SET** to specify the character set for the column. **CHARSET** is a synonym for **CHARACTER SET**. A collation for the character set can be specified with the **COLLATE** attribute, along with any other attributes. For details, see [Chapter 10, *Character Sets, Collations, Unicode*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html). Example:

CREATE TABLE t (c CHAR(20) CHARACTER SET utf8 COLLATE utf8\_bin);

MySQL 8.0 interprets length specifications in character column definitions in characters. Lengths for [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) and [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) are in bytes.

For [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) columns, indexes can be created that use only the leading part of column values, using ***col\_name*(*length*)** syntax to specify an index prefix length. [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns also can be indexed, but a prefix length must be given. Prefix lengths are given in characters for nonbinary string types and in bytes for binary string types. That is, index entries consist of the first ***length*** characters of each column value for [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns, and the first ***length*** bytes of each column value for [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns. Indexing only a prefix of column values like this can make the index file much smaller. For additional information about index prefixes, see [Section 13.1.15, “CREATE INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index).

Only the **InnoDB** and **MyISAM** storage engines support indexing on [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns. For example:

CREATE TABLE test (blob\_col BLOB, INDEX(blob\_col(10)));

If a specified index prefix exceeds the maximum column data type size, [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) handles the index as follows:

For a nonunique index, either an error occurs (if strict SQL mode is enabled), or the index length is reduced to lie within the maximum column data type size and a warning is produced (if strict SQL mode is not enabled).

For a unique index, an error occurs regardless of SQL mode because reducing the index length might enable insertion of nonunique entries that do not meet the specified uniqueness requirement.

[**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) columns cannot be indexed. You can work around this restriction by creating an index on a generated column that extracts a scalar value from the **JSON** column. See [Indexing a Generated Column to Provide a JSON Column Index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#json-column-indirect-index), for a detailed example.

**NOT NULL | NULL**

If neither **NULL** nor **NOT NULL** is specified, the column is treated as though **NULL** had been specified.

In MySQL 8.0, only the **InnoDB**, **MyISAM**, and **MEMORY** storage engines support indexes on columns that can have **NULL** values. In other cases, you must declare indexed columns as **NOT NULL** or an error results.

**DEFAULT**

Specifies a default value for a column. For more information about default value handling, including the case that a column definition includes no explicit **DEFAULT** value, see [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults).

If the [**NO\_ZERO\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_zero_date) or [**NO\_ZERO\_IN\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_zero_in_date) SQL mode is enabled and a date-valued default is not correct according to that mode, [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) produces a warning if strict SQL mode is not enabled and an error if strict mode is enabled. For example, with [**NO\_ZERO\_IN\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_zero_in_date) enabled, **c1 DATE DEFAULT '2010-00-00'** produces a warning.

**VISIBLE**, **INVISIBLE**

Specify column visibility. The default is **VISIBLE** if neither keyword is present. A table must have at least one visible column. Attempting to make all columns invisible produces an error. For more information, see [Section 13.1.20.10, “Invisible Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#invisible-columns).

The **VISIBLE** and **INVISIBLE** keywords are available as of MySQL 8.0.23. Prior to MySQL 8.0.23, all columns are visible.

**AUTO\_INCREMENT**

An integer or floating-point column can have the additional attribute **AUTO\_INCREMENT**. When you insert a value of **NULL** (recommended) or **0** into an indexed **AUTO\_INCREMENT** column, the column is set to the next sequence value. Typically this is ***value*+1**, where ***value*** is the largest value for the column currently in the table. **AUTO\_INCREMENT** sequences begin with **1**.

To retrieve an **AUTO\_INCREMENT** value after inserting a row, use the [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) SQL function or the [**mysql\_insert\_id()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-insert-id.html) C API function. See [Section 12.16, “Information Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#information-functions), and [mysql\_insert\_id()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-insert-id.html).

If the [**NO\_AUTO\_VALUE\_ON\_ZERO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_auto_value_on_zero) SQL mode is enabled, you can store **0** in **AUTO\_INCREMENT** columns as **0** without generating a new sequence value. See [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

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**.

For **MyISAM** tables, you can specify an **AUTO\_INCREMENT** secondary column in a multiple-column key. See [Section 3.6.9, “Using AUTO\_INCREMENT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\tutorial.html#example-auto-increment).

To make MySQL compatible with some ODBC applications, you can find the **AUTO\_INCREMENT** value for the last inserted row with the following query:

SELECT \* FROM ***tbl\_name*** WHERE ***auto\_col*** IS NULL

This method requires that [**sql\_auto\_is\_null**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_auto_is_null) variable is not set to 0. See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

For information about **InnoDB** and **AUTO\_INCREMENT**, see [Section 15.6.1.6, “AUTO\_INCREMENT Handling in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-handling). For information about **AUTO\_INCREMENT** and MySQL Replication, see [Section 17.5.1.1, “Replication and AUTO\_INCREMENT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-auto-increment).

**COMMENT**

A comment for a column can be specified with the **COMMENT** option, up to 1024 characters long. The comment is displayed by the [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) and [**SHOW FULL COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns) statements.

**COLUMN\_FORMAT**

In NDB Cluster, it is also possible to specify a data storage format for individual columns of [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables using **COLUMN\_FORMAT**. Permissible column formats are **FIXED**, **DYNAMIC**, and **DEFAULT**. **FIXED** is used to specify fixed-width storage, **DYNAMIC** permits the column to be variable-width, and **DEFAULT** causes the column to use fixed-width or variable-width storage as determined by the column's data type (possibly overridden by a **ROW\_FORMAT** specifier).

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, the default value for **COLUMN\_FORMAT** is **FIXED**.

In NDB Cluster, the maximum possible offset for a column defined with **COLUMN\_FORMAT=FIXED** is 8188 bytes. For more information and possible workarounds, see [Section 23.1.7.5, “Limits Associated with Database Objects in NDB Cluster”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-limitations-database-objects).

**COLUMN\_FORMAT** currently has no effect on columns of tables using storage engines other than [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html). MySQL 8.0 silently ignores **COLUMN\_FORMAT**.

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** options (available as of MySQL 8.0.21) are used to specify column attributes for primary and secondary storage engines. The options are reserved for future use.

Permitted values are a string literal containing a valid **JSON** document or an empty string (''). Invalid **JSON** is rejected.

CREATE TABLE t1 (c1 INT ENGINE\_ATTRIBUTE='{"***key***":"***value***"}');

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values can be repeated without error. In this case, the last specified value is used.

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values are not checked by the server, nor are they cleared when the table's storage engine is changed.

**STORAGE**

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, it is possible to specify whether the column is stored on disk or in memory by using a **STORAGE** clause. **STORAGE DISK** causes the column to be stored on disk, and **STORAGE MEMORY** causes in-memory storage to be used. The [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement used must still include a **TABLESPACE** clause:

mysql> **CREATE TABLE t1 (**

-> **c1 INT STORAGE DISK,**

-> **c2 INT STORAGE MEMORY**

-> **) ENGINE NDB;**

ERROR 1005 (HY000): Can't create table 'c.t1' (errno: 140)

mysql> **CREATE TABLE t1 (**

-> **c1 INT STORAGE DISK,**

-> **c2 INT STORAGE MEMORY**

-> **) TABLESPACE ts\_1 ENGINE NDB;**

Query OK, 0 rows affected (1.06 sec)

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, **STORAGE DEFAULT** is equivalent to **STORAGE MEMORY**.

The **STORAGE** clause has no effect on tables using storage engines other than [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html). The **STORAGE** keyword is supported only in the build of [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) that is supplied with NDB Cluster; it is not recognized in any other version of MySQL, where any attempt to use the **STORAGE** keyword causes a syntax error.

**GENERATED ALWAYS**

Used to specify a generated column expression. For information about [generated columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_generated_column), see [Section 13.1.20.8, “CREATE TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-generated-columns).

[Stored generated columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_stored_generated_column) can be indexed. **InnoDB** supports secondary indexes on [virtual generated columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_virtual_generated_column). See [Section 13.1.20.9, “Secondary Indexes and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-secondary-indexes).

#### Indexes, Foreign Keys, and CHECK Constraints

Several keywords apply to creation of indexes, foreign keys, and **CHECK** constraints. For general background in addition to the following descriptions, see [Section 13.1.15, “CREATE INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index), [Section 13.1.20.5, “FOREIGN KEY Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys), and [Section 13.1.20.6, “CHECK Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-check-constraints).

**CONSTRAINT *symbol***

The **CONSTRAINT *symbol*** clause may be given to name a constraint. If the clause is not given, or a ***symbol*** is not included following the **CONSTRAINT** keyword, MySQL automatically generates a constraint name, with the exception noted below. The ***symbol*** value, if used, must be unique per schema (database), per constraint type. A duplicate ***symbol*** results in an error. See also the discussion about length limits of generated constraint identifiers at [Section 9.2.1, “Identifier Length Limits”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifier-length).

**Note**

If the **CONSTRAINT *symbol*** clause is not given in a foreign key definition, or a ***symbol*** is not included following the **CONSTRAINT** keyword, MySQL uses the foreign key index name up to MySQL 8.0.15, and automatically generates a constraint name thereafter.

The SQL standard specifies that all types of constraints (primary key, unique index, foreign key, check) belong to the same namespace. In MySQL, each constraint type has its own namespace per schema. Consequently, names for each type of constraint must be unique per schema, but constraints of different types can have the same name.

**PRIMARY KEY**

A unique index where all key columns must be defined as **NOT NULL**. If they are not explicitly declared as **NOT NULL**, MySQL declares them so implicitly (and silently). A table can have only one **PRIMARY KEY**. The name of a **PRIMARY KEY** is always **PRIMARY**, which thus cannot be used as the name for any other kind of index.

If you do not have a **PRIMARY KEY** and an application asks for the **PRIMARY KEY** in your tables, MySQL returns the first **UNIQUE** index that has no **NULL** columns as the **PRIMARY KEY**.

In **InnoDB** tables, keep the **PRIMARY KEY** short to minimize storage overhead for secondary indexes. Each secondary index entry contains a copy of the primary key columns for the corresponding row. (See [Section 15.6.2.1, “Clustered and Secondary Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-index-types).)

In the created table, a **PRIMARY KEY** is placed first, followed by all **UNIQUE** indexes, and then the nonunique indexes. This helps the MySQL optimizer to prioritize which index to use and also more quickly to detect duplicated **UNIQUE** keys.

A **PRIMARY KEY** can be a multiple-column index. However, you cannot create a multiple-column index using the **PRIMARY KEY** key attribute in a column specification. Doing so only marks that single column as primary. You must use a separate **PRIMARY KEY(*key\_part*, ...)** clause.

If a table has a **PRIMARY KEY** or **UNIQUE NOT NULL** index that consists of a single column that has an integer type, you can use **\_rowid** to refer to the indexed column in [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, as described in [Unique Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-unique).

In MySQL, the name of a **PRIMARY KEY** is **PRIMARY**. For other indexes, if you do not assign a name, the index is assigned the same name as the first indexed column, with an optional suffix (**\_2**, **\_3**, **...**) to make it unique. You can see index names for a table using **SHOW INDEX FROM *tbl\_name***. See [Section 13.7.7.22, “SHOW INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index).

**KEY | INDEX**

**KEY** is normally a synonym for **INDEX**. The key attribute **PRIMARY KEY** can also be specified as just **KEY** when given in a column definition. This was implemented for compatibility with other database systems.

**UNIQUE**

A **UNIQUE** index creates a constraint such that all values in the index must be distinct. An error occurs if you try to add a new row with a key value that matches an existing row. For all engines, a **UNIQUE** index permits multiple **NULL** values for columns that can contain **NULL**. If you specify a prefix value for a column in a **UNIQUE** index, the column values must be unique within the prefix length.

If a table has a **PRIMARY KEY** or **UNIQUE NOT NULL** index that consists of a single column that has an integer type, you can use **\_rowid** to refer to the indexed column in [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, as described in [Unique Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-unique).

**FULLTEXT**

A **FULLTEXT** index is a special type of index used for full-text searches. Only the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) storage engines support **FULLTEXT** indexes. They can be created only from [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns. Indexing always happens over the entire column; column prefix indexing is not supported and any prefix length is ignored if specified. See [Section 12.10, “Full-Text Search Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#fulltext-search), for details of operation. A **WITH PARSER** clause can be specified as an ***index\_option*** value to associate a parser plugin with the index if full-text indexing and searching operations need special handling. This clause is valid only for **FULLTEXT** indexes. [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) support full-text parser plugins. See [Full-Text Parser Plugins](https://dev.mysql.com/doc/extending-mysql/8.0/en/plugin-types.html#full-text-plugin-type) and [Writing Full-Text Parser Plugins](https://dev.mysql.com/doc/extending-mysql/8.0/en/writing-full-text-plugins.html) for more information.

**SPATIAL**

You can create **SPATIAL** indexes on spatial data types. Spatial types are supported only for **InnoDB** and **MyISAM** tables, and indexed columns must be declared as **NOT NULL**. See [Section 11.4, “Spatial Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-types).

**FOREIGN KEY**

MySQL supports foreign keys, which let you cross-reference related data across tables, and foreign key constraints, which help keep this spread-out data consistent. For definition and option information, see [***reference\_definition***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-reference-definition), and [***reference\_option***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-reference-option).

Partitioned tables employing the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) storage engine do not support foreign keys. See [Section 24.6, “Restrictions and Limitations on Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-limitations), for more information.

**CHECK**

The **CHECK** clause enables the creation of constraints to be checked for data values in table rows. See [Section 13.1.20.6, “CHECK Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-check-constraints).

***key\_part***

A ***key\_part*** specification can end with **ASC** or **DESC** to specify whether index values are stored in ascending or descending order. The default is ascending if no order specifier is given.

Prefixes, defined by the ***length*** attribute, can be up to 767 bytes long for **InnoDB** tables that use the [**REDUNDANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_redundant_row_format) or [**COMPACT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compact_row_format) row format. The prefix length limit is 3072 bytes for **InnoDB** tables that use the [**DYNAMIC**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dynamic_row_format) or [**COMPRESSED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compressed_row_format) row format. For **MyISAM** tables, the prefix length limit is 1000 bytes.

Prefix limits are measured in bytes. However, prefix lengths for index specifications in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), and [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statements are interpreted as number of characters for nonbinary string types ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)) and number of bytes for binary string types ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)). Take this into account when specifying a prefix length for a nonbinary string column that uses a multibyte character set.

Beginning with MySQL 8.0.17, the ***expr*** for a ***key\_part*** specification can take the form **(CAST *json\_path* AS *type* ARRAY)** to create a multi-valued index on a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column. [Multi-Valued Indexes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-multi-valued), provides detailed information regarding creation of, usage of, and restrictions and limitations on multi-valued indexes.

***index\_type***

Some storage engines permit you to specify an index type when creating an index. The syntax for the ***index\_type*** specifier is **USING *type\_name***.

Example:

CREATE TABLE lookup

(id INT, INDEX USING BTREE (id))

ENGINE = MEMORY;

The preferred position for **USING** is after the index column list. It can be given before the column list, but support for use of the option in that position is deprecated and you should expect it to be removed in a future MySQL release.

***index\_option***

***index\_option*** values specify additional options for an index.

**KEY\_BLOCK\_SIZE**

For [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables, **KEY\_BLOCK\_SIZE** optionally specifies the size in bytes to use for index key blocks. The value is treated as a hint; a different size could be used if necessary. A **KEY\_BLOCK\_SIZE** value specified for an individual index definition overrides the table-level **KEY\_BLOCK\_SIZE** value.

For information about the table-level **KEY\_BLOCK\_SIZE** attribute, see [Table Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-options).

**WITH PARSER**

The **WITH PARSER** option can be used only with **FULLTEXT** indexes. It associates a parser plugin with the index if full-text indexing and searching operations need special handling. [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) support full-text parser plugins. If you have a [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) table with an associated full-text parser plugin, you can convert the table to **InnoDB** using **ALTER TABLE**.

**COMMENT**

Index definitions can include an optional comment of up to 1024 characters.

You can set the **InnoDB** **MERGE\_THRESHOLD** value for an individual index using the ***index\_option*** **COMMENT** clause. See [Section 15.8.11, “Configuring the Merge Threshold for Index Pages”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#index-page-merge-threshold).

**VISIBLE**, **INVISIBLE**

Specify index visibility. Indexes are visible by default. An invisible index is not used by the optimizer. Specification of index visibility applies to indexes other than primary keys (either explicit or implicit). For more information, see [Section 8.3.12, “Invisible Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#invisible-indexes).

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** options (available as of MySQL 8.0.21) are used to specify index attributes for primary and secondary storage engines. The options are reserved for future use.

For more information about permissible ***index\_option*** values, see [Section 13.1.15, “CREATE INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index). For more information about indexes, see [Section 8.3.1, “How MySQL Uses Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#mysql-indexes).

***reference\_definition***

For ***reference\_definition*** syntax details and examples, see [Section 13.1.20.5, “FOREIGN KEY Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys).

[**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables support checking of foreign key constraints. The columns of the referenced table must always be explicitly named. Both **ON DELETE** and **ON UPDATE** actions on foreign keys are supported. For more detailed information and examples, see [Section 13.1.20.5, “FOREIGN KEY Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys).

For other storage engines, MySQL Server parses and ignores the **FOREIGN KEY** and **REFERENCES** syntax in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements. See [Section 1.7.2.3, “FOREIGN KEY Constraint Differences”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\introduction.html#ansi-diff-foreign-keys).

**Important**

For users familiar with the ANSI/ISO SQL Standard, please note that no storage engine, including **InnoDB**, recognizes or enforces the **MATCH** clause used in referential integrity constraint definitions. Use of an explicit **MATCH** clause does not have the specified effect, and also causes **ON DELETE** and **ON UPDATE** clauses to be ignored. For these reasons, specifying **MATCH** should be avoided.

The **MATCH** clause in the SQL standard controls how **NULL** values in a composite (multiple-column) foreign key are handled when comparing to a primary key. **InnoDB** essentially implements the semantics defined by **MATCH SIMPLE**, which permit a foreign key to be all or partially **NULL**. In that case, the (child table) row containing such a foreign key is permitted to be inserted, and does not match any row in the referenced (parent) table. It is possible to implement other semantics using triggers.

Additionally, MySQL requires that the referenced columns be indexed for performance. However, **InnoDB** does not enforce any requirement that the referenced columns be declared **UNIQUE** or **NOT NULL**. The handling of foreign key references to nonunique keys or keys that contain **NULL** values is not well defined for operations such as **UPDATE** or **DELETE CASCADE**. You are advised to use foreign keys that reference only keys that are both **UNIQUE** (or **PRIMARY**) and **NOT NULL**.

MySQL parses but ignores “inline **REFERENCES** specifications” (as defined in the SQL standard) where the references are defined as part of the column specification. MySQL accepts **REFERENCES** clauses only when specified as part of a separate **FOREIGN KEY** specification.

***reference\_option***

For information about the **RESTRICT**, **CASCADE**, **SET NULL**, **NO ACTION**, and **SET DEFAULT** options, see [Section 13.1.20.5, “FOREIGN KEY Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys).

#### Table Options

Table options are used to optimize the behavior of the table. In most cases, you do not have to specify any of them. These options apply to all storage engines unless otherwise indicated. Options that do not apply to a given storage engine may be accepted and remembered as part of the table definition. Such options then apply if you later use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to convert the table to use a different storage engine.

**ENGINE**

Specifies the storage engine for the table, using one of the names shown in the following table. The engine name can be unquoted or quoted. The quoted name **'DEFAULT'** is recognized but ignored.

| **Storage Engine** | **Description** |
| --- | --- |
| **InnoDB** | Transaction-safe tables with row locking and foreign keys. The default storage engine for new tables. See [Chapter 15, *The InnoDB Storage Engine*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), and in particular [Section 15.1, “Introduction to InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-introduction) if you have MySQL experience but are new to **InnoDB**. |
| **MyISAM** | The binary portable storage engine that is primarily used for read-only or read-mostly workloads. See [Section 16.2, “The MyISAM Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine). |
| **MEMORY** | The data for this storage engine is stored only in memory. See [Section 16.3, “The MEMORY Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine). |
| **CSV** | Tables that store rows in comma-separated values format. See [Section 16.4, “The CSV Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#csv-storage-engine). |
| **ARCHIVE** | The archiving storage engine. See [Section 16.5, “The ARCHIVE Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#archive-storage-engine). |
| **EXAMPLE** | An example engine. See [Section 16.9, “The EXAMPLE Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#example-storage-engine). |
| **FEDERATED** | Storage engine that accesses remote tables. See [Section 16.8, “The FEDERATED Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#federated-storage-engine). |
| **HEAP** | This is a synonym for **MEMORY**. |
| **MERGE** | A collection of **MyISAM** tables used as one table. Also known as **MRG\_MyISAM**. See [Section 16.7, “The MERGE Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#merge-storage-engine). |
| [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) | Clustered, fault-tolerant, memory-based tables, supporting transactions and foreign keys. Also known as [**NDBCLUSTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html). See [Chapter 23, *MySQL NDB Cluster 8.0*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html). |

By default, if a storage engine is specified that is not available, the statement fails with an error. You can override this behavior by removing [**NO\_ENGINE\_SUBSTITUTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_engine_substitution) from the server SQL mode (see [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode)) so that MySQL allows substitution of the specified engine with the default storage engine instead. Normally in such cases, this is **InnoDB**, which is the default value for the [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) system variable. When **NO\_ENGINE\_SUBSTITUTION** is disabled, a warning occurs if the storage engine specification is not honored.

**AUTOEXTEND\_SIZE**

Defines the amount by which **InnoDB** extends the size of the tablespace when it becomes full. Introduced in MySQL 8.0.23. The setting must be a multiple of 4MB. The default setting is 0, which causes the tablespace to be extended according to the implicit default behavior. For more information, see [Section 15.6.3.9, “Tablespace AUTOEXTEND\_SIZE Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tablespace-autoextend-size).

**AUTO\_INCREMENT**

The initial **AUTO\_INCREMENT** value for the table. In MySQL 8.0, this works for **MyISAM**, **MEMORY**, **InnoDB**, and **ARCHIVE** tables. To set the first auto-increment value for engines that do not support the **AUTO\_INCREMENT** table option, insert a “dummy” row with a value one less than the desired value after creating the table, and then delete the dummy row.

For engines that support the **AUTO\_INCREMENT** table option in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements, you can also use **ALTER TABLE *tbl\_name* AUTO\_INCREMENT = *N*** to reset the **AUTO\_INCREMENT** value. The value cannot be set lower than the maximum value currently in the column.

**AVG\_ROW\_LENGTH**

An approximation of the average row length for your table. You need to set this only for large tables with variable-size rows.

When you create a **MyISAM** table, MySQL uses the product of the **MAX\_ROWS** and **AVG\_ROW\_LENGTH** options to decide how big the resulting table is. If you don't specify either option, the maximum size for **MyISAM** data and index files is 256TB by default. (If your operating system does not support files that large, table sizes are constrained by the file size limit.) If you want to keep down the pointer sizes to make the index smaller and faster and you don't really need big files, you can decrease the default pointer size by setting the [**myisam\_data\_pointer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_myisam_data_pointer_size) system variable. (See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).) If you want all your tables to be able to grow above the default limit and are willing to have your tables slightly slower and larger than necessary, you can increase the default pointer size by setting this variable. Setting the value to 7 permits table sizes up to 65,536TB.

**[DEFAULT] CHARACTER SET**

Specifies a default character set for the table. **CHARSET** is a synonym for **CHARACTER SET**. If the character set name is **DEFAULT**, the database character set is used.

**CHECKSUM**

Set this to 1 if you want MySQL to maintain a live checksum for all rows (that is, a checksum that MySQL updates automatically as the table changes). This makes the table a little slower to update, but also makes it easier to find corrupted tables. The [**CHECKSUM TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#checksum-table) statement reports the checksum. (**MyISAM** only.)

**[DEFAULT] COLLATE**

Specifies a default collation for the table.

**COMMENT**

A comment for the table, up to 2048 characters long.

You can set the **InnoDB** **MERGE\_THRESHOLD** value for a table using the ***table\_option*** **COMMENT** clause. See [Section 15.8.11, “Configuring the Merge Threshold for Index Pages”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#index-page-merge-threshold).

**Setting NDB\_TABLE options.** The table comment in a **CREATE TABLE** that creates an **NDB** table or an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement which alters one can also be used to specify one to four of the **NDB\_TABLE** options **NOLOGGING**, **READ\_BACKUP**, **PARTITION\_BALANCE**, or **FULLY\_REPLICATED** as a set of name-value pairs, separated by commas if need be, immediately following the string **NDB\_TABLE=** that begins the quoted comment text. An example statement using this syntax is shown here (emphasized text):

CREATE TABLE t1 (

c1 INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

c2 VARCHAR(100),

c3 VARCHAR(100) )

ENGINE=NDB

COMMENT="NDB\_TABLE=READ\_BACKUP=0,PARTITION\_BALANCE=FOR\_RP\_BY\_NODE";

Spaces are not permitted within the quoted string. The string is case-insensitive.

The comment is displayed as part of the ouput of [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table). The text of the comment is also available as the TABLE\_COMMENT column of the MySQL Information Schema [**TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table) table.

This comment syntax is also supported with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements for **NDB** tables. Keep in mind that a table comment used with **ALTER TABLE** replaces any existing comment which the table might have had perviously.

Setting the **MERGE\_THRESHOLD** option in table comments is not supported for [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables (it is ignored).

For complete syntax information and examples, see [Section 13.1.20.11, “Setting NDB\_TABLE Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-ndb-table-comment-options).

**COMPRESSION**

The compression algorithm used for page level compression for **InnoDB** tables. Supported values include **Zlib**, **LZ4**, and **None**. The **COMPRESSION** attribute was introduced with the transparent page compression feature. Page compression is only supported with **InnoDB** tables that reside in [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) tablespaces, and is only available on Linux and Windows platforms that support sparse files and hole punching. For more information, see [Section 15.9.2, “InnoDB Page Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-page-compression).

**CONNECTION**

The connection string for a **FEDERATED** table.

**Note**

Older versions of MySQL used a **COMMENT** option for the connection string.

**DATA DIRECTORY**, **INDEX DIRECTORY**

For **InnoDB**, the **DATA DIRECTORY='*directory*'** clause permits creating tables outside of the data directory. The [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) variable must be enabled to use the **DATA DIRECTORY** clause. The full directory path must be specified. As of MySQL 8.0.21, the directory specified must be known to **InnoDB**. For more information, see [Section 15.6.1.2, “Creating Tables Externally”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external).

When creating **MyISAM** tables, you can use the **DATA DIRECTORY='*directory*'** clause, the **INDEX DIRECTORY='*directory*'** clause, or both. They specify where to put a **MyISAM** table's data file and index file, respectively. Unlike **InnoDB** tables, MySQL does not create subdirectories that correspond to the database name when creating a **MyISAM** table with a **DATA DIRECTORY** or **INDEX DIRECTORY** option. Files are created in the directory that is specified.

You must have the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege to use the **DATA DIRECTORY** or **INDEX DIRECTORY** table option.

**Important**

Table-level **DATA DIRECTORY** and **INDEX DIRECTORY** options are ignored for partitioned tables. (Bug #32091)

These options work only when you are not using the [--skip-symbolic-links](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_symbolic-links) option. Your operating system must also have a working, thread-safe **realpath()** call. See [Section 8.12.2.2, “Using Symbolic Links for MyISAM Tables on Unix”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#symbolic-links-to-tables), for more complete information.

If a **MyISAM** table is created with no **DATA DIRECTORY** option, the .MYD file is created in the database directory. By default, if **MyISAM** finds an existing .MYD file in this case, it overwrites it. The same applies to .MYI files for tables created with no **INDEX DIRECTORY** option. To suppress this behavior, start the server with the [--keep\_files\_on\_create](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_keep_files_on_create) option, in which case **MyISAM** does not overwrite existing files and returns an error instead.

If a **MyISAM** table is created with a **DATA DIRECTORY** or **INDEX DIRECTORY** option and an existing .MYD or .MYI file is found, **MyISAM** always returns an error, and does not overwrite a file in the specified directory.

**Important**

You cannot use path names that contain the MySQL data directory with **DATA DIRECTORY** or **INDEX DIRECTORY**. This includes partitioned tables and individual table partitions. (See Bug #32167.)

**DELAY\_KEY\_WRITE**

Set this to 1 if you want to delay key updates for the table until the table is closed. See the description of the [**delay\_key\_write**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_delay_key_write) system variable in [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables). (**MyISAM** only.)

**ENCRYPTION**

The **ENCRYPTION** clause enables or disables page-level data encryption for an **InnoDB** table. A keyring plugin must be installed and configured before encryption can be enabled. Prior to MySQL 8.0.16, the **ENCRYPTION** clause can only be specified when creating a table in an a file-per-table tablespace. As of MySQL 8.0.16, the **ENCRYPTION** clause can also be specified when creating a table in a general tablespace.

As of MySQL 8.0.16, a table inherits the default schema encryption if an **ENCRYPTION** clause is not specified. If the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) variable is enabled, the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege is required to create a table with an **ENCRYPTION** clause setting that differs from the default schema encryption. When creating a table in a general tablespace, table and tablespace encryption must match.

As of MySQL 8.0.16, specifying an **ENCRYPTION** clause with a value other than **'N'** or **''** is not permitted when using a storage engine that does not support encryption. Previously, the clause was accepted.

For more information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption).

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** options (available as of MySQL 8.0.21) are used to specify table attributes for primary and secondary storage engines. The options are reserved for future use.

Permitted values are a string literal containing a valid **JSON** document or an empty string (''). Invalid **JSON** is rejected.

CREATE TABLE t1 (c1 INT) ENGINE\_ATTRIBUTE='{"***key***":"***value***"}';

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values can be repeated without error. In this case, the last specified value is used.

**ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values are not checked by the server, nor are they cleared when the table's storage engine is changed.

**INSERT\_METHOD**

If you want to insert data into a **MERGE** table, you must specify with **INSERT\_METHOD** the table into which the row should be inserted. **INSERT\_METHOD** is an option useful for **MERGE** tables only. Use a value of **FIRST** or **LAST** to have inserts go to the first or last table, or a value of **NO** to prevent inserts. See [Section 16.7, “The MERGE Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#merge-storage-engine).

**KEY\_BLOCK\_SIZE**

For [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables, **KEY\_BLOCK\_SIZE** optionally specifies the size in bytes to use for index key blocks. The value is treated as a hint; a different size could be used if necessary. A **KEY\_BLOCK\_SIZE** value specified for an individual index definition overrides the table-level **KEY\_BLOCK\_SIZE** value.

For [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, **KEY\_BLOCK\_SIZE** specifies the [page](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) size in kilobytes to use for [compressed](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_compression) **InnoDB** tables. The **KEY\_BLOCK\_SIZE** value is treated as a hint; a different size could be used by **InnoDB** if necessary. **KEY\_BLOCK\_SIZE** can only be less than or equal to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. A value of 0 represents the default compressed page size, which is half of the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. Depending on [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size), possible **KEY\_BLOCK\_SIZE** values include 0, 1, 2, 4, 8, and 16. See [Section 15.9.1, “InnoDB Table Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-compression) for more information.

Oracle recommends enabling [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) when specifying **KEY\_BLOCK\_SIZE** for **InnoDB** tables. When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled, specifying an invalid **KEY\_BLOCK\_SIZE** value returns an error. If [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is disabled, an invalid **KEY\_BLOCK\_SIZE** value results in a warning, and the **KEY\_BLOCK\_SIZE** option is ignored.

The **Create\_options** column in response to [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) reports the actual **KEY\_BLOCK\_SIZE** used by the table, as does [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table).

**InnoDB** only supports **KEY\_BLOCK\_SIZE** at the table level.

**KEY\_BLOCK\_SIZE** is not supported with 32KB and 64KB [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) values. **InnoDB** table compression does not support these pages sizes.

**InnoDB** does not support the **KEY\_BLOCK\_SIZE** option when creating temporary tables.

**MAX\_ROWS**

The maximum number of rows you plan to store in the table. This is not a hard limit, but rather a hint to the storage engine that the table must be able to store at least this many rows.

**Important**

The use of **MAX\_ROWS** with **NDB** tables to control the number of table partitions is deprecated. It remains supported in later versions for backward compatibility, but is subject to removal in a future release. Use PARTITION\_BALANCE instead; see [Setting NDB\_TABLE options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-comment-ndb-table-options).

The [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine treats this value as a maximum. If you plan to create very large NDB Cluster tables (containing millions of rows), you should use this option to insure that [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) allocates sufficient number of index slots in the hash table used for storing hashes of the table's primary keys by setting **MAX\_ROWS = 2 \* *rows***, where ***rows*** is the number of rows that you expect to insert into the table.

The maximum **MAX\_ROWS** value is 4294967295; larger values are truncated to this limit.

**MIN\_ROWS**

The minimum number of rows you plan to store in the table. The [**MEMORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#memory-storage-engine) storage engine uses this option as a hint about memory use.

**PACK\_KEYS**

Takes effect only with **MyISAM** tables. Set this option to 1 if you want to have smaller indexes. This usually makes updates slower and reads faster. Setting the option to 0 disables all packing of keys. Setting it to **DEFAULT** tells the storage engine to pack only long [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), or [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) columns.

If you do not use **PACK\_KEYS**, the default is to pack strings, but not numbers. If you use **PACK\_KEYS=1**, numbers are packed as well.

When packing binary number keys, MySQL uses prefix compression:

Every key needs one extra byte to indicate how many bytes of the previous key are the same for the next key.

The pointer to the row is stored in high-byte-first order directly after the key, to improve compression.

This means that if you have many equal keys on two consecutive rows, all following “same” keys usually only take two bytes (including the pointer to the row). Compare this to the ordinary case where the following keys takes **storage\_size\_for\_key + pointer\_size** (where the pointer size is usually 4). Conversely, you get a significant benefit from prefix compression only if you have many numbers that are the same. If all keys are totally different, you use one byte more per key, if the key is not a key that can have **NULL** values. (In this case, the packed key length is stored in the same byte that is used to mark if a key is **NULL**.)

**PASSWORD**

This option is unused.

**ROW\_FORMAT**

Defines the physical format in which the rows are stored.

When creating a table with [strict mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_strict_mode) disabled, the storage engine's default row format is used if the specified row format is not supported. The actual row format of the table is reported in the **Row\_format** column in response to [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status). The **Create\_options** column shows the row format that was specified in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, as does [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table).

Row format choices differ depending on the storage engine used for the table.

For **InnoDB** tables:

The default row format is defined by [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format), which has a default setting of **DYNAMIC**. The default row format is used when the **ROW\_FORMAT** option is not defined or when **ROW\_FORMAT=DEFAULT** is used.

If the **ROW\_FORMAT** option is not defined, or if **ROW\_FORMAT=DEFAULT** is used, operations that rebuild a table also silently change the row format of the table to the default defined by [**innodb\_default\_row\_format**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_default_row_format). For more information, see [Defining the Row Format of a Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-defining).

For more efficient **InnoDB** storage of data types, especially [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types, use the **DYNAMIC**. See [DYNAMIC Row Format](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format-dynamic) for requirements associated with the **DYNAMIC** row format.

To enable compression for **InnoDB** tables, specify **ROW\_FORMAT=COMPRESSED**. The **ROW\_FORMAT=COMPRESSED** option is not supported when creating temporary tables. See [Section 15.9, “InnoDB Table and Page Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression) for requirements associated with the **COMPRESSED** row format.

The row format used in older versions of MySQL can still be requested by specifying the **REDUNDANT** row format.

When you specify a non-default **ROW\_FORMAT** clause, consider also enabling the [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) configuration option.

**ROW\_FORMAT=FIXED** is not supported. If **ROW\_FORMAT=FIXED** is specified while [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is disabled, **InnoDB** issues a warning and assumes **ROW\_FORMAT=DYNAMIC**. If **ROW\_FORMAT=FIXED** is specified while [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled, which is the default, **InnoDB** returns an error.

For additional information about **InnoDB** row formats, see [Section 15.10, “InnoDB Row Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format).

For **MyISAM** tables, the option value can be **FIXED** or **DYNAMIC** for static or variable-length row format. [**myisampack**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack) sets the type to **COMPRESSED**. See [Section 16.2.3, “MyISAM Table Storage Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-table-formats).

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, the default **ROW\_FORMAT** is **DYNAMIC**.

**STATS\_AUTO\_RECALC**

Specifies whether to automatically recalculate [persistent statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_persistent_statistics) for an **InnoDB** table. The value **DEFAULT** causes the persistent statistics setting for the table to be determined by the [**innodb\_stats\_auto\_recalc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_auto_recalc) configuration option. The value **1** causes statistics to be recalculated when 10% of the data in the table has changed. The value **0** prevents automatic recalculation for this table; with this setting, issue an [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) statement to recalculate the statistics after making substantial changes to the table. For more information about the persistent statistics feature, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

**STATS\_PERSISTENT**

Specifies whether to enable [persistent statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_persistent_statistics) for an **InnoDB** table. The value **DEFAULT** causes the persistent statistics setting for the table to be determined by the [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) configuration option. The value **1** enables persistent statistics for the table, while the value **0** turns off this feature. After enabling persistent statistics through a **CREATE TABLE** or **ALTER TABLE** statement, issue an [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) statement to calculate the statistics, after loading representative data into the table. For more information about the persistent statistics feature, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

**STATS\_SAMPLE\_PAGES**

The number of index pages to sample when estimating cardinality and other statistics for an indexed column, such as those calculated by [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table). For more information, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats).

**TABLESPACE**

The **TABLESPACE** clause can be used to create a table in an existing general tablespace, a file-per-table tablespace, or the system tablespace.

CREATE TABLE ***tbl\_name*** ... TABLESPACE [=] ***tablespace\_name***

The general tablespace that you specify must exist prior to using the **TABLESPACE** clause. For information about general tablespaces, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

The ***tablespace\_name*** is a case-sensitive identifier. It may be quoted or unquoted. The forward slash character (“/”) is not permitted. Names beginning with “innodb\_” are reserved for special use.

To create a table in the system tablespace, specify **innodb\_system** as the tablespace name.

CREATE TABLE ***tbl\_name*** ... TABLESPACE [=] innodb\_system

Using **TABLESPACE [=] innodb\_system**, you can place a table of any uncompressed row format in the system tablespace regardless of the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) setting. For example, you can add a table with **ROW\_FORMAT=DYNAMIC** to the system tablespace using **TABLESPACE [=] innodb\_system**.

To create a table in a file-per-table tablespace, specify **innodb\_file\_per\_table** as the tablespace name.

CREATE TABLE ***tbl\_name*** ... TABLESPACE [=] innodb\_file\_per\_table

**Note**

If [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) is enabled, you need not specify **TABLESPACE=innodb\_file\_per\_table** to create an **InnoDB** file-per-table tablespace. **InnoDB** tables are created in file-per-table tablespaces by default when [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) is enabled.

The **DATA DIRECTORY** clause is permitted with **CREATE TABLE ... TABLESPACE=innodb\_file\_per\_table** but is otherwise not supported for use in combination with the **TABLESPACE** clause. As of MySQL 8.0.21, the directory specified in a **DATA DIRECTORY** clause must be known to **InnoDB**. For more information, see [Using the DATA DIRECTORY Clause](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external-data-directory).

**Note**

Support for **TABLESPACE = innodb\_file\_per\_table** and **TABLESPACE = innodb\_temporary** clauses with [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) is deprecated as of MySQL 8.0.13; expect it to be removed in a future version of MySQL.

The **STORAGE** table option is employed only with [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables. **STORAGE** determines the type of storage used (disk or memory), and can be either **DISK** or **MEMORY**.

**TABLESPACE ... STORAGE DISK** assigns a table to an NDB Cluster Disk Data tablespace. The tablespace must already have been created using [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace). See [Section 23.5.10, “NDB Cluster Disk Data Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data), for more information.

**Important**

A **STORAGE** clause cannot be used in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement without a **TABLESPACE** clause.

[**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union)

Used to access a collection of identical **MyISAM** tables as one. This works only with **MERGE** tables. See [Section 16.7, “The MERGE Storage Engine”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#merge-storage-engine).

You must have [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privileges for the tables you map to a **MERGE** table.

**Note**

Formerly, all tables used had to be in the same database as the **MERGE** table itself. This restriction no longer applies.

#### Table Partitioning

***partition\_options*** can be used to control partitioning of the table created with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table).

Not all options shown in the syntax for ***partition\_options*** at the beginning of this section are available for all partitioning types. Please see the listings for the following individual types for information specific to each type, and see [Chapter 24, *Partitioning*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html), for more complete information about the workings of and uses for partitioning in MySQL, as well as additional examples of table creation and other statements relating to MySQL partitioning.

Partitions can be modified, merged, added to tables, and dropped from tables. For basic information about the MySQL statements to accomplish these tasks, see [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). For more detailed descriptions and examples, see [Section 24.3, “Partition Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-management).

**PARTITION BY**

If used, a ***partition\_options*** clause begins with **PARTITION BY**. This clause contains the function that is used to determine the partition; the function returns an integer value ranging from 1 to ***num***, where ***num*** is the number of partitions. (The maximum number of user-defined partitions which a table may contain is 1024; the number of subpartitions—discussed later in this section—is included in this maximum.)

**Note**

The expression (***expr***) used in a **PARTITION BY** clause cannot refer to any columns not in the table being created; such references are specifically not permitted and cause the statement to fail with an error. (Bug #29444)

**HASH(*expr*)**

Hashes one or more columns to create a key for placing and locating rows. ***expr*** is an expression using one or more table columns. This can be any valid MySQL expression (including MySQL functions) that yields a single integer value. For example, these are both valid [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements using **PARTITION BY HASH**:

CREATE TABLE t1 (col1 INT, col2 CHAR(5))

PARTITION BY HASH(col1);

CREATE TABLE t1 (col1 INT, col2 CHAR(5), col3 DATETIME)

PARTITION BY HASH ( YEAR(col3) );

You may not use either **VALUES LESS THAN** or **VALUES IN** clauses with **PARTITION BY HASH**.

**PARTITION BY HASH** uses the remainder of ***expr*** divided by the number of partitions (that is, the modulus). For examples and additional information, see [Section 24.2.4, “HASH Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-hash).

The **LINEAR** keyword entails a somewhat different algorithm. In this case, the number of the partition in which a row is stored is calculated as the result of one or more logical [**AND**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and) operations. For discussion and examples of linear hashing, see [Section 24.2.4.1, “LINEAR HASH Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-linear-hash).

**KEY(*column\_list*)**

This is similar to **HASH**, except that MySQL supplies the hashing function so as to guarantee an even data distribution. The ***column\_list*** argument is simply a list of 1 or more table columns (maximum: 16). This example shows a simple table partitioned by key, with 4 partitions:

CREATE TABLE tk (col1 INT, col2 CHAR(5), col3 DATE)

PARTITION BY KEY(col3)

PARTITIONS 4;

For tables that are partitioned by key, you can employ linear partitioning by using the **LINEAR** keyword. This has the same effect as with tables that are partitioned by **HASH**. That is, the partition number is found using the [**&**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_bitwise-and) operator rather than the modulus (see [Section 24.2.4.1, “LINEAR HASH Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-linear-hash), and [Section 24.2.5, “KEY Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-key), for details). This example uses linear partitioning by key to distribute data between 5 partitions:

CREATE TABLE tk (col1 INT, col2 CHAR(5), col3 DATE)

PARTITION BY LINEAR KEY(col3)

PARTITIONS 5;

The **ALGORITHM={1 | 2}** option is supported with **[SUB]PARTITION BY [LINEAR] KEY**. **ALGORITHM=1** causes the server to use the same key-hashing functions as MySQL 5.1; **ALGORITHM=2** means that the server employs the key-hashing functions implemented and used by default for new **KEY** partitioned tables in MySQL 5.5 and later. (Partitioned tables created with the key-hashing functions employed in MySQL 5.5 and later cannot be used by a MySQL 5.1 server.) Not specifying the option has the same effect as using **ALGORITHM=2**. This option is intended for use chiefly when upgrading or downgrading **[LINEAR] KEY** partitioned tables between MySQL 5.1 and later MySQL versions, or for creating tables partitioned by **KEY** or **LINEAR KEY** on a MySQL 5.5 or later server which can be used on a MySQL 5.1 server. For more information, see [Section 13.1.9.1, “ALTER TABLE Partition Operations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table-partition-operations).

[**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) in MySQL 5.7 (and later) writes this option encased in versioned comments, like this:

CREATE TABLE t1 (a INT)

/\*!50100 PARTITION BY KEY \*/ /\*!50611 ALGORITHM = 1 \*/ /\*!50100 ()

PARTITIONS 3 \*/

This causes MySQL 5.6.10 and earlier servers to ignore the option, which would otherwise cause a syntax error in those versions. If you plan to load a dump made on a MySQL 5.7 server where you use tables that are partitioned or subpartitioned by **KEY** into a MySQL 5.6 server previous to version 5.6.11, be sure to consult [Changes in MySQL 5.6](https://dev.mysql.com/doc/refman/5.6/en/upgrading-from-previous-series.html), before proceeding. (The information found there also applies if you are loading a dump containing **KEY** partitioned or subpartitioned tables made from a MySQL 5.7—actually 5.6.11 or later—server into a MySQL 5.5.30 or earlier server.)

Also in MySQL 5.6.11 and later, **ALGORITHM=1** is shown when necessary in the output of [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) using versioned comments in the same manner as [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump). **ALGORITHM=2** is always omitted from **SHOW CREATE TABLE** output, even if this option was specified when creating the original table.

You may not use either **VALUES LESS THAN** or **VALUES IN** clauses with **PARTITION BY KEY**.

**RANGE(*expr*)**

In this case, ***expr*** shows a range of values using a set of **VALUES LESS THAN** operators. When using range partitioning, you must define at least one partition using **VALUES LESS THAN**. You cannot use **VALUES IN** with range partitioning.

**Note**

For tables partitioned by **RANGE**, **VALUES LESS THAN** must be used with either an integer literal value or an expression that evaluates to a single integer value. In MySQL 8.0, you can overcome this limitation in a table that is defined using **PARTITION BY RANGE COLUMNS**, as described later in this section.

Suppose that you have a table that you wish to partition on a column containing year values, according to the following scheme.

| **Partition Number:** | **Years Range:** |
| --- | --- |
| 0 | 1990 and earlier |
| 1 | 1991 to 1994 |
| 2 | 1995 to 1998 |
| 3 | 1999 to 2002 |
| 4 | 2003 to 2005 |
| 5 | 2006 and later |

A table implementing such a partitioning scheme can be realized by the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement shown here:

CREATE TABLE t1 (

year\_col INT,

some\_data INT

)

PARTITION BY RANGE (year\_col) (

PARTITION p0 VALUES LESS THAN (1991),

PARTITION p1 VALUES LESS THAN (1995),

PARTITION p2 VALUES LESS THAN (1999),

PARTITION p3 VALUES LESS THAN (2002),

PARTITION p4 VALUES LESS THAN (2006),

PARTITION p5 VALUES LESS THAN MAXVALUE

);

**PARTITION ... VALUES LESS THAN ...** statements work in a consecutive fashion. **VALUES LESS THAN MAXVALUE** works to specify “leftover” values that are greater than the maximum value otherwise specified.

**VALUES LESS THAN** clauses work sequentially in a manner similar to that of the **case** portions of a **switch ... case** block (as found in many programming languages such as C, Java, and PHP). That is, the clauses must be arranged in such a way that the upper limit specified in each successive **VALUES LESS THAN** is greater than that of the previous one, with the one referencing **MAXVALUE** coming last of all in the list.

**RANGE COLUMNS(*column\_list*)**

This variant on **RANGE** facilitates partition pruning for queries using range conditions on multiple columns (that is, having conditions such as **WHERE a = 1 AND b < 10** or **WHERE a = 1 AND b = 10 AND c < 10**). It enables you to specify value ranges in multiple columns by using a list of columns in the **COLUMNS** clause and a set of column values in each **PARTITION ... VALUES LESS THAN (*value\_list*)** partition definition clause. (In the simplest case, this set consists of a single column.) The maximum number of columns that can be referenced in the ***column\_list*** and ***value\_list*** is 16.

The ***column\_list*** used in the **COLUMNS** clause may contain only names of columns; each column in the list must be one of the following MySQL data types: the integer types; the string types; and time or date column types. Columns using **BLOB**, **TEXT**, **SET**, **ENUM**, **BIT**, or spatial data types are not permitted; columns that use floating-point number types are also not permitted. You also may not use functions or arithmetic expressions in the **COLUMNS** clause.

The **VALUES LESS THAN** clause used in a partition definition must specify a literal value for each column that appears in the **COLUMNS()** clause; that is, the list of values used for each **VALUES LESS THAN** clause must contain the same number of values as there are columns listed in the **COLUMNS** clause. An attempt to use more or fewer values in a **VALUES LESS THAN** clause than there are in the **COLUMNS** clause causes the statement to fail with the error Inconsistency in usage of column lists for partitioning.... You cannot use **NULL** for any value appearing in **VALUES LESS THAN**. It is possible to use **MAXVALUE** more than once for a given column other than the first, as shown in this example:

CREATE TABLE rc (

a INT NOT NULL,

b INT NOT NULL

)

PARTITION BY RANGE COLUMNS(a,b) (

PARTITION p0 VALUES LESS THAN (10,5),

PARTITION p1 VALUES LESS THAN (20,10),

PARTITION p2 VALUES LESS THAN (50,MAXVALUE),

PARTITION p3 VALUES LESS THAN (65,MAXVALUE),

PARTITION p4 VALUES LESS THAN (MAXVALUE,MAXVALUE)

);

Each value used in a **VALUES LESS THAN** value list must match the type of the corresponding column exactly; no conversion is made. For example, you cannot use the string **'1'** for a value that matches a column that uses an integer type (you must use the numeral **1** instead), nor can you use the numeral **1** for a value that matches a column that uses a string type (in such a case, you must use a quoted string: **'1'**).

For more information, see [Section 24.2.1, “RANGE Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-range), and [Section 24.4, “Partition Pruning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-pruning).

**LIST(*expr*)**

This is useful when assigning partitions based on a table column with a restricted set of possible values, such as a state or country code. In such a case, all rows pertaining to a certain state or country can be assigned to a single partition, or a partition can be reserved for a certain set of states or countries. It is similar to **RANGE**, except that only **VALUES IN** may be used to specify permissible values for each partition.

**VALUES IN** is used with a list of values to be matched. For instance, you could create a partitioning scheme such as the following:

CREATE TABLE client\_firms (

id INT,

name VARCHAR(35)

)

PARTITION BY LIST (id) (

PARTITION r0 VALUES IN (1, 5, 9, 13, 17, 21),

PARTITION r1 VALUES IN (2, 6, 10, 14, 18, 22),

PARTITION r2 VALUES IN (3, 7, 11, 15, 19, 23),

PARTITION r3 VALUES IN (4, 8, 12, 16, 20, 24)

);

When using list partitioning, you must define at least one partition using **VALUES IN**. You cannot use **VALUES LESS THAN** with **PARTITION BY LIST**.

**Note**

For tables partitioned by **LIST**, the value list used with **VALUES IN** must consist of integer values only. In MySQL 8.0, you can overcome this limitation using partitioning by **LIST COLUMNS**, which is described later in this section.

**LIST COLUMNS(*column\_list*)**

This variant on **LIST** facilitates partition pruning for queries using comparison conditions on multiple columns (that is, having conditions such as **WHERE a = 5 AND b = 5** or **WHERE a = 1 AND b = 10 AND c = 5**). It enables you to specify values in multiple columns by using a list of columns in the **COLUMNS** clause and a set of column values in each **PARTITION ... VALUES IN (*value\_list*)** partition definition clause.

The rules governing regarding data types for the column list used in **LIST COLUMNS(*column\_list*)** and the value list used in **VALUES IN(*value\_list*)** are the same as those for the column list used in **RANGE COLUMNS(*column\_list*)** and the value list used in **VALUES LESS THAN(*value\_list*)**, respectively, except that in the **VALUES IN** clause, **MAXVALUE** is not permitted, and you may use **NULL**.

There is one important difference between the list of values used for **VALUES IN** with **PARTITION BY LIST COLUMNS** as opposed to when it is used with **PARTITION BY LIST**. When used with **PARTITION BY LIST COLUMNS**, each element in the **VALUES IN** clause must be a set of column values; the number of values in each set must be the same as the number of columns used in the **COLUMNS** clause, and the data types of these values must match those of the columns (and occur in the same order). In the simplest case, the set consists of a single column. The maximum number of columns that can be used in the ***column\_list*** and in the elements making up the ***value\_list*** is 16.

The table defined by the following **CREATE TABLE** statement provides an example of a table using **LIST COLUMNS** partitioning:

CREATE TABLE lc (

a INT NULL,

b INT NULL

)

PARTITION BY LIST COLUMNS(a,b) (

PARTITION p0 VALUES IN( (0,0), (NULL,NULL) ),

PARTITION p1 VALUES IN( (0,1), (0,2), (0,3), (1,1), (1,2) ),

PARTITION p2 VALUES IN( (1,0), (2,0), (2,1), (3,0), (3,1) ),

PARTITION p3 VALUES IN( (1,3), (2,2), (2,3), (3,2), (3,3) )

);

**PARTITIONS *num***

The number of partitions may optionally be specified with a **PARTITIONS *num*** clause, where ***num*** is the number of partitions. If both this clause and any **PARTITION** clauses are used, ***num*** must be equal to the total number of any partitions that are declared using **PARTITION** clauses.

**Note**

Whether or not you use a **PARTITIONS** clause in creating a table that is partitioned by **RANGE** or **LIST**, you must still include at least one **PARTITION VALUES** clause in the table definition (see below).

**SUBPARTITION BY**

A partition may optionally be divided into a number of subpartitions. This can be indicated by using the optional **SUBPARTITION BY** clause. Subpartitioning may be done by **HASH** or **KEY**. Either of these may be **LINEAR**. These work in the same way as previously described for the equivalent partitioning types. (It is not possible to subpartition by **LIST** or **RANGE**.)

The number of subpartitions can be indicated using the **SUBPARTITIONS** keyword followed by an integer value.

Rigorous checking of the value used in **PARTITIONS** or **SUBPARTITIONS** clauses is applied and this value must adhere to the following rules:

The value must be a positive, nonzero integer.

No leading zeros are permitted.

The value must be an integer literal, and cannot not be an expression. For example, **PARTITIONS 0.2E+01** is not permitted, even though **0.2E+01** evaluates to **2**. (Bug #15890)

***partition\_definition***

Each partition may be individually defined using a ***partition\_definition*** clause. The individual parts making up this clause are as follows:

**PARTITION *partition\_name***

Specifies a logical name for the partition.

**VALUES**

For range partitioning, each partition must include a **VALUES LESS THAN** clause; for list partitioning, you must specify a **VALUES IN** clause for each partition. This is used to determine which rows are to be stored in this partition. See the discussions of partitioning types in [Chapter 24, *Partitioning*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html), for syntax examples.

**[STORAGE] ENGINE**

MySQL accepts a **[STORAGE] ENGINE** option for both **PARTITION** and **SUBPARTITION**. Currently, the only way in which this option can be used is to set all partitions or all subpartitions to the same storage engine, and an attempt to set different storage engines for partitions or subpartitions in the same table raises the error ERROR 1469 (HY000): The mix of handlers in the partitions is not permitted in this version of MySQL.

**COMMENT**

An optional **COMMENT** clause may be used to specify a string that describes the partition. Example:

COMMENT = 'Data for the years previous to 1999'

The maximum length for a partition comment is 1024 characters.

**DATA DIRECTORY** and **INDEX DIRECTORY**

**DATA DIRECTORY** and **INDEX DIRECTORY** may be used to indicate the directory where, respectively, the data and indexes for this partition are to be stored. Both the ***data\_dir*** and the ***index\_dir*** must be absolute system path names.

As of MySQL 8.0.21, the directory specified in a **DATA DIRECTORY** clause must be known to **InnoDB**. For more information, see [Using the DATA DIRECTORY Clause](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-create-table-external-data-directory).

You must have the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege to use the **DATA DIRECTORY** or **INDEX DIRECTORY** partition option.

Example:

CREATE TABLE th (id INT, name VARCHAR(30), adate DATE)

PARTITION BY LIST(YEAR(adate))

(

PARTITION p1999 VALUES IN (1995, 1999, 2003)

DATA DIRECTORY = '/var/appdata/95/data'

INDEX DIRECTORY = '/var/appdata/95/idx',

PARTITION p2000 VALUES IN (1996, 2000, 2004)

DATA DIRECTORY = '/var/appdata/96/data'

INDEX DIRECTORY = '/var/appdata/96/idx',

PARTITION p2001 VALUES IN (1997, 2001, 2005)

DATA DIRECTORY = '/var/appdata/97/data'

INDEX DIRECTORY = '/var/appdata/97/idx',

PARTITION p2002 VALUES IN (1998, 2002, 2006)

DATA DIRECTORY = '/var/appdata/98/data'

INDEX DIRECTORY = '/var/appdata/98/idx'

);

**DATA DIRECTORY** and **INDEX DIRECTORY** behave in the same way as in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement's ***table\_option*** clause as used for **MyISAM** tables.

One data directory and one index directory may be specified per partition. If left unspecified, the data and indexes are stored by default in the table's database directory.

The **DATA DIRECTORY** and **INDEX DIRECTORY** options are ignored for creating partitioned tables if [**NO\_DIR\_IN\_CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_dir_in_create) is in effect.

**MAX\_ROWS** and **MIN\_ROWS**

May be used to specify, respectively, the maximum and minimum number of rows to be stored in the partition. The values for ***max\_number\_of\_rows*** and ***min\_number\_of\_rows*** must be positive integers. As with the table-level options with the same names, these act only as “suggestions” to the server and are not hard limits.

**TABLESPACE**

May be used to designate an **InnoDB** file-per-table tablespace for the partition by specifying **TABLESPACE `innodb\_file\_per\_table`**. All partitions must belong to the same storage engine.

Placing **InnoDB** table partitions in shared **InnoDB** tablespaces is not supported. Shared tablespaces include the **InnoDB** system tablespace and general tablespaces.

***subpartition\_definition***

The partition definition may optionally contain one or more ***subpartition\_definition*** clauses. Each of these consists at a minimum of the **SUBPARTITION *name***, where ***name*** is an identifier for the subpartition. Except for the replacement of the **PARTITION** keyword with **SUBPARTITION**, the syntax for a subpartition definition is identical to that for a partition definition.

Subpartitioning must be done by **HASH** or **KEY**, and can be done only on **RANGE** or **LIST** partitions. See [Section 24.2.6, “Subpartitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-subpartitions).

***Partitioning by Generated Columns***

Partitioning by generated columns is permitted. For example:

CREATE TABLE t1 (

s1 INT,

s2 INT AS (EXP(s1)) STORED

)

PARTITION BY LIST (s2) (

PARTITION p1 VALUES IN (1)

);

Partitioning sees a generated column as a regular column, which enables workarounds for limitations on functions that are not permitted for partitioning (see [Section 24.6.3, “Partitioning Limitations Relating to Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-limitations-functions)). The preceding example demonstrates this technique: [**EXP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_exp) cannot be used directly in the **PARTITION BY** clause, but a generated column defined using [**EXP()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_exp) is permitted.

#### 13.1.20.1 Files Created by CREATE TABLE

For an **InnoDB** table created in a file-per-table tablespace or general tablespace, table data and associated indexes are stored in a [.ibd file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) in the database directory. When an **InnoDB** table is created in the system tablespace, table data and indexes are stored in the [ibdata\* files](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibdata_file) that represent the system tablespace. The [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) option controls whether tables are created in file-per-table tablespaces or the system tablespace, by default. The **TABLESPACE** option can be used to place a table in a file-per-table tablespace, general tablespace, or the system tablespace, regardless of the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) setting.

For **MyISAM** tables, the storage engine creates data and index files. Thus, for each **MyISAM** table ***tbl\_name***, there are two disk files.

| **File** | **Purpose** |
| --- | --- |
| ***tbl\_name***.MYD | Data file |
| ***tbl\_name***.MYI | Index file |

[Chapter 16, *Alternative Storage Engines*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html), describes what files each storage engine creates to represent tables. If a table name contains special characters, the names for the table files contain encoded versions of those characters as described in [Section 9.2.4, “Mapping of Identifiers to File Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifier-mapping).

#### 13.1.20.2 CREATE TEMPORARY TABLE Statement

You can use the **TEMPORARY** keyword when creating a table. A **TEMPORARY** table is visible only within the current session, and is dropped automatically when the session is closed. This means that two different sessions can use the same temporary table name without conflicting with each other or with an existing non-**TEMPORARY** table of the same name. (The existing table is hidden until the temporary table is dropped.)

**InnoDB** does not support compressed temporary tables. When [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled (the default), [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) returns an error if **ROW\_FORMAT=COMPRESSED** or **KEY\_BLOCK\_SIZE** is specified. If [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is disabled, warnings are issued and the temporary table is created using a non-compressed row format. The [**innodb\_file\_per-table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) option does not affect the creation of **InnoDB** temporary tables.

[**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) causes an implicit commit, except when used with the **TEMPORARY** keyword. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

**TEMPORARY** tables have a very loose relationship with databases (schemas). Dropping a database does not automatically drop any **TEMPORARY** tables created within that database.

To create a temporary table, you must have the [**CREATE TEMPORARY TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-temporary-tables) privilege. After a session has created a temporary table, the server performs no further privilege checks on the table. The creating session can perform any operation on the table, such as [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), or [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

One implication of this behavior is that a session can manipulate its temporary tables even if the current user has no privilege to create them. Suppose that the current user does not have the [**CREATE TEMPORARY TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-temporary-tables) privilege but is able to execute a definer-context stored procedure that executes with the privileges of a user who does have [**CREATE TEMPORARY TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-temporary-tables) and that creates a temporary table. While the procedure executes, the session uses the privileges of the defining user. After the procedure returns, the effective privileges revert to those of the current user, which can still see the temporary table and perform any operation on it.

You cannot use **CREATE TEMPORY TABLE ... LIKE** to create an empty table based on the definition of a table that resides in the **mysql** tablespace, **InnoDB** system tablespace (**innodb\_system**), or a general tablespace. The tablespace definition for such a table includes a **TABLESPACE** attribute that defines the tablespace where the table resides, and the aforementioned tablespaces do not support temporary tables. To create a temporary table based on the definition of such a table, use this syntax instead:

CREATE TEMPORARY TABLE ***new\_tbl*** SELECT \* FROM ***orig\_tbl*** LIMIT 0;

**Note**

Support for **TABLESPACE = innodb\_file\_per\_table** and **TABLESPACE = innodb\_temporary** clauses with [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) is deprecated as of MySQL 8.0.13; expect it be removed in a future version of MySQL.

#### 13.1.20.3 CREATE TABLE ... LIKE Statement

Use **CREATE TABLE ... LIKE** to create an empty table based on the definition of another table, including any column attributes and indexes defined in the original table:

CREATE TABLE ***new\_tbl*** LIKE ***orig\_tbl***;

The copy is created using the same version of the table storage format as the original table. The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege is required on the original table.

**LIKE** works only for base tables, not for views.

**Important**

You cannot execute **CREATE TABLE** or **CREATE TABLE ... LIKE** while a [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement is in effect.

[**CREATE TABLE ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) makes the same checks as [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). This means that if the current SQL mode is different from the mode in effect when the original table was created, the table definition might be considered invalid for the new mode and cause the statement to fail.

For **CREATE TABLE ... LIKE**, the destination table preserves generated column information from the original table.

For **CREATE TABLE ... LIKE**, the destination table preserves expression default values from the original table.

For **CREATE TABLE ... LIKE**, the destination table preserves **CHECK** constraints from the original table, except that all the constraint names are generated.

**CREATE TABLE ... LIKE** does not preserve any **DATA DIRECTORY** or **INDEX DIRECTORY** table options that were specified for the original table, or any foreign key definitions.

If the original table is a **TEMPORARY** table, **CREATE TABLE ... LIKE** does not preserve **TEMPORARY**. To create a **TEMPORARY** destination table, use **CREATE TEMPORARY TABLE ... LIKE**.

Tables created in the **mysql** tablespace, the **InnoDB** system tablespace (**innodb\_system**), or general tablespaces include a **TABLESPACE** attribute in the table definition, which defines the tablespace where the table resides. Due to a temporary regression, **CREATE TABLE ... LIKE** preserves the **TABLESPACE** attribute and creates the table in the defined tablespace regardless of the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) setting. To avoid the **TABLESPACE** attribute when creating an empty table based on the definition of such a table, use this syntax instead:

CREATE TABLE ***new\_tbl*** SELECT \* FROM ***orig\_tbl*** LIMIT 0;

[**CREATE TABLE ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like) operations apply all **ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values to the new table.

#### 13.1.20.4 CREATE TABLE ... SELECT Statement

You can create one table from another by adding a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement at the end of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement:

CREATE TABLE ***new\_tbl*** [AS] SELECT \* FROM ***orig\_tbl***;

MySQL creates new columns for all elements in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). For example:

mysql> **CREATE TABLE test (a INT NOT NULL AUTO\_INCREMENT,**

-> **PRIMARY KEY (a), KEY(b))**

-> **ENGINE=InnoDB SELECT b,c FROM test2;**

This creates an [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) table with three columns, **a**, **b**, and **c**. The **ENGINE** option is part of the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, and should not be used following the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select); this would result in a syntax error. The same is true for other [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) options such as **CHARSET**.

Notice that the columns from the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement are appended to the right side of the table, not overlapped onto it. Take the following example:

mysql> **SELECT \* FROM foo;**

+---+

| n |

+---+

| 1 |

+---+

mysql> **CREATE TABLE bar (m INT) SELECT n FROM foo;**

Query OK, 1 row affected (0.02 sec)

Records: 1 Duplicates: 0 Warnings: 0

mysql> **SELECT \* FROM bar;**

+------+---+

| m | n |

+------+---+

| NULL | 1 |

+------+---+

1 row in set (0.00 sec)

For each row in table **foo**, a row is inserted in **bar** with the values from **foo** and default values for the new columns.

In a table resulting from [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), columns named only in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) part come first. Columns named in both parts or only in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part come after that. The data type of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) columns can be overridden by also specifying the column in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) part.

If errors occur while copying data to the table, the table is automatically dropped and not created. However, prior to MySQL 8.0.21, when row-based replication is in use, a [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statement is recorded in the binary log as two transactions, one to create the table, and the other to insert data. When the statement applied from the binary log, a failure between the two transactions or while copying data can result in replication of an empty table. That limitation is removed in MySQL 8.0.21. On storage engines that support atomic DDL, [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) is now recorded and applied as one transaction when row-based replication is in use. For more information, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

As of MySQL 8.0.21, on storage engines that support both atomic DDL and foreign key constraints, creation of foreign keys is not permitted in [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statements when row-based replication is in use. Foreign key constraints can be added later using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

You can precede the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) by **IGNORE** or **REPLACE** to indicate how to handle rows that duplicate unique key values. With **IGNORE**, rows that duplicate an existing row on a unique key value are discarded. With **REPLACE**, new rows replace rows that have the same unique key value. If neither **IGNORE** nor **REPLACE** is specified, duplicate unique key values result in an error. For more information, see [The Effect of IGNORE on Statement Execution](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#ignore-effect-on-execution).

In MySQL 8.0.19 and later, you can also use a [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statement in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part of **CREATE TABLE ... SELECT**; the **VALUES** portion of the statement must include a table alias using an **AS** clause. To name the columns coming from **VALUES**, supply column aliases with the table alias; otherwise, the default column names **column\_0**, **column\_1**, **column\_2**, ..., are used.

Otherwise, naming of columns in the table thus created follows the same rules as described previously in this section. Examples:

mysql> **CREATE TABLE tv1**

> **SELECT \* FROM (VALUES ROW(1,3,5), ROW(2,4,6)) AS v;**

mysql> **TABLE tv1;**

+----------+----------+----------+

| column\_0 | column\_1 | column\_2 |

+----------+----------+----------+

| 1 | 3 | 5 |

| 2 | 4 | 6 |

+----------+----------+----------+

mysql> CREATE TABLE tv2

> SELECT \* FROM (VALUES ROW(1,3,5), ROW(2,4,6)) AS v(x,y,z);

mysql> **TABLE tv2;**

+---+---+---+

| x | y | z |

+---+---+---+

| 1 | 3 | 5 |

| 2 | 4 | 6 |

+---+---+---+

mysql> **CREATE TABLE tv3 (a INT, b INT, c INT)**

> **SELECT \* FROM (VALUES ROW(1,3,5), ROW(2,4,6)) AS v(x,y,z);**

mysql> TABLE tv3;

+------+------+------+----------+----------+----------+

| a | b | c | column\_0 | column\_1 | column\_2 |

+------+------+------+----------+----------+----------+

| NULL | NULL | NULL | 1 | 3 | 5 |

| NULL | NULL | NULL | 2 | 4 | 6 |

+------+------+------+----------+----------+----------+

mysql> **CREATE TABLE tv4 (a INT, b INT, c INT)**

> **SELECT \* FROM (VALUES ROW(1,3,5), ROW(2,4,6)) AS v(x,y,z);**

mysql> **TABLE tv4;**

+------+------+------+---+---+---+

| a | b | c | x | y | z |

+------+------+------+---+---+---+

| NULL | NULL | NULL | 1 | 3 | 5 |

| NULL | NULL | NULL | 2 | 4 | 6 |

+------+------+------+---+---+---+

mysql> **CREATE TABLE tv5 (a INT, b INT, c INT)**

> **SELECT \* FROM (VALUES ROW(1,3,5), ROW(2,4,6)) AS v(a,b,c);**

mysql> **TABLE tv5;**

+------+------+------+

| a | b | c |

+------+------+------+

| 1 | 3 | 5 |

| 2 | 4 | 6 |

+------+------+------+

When selecting all columns and using the default column names, you can omit **SELECT \***, so the statement just used to create table **tv1** can also be written as shown here:

mysql> **CREATE TABLE tv1 VALUES ROW(1,3,5), ROW(2,4,6);**

mysql> **TABLE tv1;**

+----------+----------+----------+

| column\_0 | column\_1 | column\_2 |

+----------+----------+----------+

| 1 | 3 | 5 |

| 2 | 4 | 6 |

+----------+----------+----------+

When using [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) as the source of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), all columns are always selected into the new table, and individual columns cannot be selected as they can be when selecting from a named table; each of the following statements produces an error ([**ER\_OPERAND\_COLUMNS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_operand_columns)):

CREATE TABLE tvx

SELECT (x,z) FROM (VALUES ROW(1,3,5), ROW(2,4,6)) AS v(x,y,z);

CREATE TABLE tvx (a INT, c INT)

SELECT (x,z) FROM (VALUES ROW(1,3,5), ROW(2,4,6)) AS v(x,y,z);

Similarly, you can use a [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement in place of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). This follows the same rules as with [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values); all columns of the source table and their names in the source table are always inserted into the new table. Examples:

mysql> **TABLE t1;**

+----+----+

| a | b |

+----+----+

| 1 | 2 |

| 6 | 7 |

| 10 | -4 |

| 14 | 6 |

+----+----+

mysql> **CREATE TABLE tt1 TABLE t1;**

mysql> **TABLE tt1;**

+----+----+

| a | b |

+----+----+

| 1 | 2 |

| 6 | 7 |

| 10 | -4 |

| 14 | 6 |

+----+----+

mysql> **CREATE TABLE tt2 (x INT) TABLE t1;**

mysql> **TABLE tt2;**

+------+----+----+

| x | a | b |

+------+----+----+

| NULL | 1 | 2 |

| NULL | 6 | 7 |

| NULL | 10 | -4 |

| NULL | 14 | 6 |

+------+----+----+

Because the ordering of the rows in the underlying [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements cannot always be determined, **CREATE TABLE ... IGNORE SELECT** and **CREATE TABLE ... REPLACE SELECT** statements are flagged as unsafe for statement-based replication. Such statements produce a warning in the error log when using statement-based mode and are written to the binary log using the row-based format when using **MIXED** mode. See also [Section 17.2.1.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr).

[**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) does not automatically create any indexes for you. This is done intentionally to make the statement as flexible as possible. If you want to have indexes in the created table, you should specify these before the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement:

mysql> **CREATE TABLE bar (UNIQUE (n)) SELECT n FROM foo;**

For **CREATE TABLE ... SELECT**, the destination table does not preserve information about whether columns in the selected-from table are generated columns. The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part of the statement cannot assign values to generated columns in the destination table.

For **CREATE TABLE ... SELECT**, the destination table does preserve expression default values from the original table.

Some conversion of data types might occur. For example, the **AUTO\_INCREMENT** attribute is not preserved, and [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns can become [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) columns. Retrained attributes are **NULL** (or **NOT NULL**) and, for those columns that have them, **CHARACTER SET**, **COLLATION**, **COMMENT**, and the **DEFAULT** clause.

When creating a table with [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select), make sure to alias any function calls or expressions in the query. If you do not, the **CREATE** statement might fail or result in undesirable column names.

CREATE TABLE artists\_and\_works

SELECT artist.name, COUNT(work.artist\_id) AS number\_of\_works

FROM artist LEFT JOIN work ON artist.id = work.artist\_id

GROUP BY artist.id;

You can also explicitly specify the data type for a column in the created table:

CREATE TABLE foo (a TINYINT NOT NULL) SELECT b+1 AS a FROM bar;

For [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), if **IF NOT EXISTS** is given and the target table exists, nothing is inserted into the destination table, and the statement is not logged.

To ensure that the binary log can be used to re-create the original tables, MySQL does not permit concurrent inserts during [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). However, prior to MySQL 8.0.21, when a [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) operation is applied from the binary log when row-based replication is in use, concurrent inserts are permitted on the replicated table while copying data. That limitation is removed in MySQL 8.0.21 on storage engines that support atomic DDL. For more information, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

You cannot use **FOR UPDATE** as part of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) in a statement such as [**CREATE TABLE *new\_table* SELECT ... FROM *old\_table* ...**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select). If you attempt to do so, the statement fails.

[**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) operations apply **ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values to columns only. Table and index **ENGINE\_ATTRIBUTE** and **SECONDARY\_ENGINE\_ATTRIBUTE** values are not applied to the new table unless specified explicitly.

#### 13.1.20.5 FOREIGN KEY Constraints

MySQL supports foreign keys, which permit cross-referencing related data across tables, and foreign key constraints, which help keep the related data consistent.

A foreign key relationship involves a parent table that holds the initial column values, and a child table with column values that reference the parent column values. A foreign key constraint is defined on the child table.

The essential syntax for a defining a foreign key constraint in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement includes the following:

[CONSTRAINT [***symbol***]] FOREIGN KEY

[***index\_name***] (***col\_name***, ...)

REFERENCES ***tbl\_name*** (***col\_name***,...)

[ON DELETE ***reference\_option***]

[ON UPDATE ***reference\_option***]

***reference\_option***:

RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

Foreign key constraint usage is described under the following topics in this section:

[Identifiers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-identifiers)

[Conditions and Restrictions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-restrictions)

[Referential Actions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-referential-actions)

[Foreign Key Constraint Examples](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-examples)

[Adding Foreign Key Constraints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-adding)

[Dropping Foreign Key Constraints](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-dropping)

[Foreign Key Checks](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-checks)

[Locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-locking)

[Foreign Key Definitions and Metadata](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-metadata)

[Foreign Key Errors](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#foreign-key-errors)

##### Identifiers

Foreign key constraint naming is governed by the following rules:

The **CONSTRAINT** ***symbol*** value is used, if defined.

If the **CONSTRAINT** ***symbol*** clause is not defined, or a symbol is not included following the **CONSTRAINT** keyword, a constraint name name is generated automatically.

Prior to MySQL 8.0.16, if the **CONSTRAINT** ***symbol*** clause was not defined, or a symbol was not included following the **CONSTRAINT** keyword, both [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engines would use the **FOREIGN\_KEY *index\_name*** if defined. In MySQL 8.0.16 and higher, the **FOREIGN\_KEY *index\_name*** is ignored.

The **CONSTRAINT *symbol*** value, if defined, must be unique in the database. A duplicate ***symbol*** results in an error similar to: ERROR 1005 (HY000): Can't create table 'test.fk1' (errno: 121).

NDB Cluster stores foreign names using the same lettercase with which they are created. Prior to version 8.0.20, when processing [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and other SQL statements, [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) compared the names of foreign keys in such statements with the names as stored in a case-sensitive fashion when [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) was equal to 0. In NDB 8.0.20 and later, this value no longer has any effect on how such comparisons are made, and they are always done without regard to lettercase. (Bug #30512043)

Table and column identifiers in a **FOREIGN KEY ... REFERENCES** clause can be quoted within backticks (**`**). Alternatively, double quotation marks (**"**) can be used if the [**ANSI\_QUOTES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_ansi_quotes) SQL mode is enabled. The [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) system variable setting is also taken into account.

##### Conditions and Restrictions

Foreign key constraints are subject to the following conditions and restrictions:

Parent and child tables must use the same storage engine, and they cannot be defined as temporary tables.

Creating a foreign key constraint requires the [**REFERENCES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_references) privilege on the parent table.

Corresponding columns in the foreign key and the referenced key must have similar data types. The size and sign of fixed precision types such as [***INTEGER***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) and [***DECIMAL***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) must be the same. The length of string types need not be the same. For nonbinary (character) string columns, the character set and collation must be the same.

MySQL supports foreign key references between one column and another within a table. (A column cannot have a foreign key reference to itself.) In these cases, a “child table record” refers to a dependent record within the same table.

MySQL requires indexes on foreign keys and referenced keys so that foreign key checks can be fast and not require a table scan. In the referencing table, there must be an index where the foreign key columns are listed as the first columns in the same order. Such an index is created on the referencing table automatically if it does not exist. This index might be silently dropped later if you create another index that can be used to enforce the foreign key constraint. ***index\_name***, if given, is used as described previously.

**InnoDB** permits a foreign key to reference any index column or group of columns. However, in the referenced table, there must be an index where the referenced columns are the first columns in the same order. Hidden columns that **InnoDB** adds to an index are also considered (see [Section 15.6.2.1, “Clustered and Secondary Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-index-types)).

**NDB** requires an explicit unique key (or primary key) on any column referenced as a foreign key. **InnoDB** does not, which is an extension of standard SQL.

Index prefixes on foreign key columns are not supported. Consequently, [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns cannot be included in a foreign key because indexes on those columns must always include a prefix length.

[**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) does not currently support foreign keys for tables with user-defined partitioning. This includes both parent and child tables.

This restriction does not apply for [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables that are partitioned by **KEY** or **LINEAR KEY** (the only user partitioning types supported by the **NDB** storage engine); these may have foreign key references or be the targets of such references.

A table in a foreign key relationship cannot be altered to use another storage engine. To change the storage engine, you must drop any foreign key constraints first.

A foreign key constraint cannot reference a virtual generated column.

For information about how the MySQL implementation of foreign key constraints differs from the SQL standard, see [Section 1.7.2.3, “FOREIGN KEY Constraint Differences”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\introduction.html#ansi-diff-foreign-keys).

##### Referential Actions

When an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operation affects a key value in the parent table that has matching rows in the child table, the result depends on the referential action specified by **ON UPDATE** and **ON DELETE** subclauses of the **FOREIGN KEY** clause. Referential actions include:

**CASCADE**: Delete or update the row from the parent table and automatically delete or update the matching rows in the child table. Both **ON DELETE CASCADE** and **ON UPDATE CASCADE** are supported. Between two tables, do not define several **ON UPDATE CASCADE** clauses that act on the same column in the parent table or in the child table.

If a **FOREIGN KEY** clause is defined on both tables in a foreign key relationship, making both tables a parent and child, an **ON UPDATE CASCADE** or **ON DELETE CASCADE** subclause defined for one **FOREIGN KEY** clause must be defined for the other in order for cascading operations to succeed. If an **ON UPDATE CASCADE** or **ON DELETE CASCADE** subclause is only defined for one **FOREIGN KEY** clause, cascading operations fail with an error.

**Note**

Cascaded foreign key actions do not activate triggers.

**SET NULL**: Delete or update the row from the parent table and set the foreign key column or columns in the child table to **NULL**. Both **ON DELETE SET NULL** and **ON UPDATE SET NULL** clauses are supported.

If you specify a **SET NULL** action, make sure that you have not declared the columns in the child table as ***NOT NULL***.

**RESTRICT**: Rejects the delete or update operation for the parent table. Specifying **RESTRICT** (or **NO ACTION**) is the same as omitting the **ON DELETE** or **ON UPDATE** clause.

**NO ACTION**: A keyword from standard SQL. In MySQL, equivalent to **RESTRICT**. The MySQL Server rejects the delete or update operation for the parent table if there is a related foreign key value in the referenced table. Some database systems have deferred checks, and **NO ACTION** is a deferred check. In MySQL, foreign key constraints are checked immediately, so **NO ACTION** is the same as **RESTRICT**.

**SET DEFAULT**: This action is recognized by the MySQL parser, but both [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) and [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) reject table definitions containing **ON DELETE SET DEFAULT** or **ON UPDATE SET DEFAULT** clauses.

For storage engines that support foreign keys, MySQL rejects any [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) operation that attempts to create a foreign key value in a child table if there is no matching candidate key value in the parent table.

For an **ON DELETE** or **ON UPDATE** that is not specified, the default action is always **NO ACTION**.

As the default, an **ON DELETE NO ACTION** or **ON UPDATE NO ACTION** clause that is specified explicitly does not appear in [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) output or in tables dumped with [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump). **RESTRICT**, which is an equivalent non-default keyword, appears in [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) output and in tables dumped with [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump).

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, **ON UPDATE CASCADE** is not supported where the reference is to the parent table's primary key.

As of NDB 8.0.16: For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, **ON DELETE CASCADE** is not supported where the child table contains one or more columns of any of the [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) types. (Bug #89511, Bug #27484882)

**InnoDB** performs cascading operations using a depth-first search algorithm on the records of the index that corresponds to the foreign key constraint.

A foreign key constraint on a stored generated column cannot use **CASCADE**, **SET NULL**, or **SET DEFAULT** as **ON UPDATE** referential actions, nor can it use **SET NULL** or **SET DEFAULT** as **ON DELETE** referential actions.

A foreign key constraint on the base column of a stored generated column cannot use **CASCADE**, **SET NULL**, or **SET DEFAULT** as **ON UPDATE** or **ON DELETE** referential actions.

##### Foreign Key Constraint Examples

This simple example relates **parent** and **child** tables through a single-column foreign key:

CREATE TABLE parent (

id INT NOT NULL,

PRIMARY KEY (id)

) ENGINE=INNODB;

CREATE TABLE child (

id INT,

parent\_id INT,

INDEX par\_ind (parent\_id),

FOREIGN KEY (parent\_id)

REFERENCES parent(id)

ON DELETE CASCADE

) ENGINE=INNODB;

This is a more complex example in which a **product\_order** table has foreign keys for two other tables. One foreign key references a two-column index in the **product** table. The other references a single-column index in the **customer** table:

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),

INDEX (customer\_id),

FOREIGN KEY (product\_category, product\_id)

REFERENCES product(category, id)

ON UPDATE CASCADE ON DELETE RESTRICT,

FOREIGN KEY (customer\_id)

REFERENCES customer(id)

) ENGINE=INNODB;

##### Adding Foreign Key Constraints

You can add a foreign key constraint to an existing table using the following [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) syntax:

ALTER TABLE ***tbl\_name***

ADD [CONSTRAINT [***symbol***]] FOREIGN KEY

[***index\_name***] (***col\_name***, ...)

REFERENCES ***tbl\_name*** (***col\_name***,...)

[ON DELETE ***reference\_option***]

[ON UPDATE ***reference\_option***]

The foreign key can be self referential (referring to the same table). When you add a foreign key constraint to a table using [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), remember to first create an index on the column(s) referenced by the foreign key.

##### Dropping Foreign Key Constraints

You can drop a foreign key constraint using the following [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) syntax:

ALTER TABLE ***tbl\_name*** DROP FOREIGN KEY ***fk\_symbol***;

If the **FOREIGN KEY** clause defined a **CONSTRAINT** name when you created the constraint, you can refer to that name to drop the foreign key constraint. Otherwise, a constraint name was generated internally, and you must use that value. To determine the foreign key constraint name, use [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table):

mysql> **SHOW CREATE TABLE child\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: child

Create Table: CREATE TABLE `child` (

`id` int DEFAULT NULL,

`parent\_id` int DEFAULT NULL,

KEY `par\_ind` (`parent\_id`),

CONSTRAINT `child\_ibfk\_1` FOREIGN KEY (`parent\_id`)

REFERENCES `parent` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

mysql> **ALTER TABLE child DROP FOREIGN KEY `child\_ibfk\_1`;**

Adding and dropping a foreign key in the same [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement is supported for [**ALTER TABLE ... ALGORITHM=INPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). It is not supported for [**ALTER TABLE ... ALGORITHM=COPY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

##### Foreign Key Checks

Foreign key checking is controlled by the [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) variable, which is enabled by default. Typically, you leave this variable enabled during normal operation to enforce referential integrity. The [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) variable has the same effect on [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables as it does for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables.

The [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) variable is dynamic and supports both global and session scopes. For information about using system variables, see [Section 5.1.9, “Using System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#using-system-variables).

Disabling foreign key checking is useful when:

Dropping a table that is referenced by a foreign key constraint. A referenced table can only be dropped after [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) is disabled. When you drop a table, constraints defined on the table are also dropped.

Reloading tables in different order than required by their foreign key relationships. For example, [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) produces correct definitions of tables in the dump file, including foreign key constraints for child tables. To make it easier to reload dump files for tables with foreign key relationships, [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) automatically includes a statement in the dump output that disables [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks). This enables you to import the tables in any order in case the dump file contains tables that are not correctly ordered for foreign keys. Disabling [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) also speeds up the import operation by avoiding foreign key checks.

Executing [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) operations, to avoid foreign key checking.

Performing an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation on a table that has a foreign key relationship.

When [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) is disabled, foreign key constraints are ignored, with the following exceptions:

Recreating a table that was previously dropped returns an error if the table definition does not conform to the foreign key constraints that reference the table. The table must have the correct column names and types. It must also have indexes on the referenced keys. If these requirements are not satisfied, MySQL returns Error 1005 that refers to errno: 150 in the error message, which means that a foreign key constraint was not correctly formed.

Altering a table returns an error (errno: 150) if a foreign key definition is incorrectly formed for the altered table.

Dropping an index required by a foreign key constraint. The foreign key constraint must be removed before dropping the index.

Creating a foreign key constraint where a column references a nonmatching column type.

Disabling [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) has these additional implications:

It is permitted to drop a database that contains tables with foreign keys that are referenced by tables outside the database.

It is permitted to drop a table with foreign keys referenced by other tables.

Enabling [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) does not trigger a scan of table data, which means that rows added to a table while [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) is disabled are not checked for consistency when [**foreign\_key\_checks**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_foreign_key_checks) is re-enabled.

##### Locking

MySQL extends metadata locks, as necessary, to tables that are related by a foreign key constraint. Extending metadata locks prevents conflicting DML and DDL operations from executing concurrently on related tables. This feature also enables updates to foreign key metadata when a parent table is modified. In earlier MySQL releases, foreign key metadata, which is owned by the child table, could not be updated safely.

If a table is locked explicitly with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), any tables related by a foreign key constraint are opened and locked implicitly. For foreign key checks, a shared read-only lock ([**LOCK TABLES READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables)) is taken on related tables. For cascading updates, a shared-nothing write lock ([**LOCK TABLES WRITE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables)) is taken on related tables that are involved in the operation.

##### Foreign Key Definitions and Metadata

To view a foreign key definition, use [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table):

mysql> **SHOW CREATE TABLE child\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: child

Create Table: CREATE TABLE `child` (

`id` int DEFAULT NULL,

`parent\_id` int DEFAULT NULL,

KEY `par\_ind` (`parent\_id`),

CONSTRAINT `child\_ibfk\_1` FOREIGN KEY (`parent\_id`)

REFERENCES `parent` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

You can obtain information about foreign keys from the [**INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-key-column-usage-table) table. An example of a query against this table is shown here:

mysql> **SELECT TABLE\_SCHEMA, TABLE\_NAME, COLUMN\_NAME, CONSTRAINT\_NAME**

**FROM INFORMATION\_SCHEMA.KEY\_COLUMN\_USAGE**

**WHERE REFERENCED\_TABLE\_SCHEMA IS NOT NULL;**

+--------------+------------+-------------+-----------------+

| TABLE\_SCHEMA | TABLE\_NAME | COLUMN\_NAME | CONSTRAINT\_NAME |

+--------------+------------+-------------+-----------------+

| test | child | parent\_id | child\_ibfk\_1 |

+--------------+------------+-------------+-----------------+

You can obtain information specific to **InnoDB** foreign keys from the [**INNODB\_FOREIGN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-table) and [**INNODB\_FOREIGN\_COLS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-foreign-cols-table) tables. Example queries are show here:

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FOREIGN \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ID: test/child\_ibfk\_1

FOR\_NAME: test/child

REF\_NAME: test/parent

N\_COLS: 1

TYPE: 1

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.INNODB\_FOREIGN\_COLS \G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ID: test/child\_ibfk\_1

FOR\_COL\_NAME: parent\_id

REF\_COL\_NAME: id

POS: 0

##### Foreign Key Errors

In the event of a foreign key error involving **InnoDB** tables (usually Error 150 in the MySQL Server), information about the latest foreign key error can be obtained by checking [**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output.

mysql> SHOW ENGINE INNODB STATUS\G

...

------------------------

LATEST FOREIGN KEY ERROR

------------------------

2018-04-12 14:57:24 0x7f97a9c91700 Transaction:

TRANSACTION 7717, ACTIVE 0 sec inserting

mysql tables in use 1, locked 1

4 lock struct(s), heap size 1136, 3 row lock(s), undo log entries 3

MySQL thread id 8, OS thread handle 140289365317376, query id 14 localhost root update

INSERT INTO child VALUES (NULL, 1), (NULL, 2), (NULL, 3), (NULL, 4), (NULL, 5), (NULL, 6)

Foreign key constraint fails for table `test`.`child`:

,

CONSTRAINT `child\_ibfk\_1` FOREIGN KEY (`parent\_id`) REFERENCES `parent` (`id`) ON DELETE

CASCADE ON UPDATE CASCADE

Trying to add in child table, in index par\_ind tuple:

DATA TUPLE: 2 fields;

0: len 4; hex 80000003; asc ;;

1: len 4; hex 80000003; asc ;;

But in parent table `test`.`parent`, in index PRIMARY,

the closest match we can find is record:

PHYSICAL RECORD: n\_fields 3; compact format; info bits 0

0: len 4; hex 80000004; asc ;;

1: len 6; hex 000000001e19; asc ;;

2: len 7; hex 81000001110137; asc 7;;

...

**Warning**

If a user has table-level privileges for all parent tables, [**ER\_NO\_REFERENCED\_ROW\_2**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_no_referenced_row_2) and [**ER\_ROW\_IS\_REFERENCED\_2**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_row_is_referenced_2) error messages for foreign key operations expose information about parent tables. If a user does not have table-level privileges for all parent tables, more generic error messages are displayed instead ([**ER\_NO\_REFERENCED\_ROW**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_no_referenced_row) and [**ER\_ROW\_IS\_REFERENCED**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_row_is_referenced)).

An exception is that, for stored programs defined to execute with **DEFINER** privileges, the user against which privileges are assessed is the user in the program **DEFINER** clause, not the invoking user. If that user has table-level parent table privileges, parent table information is still displayed. In this case, it is the responsibility of the stored program creator to hide the information by including appropriate condition handlers.

#### 13.1.20.6 CHECK Constraints

Prior to MySQL 8.0.16, [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) permits only the following limited version of table **CHECK** constraint syntax, which is parsed and ignored:

CHECK (***expr***)

As of MySQL 8.0.16, [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) permits the core features of table and column **CHECK** constraints, for all storage engines. [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) permits the following **CHECK** constraint syntax, for both table constraints and column constraints:

[CONSTRAINT [***symbol***]] CHECK (***expr***) [[NOT] ENFORCED]

The optional ***symbol*** specifies a name for the constraint. If omitted, MySQL generates a name from the table name, a literal **\_chk\_**, and an ordinal number (1, 2, 3, ...). Constraint names have a maximum length of 64 characters. They are case-sensitive, but not accent-sensitive.

***expr*** specifies the constraint condition as a boolean expression that must evaluate to **TRUE** or **UNKNOWN** (for **NULL** values) for each row of the table. If the condition evaluates to **FALSE**, it fails and a constraint violation occurs. The effect of a violation depends on the statement being executed, as described later in this section.

The optional enforcement clause indicates whether the constraint is enforced:

If omitted or specified as **ENFORCED**, the constraint is created and enforced.

If specified as **NOT ENFORCED**, the constraint is created but not enforced.

A **CHECK** constraint is specified as either a table constraint or column constraint:

A table constraint does not appear within a column definition and can refer to any table column or columns. Forward references are permitted to columns appearing later in the table definition.

A column constraint appears within a column definition and can refer only to that column.

Consider this table definition:

CREATE TABLE t1

(

CHECK (c1 <> c2),

c1 INT CHECK (c1 > 10),

c2 INT CONSTRAINT c2\_positive CHECK (c2 > 0),

c3 INT CHECK (c3 < 100),

CONSTRAINT c1\_nonzero CHECK (c1 <> 0),

CHECK (c1 > c3)

);

The definition includes table constraints and column constraints, in named and unnamed formats:

The first constraint is a table constraint: It occurs outside any column definition, so it can (and does) refer to multiple table columns. This constraint contains forward references to columns not defined yet. No constraint name is specified, so MySQL generates a name.

The next three constraints are column constraints: Each occurs within a column definition, and thus can refer only to the column being defined. One of the constraints is named explicitly. MySQL generates a name for each of the other two.

The last two constraints are table constraints. One of them is named explicitly. MySQL generates a name for the other one.

As mentioned, MySQL generates a name for any **CHECK** constraint specified without one. To see the names generated for the preceding table definition, use **SHOW CREATE TABLE**:

mysql> **SHOW CREATE TABLE t1\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t1

Create Table: CREATE TABLE `t1` (

`c1` int(11) DEFAULT NULL,

`c2` int(11) DEFAULT NULL,

`c3` int(11) DEFAULT NULL,

CONSTRAINT `c1\_nonzero` CHECK ((`c1` <> 0)),

CONSTRAINT `c2\_positive` CHECK ((`c2` > 0)),

CONSTRAINT `t1\_chk\_1` CHECK ((`c1` <> `c2`)),

CONSTRAINT `t1\_chk\_2` CHECK ((`c1` > 10)),

CONSTRAINT `t1\_chk\_3` CHECK ((`c3` < 100)),

CONSTRAINT `t1\_chk\_4` CHECK ((`c1` > `c3`))

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

The SQL standard specifies that all types of constraints (primary key, unique index, foreign key, check) belong to the same namespace. In MySQL, each constraint type has its own namespace per schema (database). Consequently, **CHECK** constraint names must be unique per schema; no two tables in the same schema can share a **CHECK** constraint name. (Exception: A **TEMPORARY** table hides a non-**TEMPORARY** table of the same name, so it can have the same **CHECK** constraint names as well.)

Beginning generated constraint names with the table name helps ensure schema uniqueness because table names also must be unique within the schema.

**CHECK** condition expressions must adhere to the following rules. An error occurs if an expression contains disallowed constructs.

Nongenerated and generated columns are permitted, except columns with the **AUTO\_INCREMENT** attribute and columns in other tables.

Literals, deterministic built-in functions, and operators are permitted. A function is deterministic if, given the same data in tables, multiple invocations produce the same result, independently of the connected user. Examples of functions that are nondeterministic and fail this definition: [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id), [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user), [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now).

Stored functions and user-defined functions are not permitted.

Stored procedure and function parameters are not permitted.

Variables (system variables, user-defined variables, and stored program local variables) are not permitted.

Subqueries are not permitted.

Foreign key referential actions (**ON UPDATE**, **ON DELETE**) are prohibited on columns used in **CHECK** constraints. Likewise, **CHECK** constraints are prohibited on columns used in foreign key referential actions.

**CHECK** constraints are evaluated for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data), and [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) statements and an error occurs if a constraint evaluates to **FALSE**. If an error occurs, handling of changes already applied differs for transactional and nontransactional storage engines, and also depends on whether strict SQL mode is in effect, as described in [Strict SQL Mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode-strict).

**CHECK** constraints are evaluated for [**INSERT IGNORE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE IGNORE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**LOAD DATA ... IGNORE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data), and [**LOAD XML ... IGNORE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) statements and a warning occurs if a constraint evaluates to **FALSE**. The insert or update for any offending row is skipped.

If the constraint expression evaluates to a data type that differs from the declared column type, implicit coercion to the declared type occurs according to the usual MySQL type-conversion rules. See [Section 12.3, “Type Conversion in Expression Evaluation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#type-conversion). If type conversion fails or results in a loss of precision, an error occurs.

**Note**

Constraint expression evaluation uses the SQL mode in effect at evaluation time. If any component of the expression depends on the SQL mode, different results may occur for different uses of the table unless the SQL mode is the same during all uses.

The [**INFORMATION\_SCHEMA.CHECK\_CONSTRAINTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-check-constraints-table) table provides information about **CHECK** constraints defined on tables. See [Section 26.3.5, “The INFORMATION\_SCHEMA CHECK\_CONSTRAINTS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-check-constraints-table).

#### 13.1.20.7 Silent Column Specification Changes

In some cases, MySQL silently changes column specifications from those given in a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. These might be changes to a data type, to attributes associated with a data type, or to an index specification.

All changes are subject to the internal row-size limit of 65,535 bytes, which may cause some attempts at data type changes to fail. See [Section 8.4.7, “Limits on Table Column Count and Row Size”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#column-count-limit).

Columns that are part of a **PRIMARY KEY** are made **NOT NULL** even if not declared that way.

Trailing spaces are automatically deleted from [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) and [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) member values when the table is created.

MySQL maps certain data types used by other SQL database vendors to MySQL types. See [Section 11.9, “Using Data Types from Other Database Engines”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#other-vendor-data-types).

If you include a **USING** clause to specify an index type that is not permitted for a given storage engine, but there is another index type available that the engine can use without affecting query results, the engine uses the available type.

If strict SQL mode is not enabled, a [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) column with a length specification greater than 65535 is converted to [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), and a [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary) column with a length specification greater than 65535 is converted to [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob). Otherwise, an error occurs in either of these cases.

Specifying the **CHARACTER SET binary** attribute for a character data type causes the column to be created as the corresponding binary data type: [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) becomes [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char) becomes [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), and [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) becomes [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob). For the [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) and [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) data types, this does not occur; they are created as declared. Suppose that you specify a table using this definition:

CREATE TABLE t

(

c1 VARCHAR(10) CHARACTER SET binary,

c2 TEXT CHARACTER SET binary,

c3 ENUM('a','b','c') CHARACTER SET binary

);

The resulting table has this definition:

CREATE TABLE t

(

c1 VARBINARY(10),

c2 BLOB,

c3 ENUM('a','b','c') CHARACTER SET binary

);

To see whether MySQL used a data type other than the one you specified, issue a [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) or [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) statement after creating or altering the table.

Certain other data type changes can occur if you compress a table using [**myisampack**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisampack). See [Section 16.2.3.3, “Compressed Table Characteristics”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#compressed-format).

#### 13.1.20.8 CREATE TABLE and Generated Columns

**[CREATE TABLE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "create-table" \o "13.1.20 CREATE TABLE Statement)** supports the specification of generated columns. Values of a generated column are computed from an expression included in the column definition.

Generated columns are also supported by the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine.

The following simple example shows a table that stores the lengths of the sides of right triangles in the **sidea** and **sideb** columns, and computes the length of the hypotenuse in **sidec** (the square root of the sums of the squares of the other sides):

CREATE TABLE triangle (

sidea DOUBLE,

sideb DOUBLE,

sidec DOUBLE AS (SQRT(sidea \* sidea + sideb \* sideb))

);

INSERT INTO triangle (sidea, sideb) VALUES(1,1),(3,4),(6,8);

Selecting from the table yields this result:

mysql> **SELECT \* FROM triangle;**

+-------+-------+--------------------+

| sidea | sideb | sidec |

+-------+-------+--------------------+

| 1 | 1 | 1.4142135623730951 |

| 3 | 4 | 5 |

| 6 | 8 | 10 |

+-------+-------+--------------------+

Any application that uses the **triangle** table has access to the hypotenuse values without having to specify the expression that calculates them.

Generated column definitions have this syntax:

***col\_name*** ***data\_type*** [GENERATED ALWAYS] AS (***expr***)

[VIRTUAL | STORED] [NOT NULL | NULL]

[UNIQUE [KEY]] [[PRIMARY] KEY]

[COMMENT '***string***']

**AS (*expr*)** indicates that the column is generated and defines the expression used to compute column values. **AS** may be preceded by **GENERATED ALWAYS** to make the generated nature of the column more explicit. Constructs that are permitted or prohibited in the expression are discussed later.

The **VIRTUAL** or **STORED** keyword indicates how column values are stored, which has implications for column use:

**VIRTUAL**: Column values are not stored, but are evaluated when rows are read, immediately after any **BEFORE** triggers. A virtual column takes no storage.

**InnoDB** supports secondary indexes on virtual columns. See [Section 13.1.20.9, “Secondary Indexes and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-secondary-indexes).

**STORED**: Column values are evaluated and stored when rows are inserted or updated. A stored column does require storage space and can be indexed.

The default is **VIRTUAL** if neither keyword is specified.

It is permitted to mix **VIRTUAL** and **STORED** columns within a table.

Other attributes may be given to indicate whether the column is indexed or can be **NULL**, or provide a comment.

Generated column expressions must adhere to the following rules. An error occurs if an expression contains disallowed constructs.

Literals, deterministic built-in functions, and operators are permitted. A function is deterministic if, given the same data in tables, multiple invocations produce the same result, independently of the connected user. Examples of functions that are nondeterministic and fail this definition: [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id), [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user), [**NOW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_now).

Stored functions and user-defined functions are not permitted.

Stored procedure and function parameters are not permitted.

Variables (system variables, user-defined variables, and stored program local variables) are not permitted.

Subqueries are not permitted.

A generated column definition can refer to other generated columns, but only those occurring earlier in the table definition. A generated column definition can refer to any base (nongenerated) column in the table whether its definition occurs earlier or later.

The **AUTO\_INCREMENT** attribute cannot be used in a generated column definition.

An **AUTO\_INCREMENT** column cannot be used as a base column in a generated column definition.

If expression evaluation causes truncation or provides incorrect input to a function, the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement terminates with an error and the DDL operation is rejected.

If the expression evaluates to a data type that differs from the declared column type, implicit coercion to the declared type occurs according to the usual MySQL type-conversion rules. See [Section 12.3, “Type Conversion in Expression Evaluation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#type-conversion).

If a generated column uses the [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) data type, the setting for [**explicit\_defaults\_for\_timestamp**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_explicit_defaults_for_timestamp) is ignored. In such cases, if this variable is disabled then **NULL** is not converted to [**CURRENT\_TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-timestamp). In MySQL 8.0.22 and later, if the column is also declared as **NOT NULL**, attempting to insert **NULL** is explicitly rejected with **ER\_BAD\_NULL\_ERROR**.

**Note**

Expression evaluation uses the SQL mode in effect at evaluation time. If any component of the expression depends on the SQL mode, different results may occur for different uses of the table unless the SQL mode is the same during all uses.

For [**CREATE TABLE ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like), the destination table preserves generated column information from the original table.

For [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select), the destination table does not preserve information about whether columns in the selected-from table are generated columns. The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part of the statement cannot assign values to generated columns in the destination table.

Partitioning by generated columns is permitted. See [Table Partitioning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-partitioning).

A foreign key constraint on a stored generated column cannot use **CASCADE**, **SET NULL**, or **SET DEFAULT** as **ON UPDATE** referential actions, nor can it use **SET NULL** or **SET DEFAULT** as **ON DELETE** referential actions.

A foreign key constraint on the base column of a stored generated column cannot use **CASCADE**, **SET NULL**, or **SET DEFAULT** as **ON UPDATE** or **ON DELETE** referential actions.

A foreign key constraint cannot reference a virtual generated column.

Triggers cannot use **NEW.*col\_name*** or use **OLD.*col\_name*** to refer to generated columns.

For [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), if a generated column is inserted into, replaced, or updated explicitly, the only permitted value is **DEFAULT**.

A generated column in a view is considered updatable because it is possible to assign to it. However, if such a column is updated explicitly, the only permitted value is **DEFAULT**.

Generated columns have several use cases, such as these:

Virtual generated columns can be used as a way to simplify and unify queries. A complicated condition can be defined as a generated column and referred to from multiple queries on the table to ensure that all of them use exactly the same condition.

Stored generated columns can be used as a materialized cache for complicated conditions that are costly to calculate on the fly.

Generated columns can simulate functional indexes: Use a generated column to define a functional expression and index it. This can be useful for working with columns of types that cannot be indexed directly, such as [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) columns; see [Indexing a Generated Column to Provide a JSON Column Index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#json-column-indirect-index), for a detailed example.

For stored generated columns, the disadvantage of this approach is that values are stored twice; once as the value of the generated column and once in the index.

If a generated column is indexed, the optimizer recognizes query expressions that match the column definition and uses indexes from the column as appropriate during query execution, even if a query does not refer to the column directly by name. For details, see [Section 8.3.11, “Optimizer Use of Generated Column Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#generated-column-index-optimizations).

Example:

Suppose that a table **t1** contains **first\_name** and **last\_name** columns and that applications frequently construct the full name using an expression like this:

SELECT CONCAT(first\_name,' ',last\_name) AS full\_name FROM t1;

One way to avoid writing out the expression is to create a view **v1** on **t1**, which simplifies applications by enabling them to select **full\_name** directly without using an expression:

CREATE VIEW v1 AS

SELECT \*, CONCAT(first\_name,' ',last\_name) AS full\_name FROM t1;

SELECT full\_name FROM v1;

A generated column also enables applications to select **full\_name** directly without the need to define a view:

CREATE TABLE t1 (

first\_name VARCHAR(10),

last\_name VARCHAR(10),

full\_name VARCHAR(255) AS (CONCAT(first\_name,' ',last\_name))

);

SELECT full\_name FROM t1;

#### 13.1.20.9 Secondary Indexes and Generated Columns

**InnoDB** supports secondary indexes on virtual generated columns. Other index types are not supported. A secondary index defined on a virtual column is sometimes referred to as a “virtual index”.

A secondary index may be created on one or more virtual columns or on a combination of virtual columns and regular columns or stored generated columns. Secondary indexes that include virtual columns may be defined as **UNIQUE**.

When a secondary index is created on a virtual generated column, generated column values are materialized in the records of the index. If the index is a [covering index](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_covering_index) (one that includes all the columns retrieved by a query), generated column values are retrieved from materialized values in the index structure instead of computed “on the fly”.

There are additional write costs to consider when using a secondary index on a virtual column due to computation performed when materializing virtual column values in secondary index records during [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) operations. Even with additional write costs, secondary indexes on virtual columns may be preferable to generated stored columns, which are materialized in the clustered index, resulting in larger tables that require more disk space and memory. If a secondary index is not defined on a virtual column, there are additional costs for reads, as virtual column values must be computed each time the column's row is examined.

Values of an indexed virtual column are MVCC-logged to avoid unnecessary recomputation of generated column values during rollback or during a purge operation. The data length of logged values is limited by the index key limit of 767 bytes for **COMPACT** and **REDUNDANT** row formats, and 3072 bytes for **DYNAMIC** and **COMPRESSED** row formats.

Adding or dropping a secondary index on a virtual column is an in-place operation.

##### Indexing a Generated Column to Provide a JSON Column Index

As noted elsewhere, [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) columns cannot be indexed directly. To create an index that references such a column indirectly, you can define a generated column that extracts the information that should be indexed, then create an index on the generated column, as shown in this example:

mysql> **CREATE TABLE jemp (**

-> **c JSON,**

-> **g INT GENERATED ALWAYS AS (c->"$.id"),**

-> **INDEX i (g)**

-> **);**

Query OK, 0 rows affected (0.28 sec)

mysql> **INSERT INTO jemp (c) VALUES**

> **('{"id": "1", "name": "Fred"}'), ('{"id": "2", "name": "Wilma"}'),**

> **('{"id": "3", "name": "Barney"}'), ('{"id": "4", "name": "Betty"}');**

Query OK, 4 rows affected (0.04 sec)

Records: 4 Duplicates: 0 Warnings: 0

mysql> **SELECT c->>"$.name" AS name**

> **FROM jemp WHERE g > 2;**

+--------+

| name |

+--------+

| Barney |

| Betty |

+--------+

2 rows in set (0.00 sec)

mysql> **EXPLAIN SELECT c->>"$.name" AS name**

> **FROM jemp WHERE g > 2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: jemp

partitions: NULL

type: range

possible\_keys: i

key: i

key\_len: 5

ref: NULL

rows: 2

filtered: 100.00

Extra: Using where

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Note

Code: 1003

Message: /\* select#1 \*/ select json\_unquote(json\_extract(`test`.`jemp`.`c`,'$.name'))

AS `name` from `test`.`jemp` where (`test`.`jemp`.`g` > 2)

1 row in set (0.00 sec)

(We have wrapped the output from the last statement in this example to fit the viewing area.)

When you use [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) on a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or other SQL statement containing one or more expressions that use the **->** or **->>** operator, these expressions are translated into their equivalents using **JSON\_EXTRACT()** and (if needed) **JSON\_UNQUOTE()** instead, as shown here in the output from [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) immediately following this **EXPLAIN** statement:

mysql> **EXPLAIN SELECT c->>"$.name"**

> **FROM jemp WHERE g > 2 ORDER BY c->"$.name"\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: jemp

partitions: NULL

type: range

possible\_keys: i

key: i

key\_len: 5

ref: NULL

rows: 2

filtered: 100.00

Extra: Using where; Using filesort

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Note

Code: 1003

Message: /\* select#1 \*/ select json\_unquote(json\_extract(`test`.`jemp`.`c`,'$.name')) AS

`c->>"$.name"` from `test`.`jemp` where (`test`.`jemp`.`g` > 2) order by

json\_extract(`test`.`jemp`.`c`,'$.name')

1 row in set (0.00 sec)

See the descriptions of the [**->**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-column-path) and [**->>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_json-inline-path) operators, as well as those of the [**JSON\_EXTRACT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-extract) and [**JSON\_UNQUOTE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-unquote) functions, for additional information and examples.

This technique also can be used to provide indexes that indirectly reference columns of other types that cannot be indexed directly, such as **GEOMETRY** columns.

In MySQL 8.0.21 and later, it is also possible to create an index on a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column using the [**JSON\_VALUE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-value) function with an expression that can be used to optimize queries employing the expression. See the description of that function for more information and examples.

###### JSON columns and indirect indexing in NDB Cluster

It is also possible to use indirect indexing of JSON columns in MySQL NDB Cluster, subject to the following conditions:

[**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) handles a [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column value internally as a [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob). This means that any **NDB** table having one or more JSON columns must have a primary key, else it cannot be recorded in the binary log.

The [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine does not support indexing of virtual columns. Since the default for generated columns is **VIRTUAL**, you must specify explicitly the generated column to which to apply the indirect index as **STORED**.

The **CREATE TABLE** statement used to create the table **jempn** shown here is a version of the **jemp** table shown previously, with modifications making it compatible with **NDB**:

CREATE TABLE jempn (

a BIGINT(20) NOT NULL AUTO\_INCREMENT PRIMARY KEY,

c JSON DEFAULT NULL,

g INT GENERATED ALWAYS AS (c->"$.name") STORED,

INDEX i (g)

) ENGINE=NDB;

We can populate this table using the following [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement:

INSERT INTO jempn (a, c) VALUES

(NULL, '{"id": "1", "name": "Fred"}'),

(NULL, '{"id": "2", "name": "Wilma"}'),

(NULL, '{"id": "3", "name": "Barney"}'),

(NULL, '{"id": "4", "name": "Betty"}');

Now **NDB** can use index **i**, as shown here:

mysql> **EXPLAIN SELECT c->>"$.name" AS name**

**FROM jempn WHERE g > 2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: jempn

partitions: p0,p1

type: range

possible\_keys: i

key: i

key\_len: 5

ref: NULL

rows: 3

filtered: 100.00

Extra: Using where with pushed condition (`test`.`jempn`.`g` > 2)

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Note

Code: 1003

Message: /\* select#1 \*/ select

json\_unquote(json\_extract(`test`.`jempn`.`c`,'$.name')) AS `name` from

`test`.`jempn` where (`test`.`jempn`.`g` > 2)

1 row in set (0.00 sec)

You should keep in mind that a stored generated column uses [**DataMemory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-datamemory), and that an index on such a column uses [**IndexMemory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-indexmemory).

#### 13.1.20.10 Invisible Columns

MySQL supports invisible columns as of MySQL 8.0.23. An invisible column is normally hidden to queries, but can be accessed if explicitly referenced. Prior to MySQL 8.0.23, all columns are visible.

As an illustration of when invisible columns may be useful, suppose that an application uses **SELECT \*** queries to access a table, and must continue to work without modification even if the table is altered to add a new column that the application does not expect to be there. In a **SELECT \*** query, the **\*** evaluates to all table columns, except those that are invisible, so the solution is to add the new column as an invisible column. The column remains “hidden” from **SELECT \*** queries, and the application continues to work as previously. A newer version of the application can refer to the invisible column if necessary by explicitly referencing it.

The following sections detail how MySQL treats invisible columns.

[DDL Statements and Invisible Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#invisible-column-ddl-statements)

[DML Statements and Invisible Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#invisible-column-dml-statements)

[Invisible Column Metadata](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#invisible-column-metadata)

[The Binary Log and Invisible Columns](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#invisible-column-binary-logging)

##### DDL Statements and Invisible Columns

Columns are visible by default. To explicitly specify visibility for a new column, use a **VISIBLE** or **INVISIBLE** keyword as part of the column definition for [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table):

CREATE TABLE t1 (

i INT,

j DATE INVISIBLE

) ENGINE = InnoDB;

ALTER TABLE t1 ADD COLUMN k INT INVISIBLE;

To alter the visibility of an existing column, use a **VISIBLE** or **INVISIBLE** keyword with one of the **ALTER TABLE** column-modification clauses:

ALTER TABLE t1 CHANGE COLUMN j j DATE VISIBLE;

ALTER TABLE t1 MODIFY COLUMN j DATE INVISIBLE;

ALTER TABLE t1 ALTER COLUMN j SET VISIBLE;

A table must have at least one visible column. Attempting to make all columns invisible produces an error.

Invisible columns support the usual column attributes: **NULL**, **NOT NULL**, **AUTO\_INCREMENT**, and so forth.

Generated columns can be invisible.

Index definitions can name invisible columns, including definitions for **PRIMARY KEY** and **UNIQUE** indexes. Although a table must have at least one visible column, an index definition need not have any visible columns.

An invisible column dropped from a table is dropped in the usual way from any index definition that names the column.

Foreign key constraints can be defined on invisible columns, and foreign key constraints can reference invisible columns.

**CHECK** constraints can be defined on invisible columns. For new or modified rows, violation of a **CHECK** constraint on an invisible column produces an error.

[**CREATE TABLE ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-like) includes invisible columns, and they are invisible in the new table.

[**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) does not include invisible columns, unless they are explicitly referenced in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part. However, even if explicitly referenced, a column that is invisible in the existing table is visible in the new table:

mysql> **CREATE TABLE t1 (col1 INT, col2 INT INVISIBLE);**

mysql> **CREATE TABLE t2 AS SELECT col1, col2 FROM t1;**

mysql> **SHOW CREATE TABLE t2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t2

Create Table: CREATE TABLE `t2` (

`col1` int DEFAULT NULL,

`col2` int DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

If invisibility should be preserved, provide a definition for the invisible column in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) part of the [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) statement:

mysql> **CREATE TABLE t1 (col1 INT, col2 INT INVISIBLE);**

mysql> **CREATE TABLE t2 (col2 INT INVISIBLE) AS SELECT col1, col2 FROM t1;**

mysql> **SHOW CREATE TABLE t2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t2

Create Table: CREATE TABLE `t2` (

`col1` int DEFAULT NULL,

`col2` int DEFAULT NULL /\*!80023 INVISIBLE \*/

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

Views can refer to invisible columns by explicitly referencing them in the **SELECT** statement that defines the view. Changing a column's visibility subsequent to defining a view that references the column does not change view behavior.

##### DML Statements and Invisible Columns

For [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, an invisible column is not part of the result set unless explicitly referenced in the select list. In a select list, the **\*** and ***tbl\_name*.\*** shorthands do not include invisible columns. Natural joins do not include invisible columns.

Consider the following statement sequence:

mysql> **CREATE TABLE t1 (col1 INT, col2 INT INVISIBLE);**

mysql> **INSERT INTO t1 (col1, col2) VALUES(1, 2), (3, 4);**

mysql> **SELECT \* FROM t1;**

+------+

| col1 |

+------+

| 1 |

| 3 |

+------+

mysql> **SELECT col1, col2 FROM t1;**

+------+------+

| col1 | col2 |

+------+------+

| 1 | 2 |

| 3 | 4 |

+------+------+

The first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) does not reference the invisible column **col2** in the select list (because **\*** does not include invisible columns), so **col2** does not appear in the statement result. The second [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) does reference **col2**, so it does appear in the result.

For statements that create new rows, an invisible column is assigned its implicit default value unless explicitly referenced and assigned a value. For information about implicit defaults, see [Implicit Default Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults-implicit).

For [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) (and [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), for non-replaced rows), implicit default assignment occurs with a missing column list, an empty column list, or a nonempty column list that does not include the invisible column:

CREATE TABLE t1 (col1 INT, col2 INT INVISIBLE);

INSERT INTO t1 VALUES(...);

INSERT INTO t1 () VALUES(...);

INSERT INTO t1 (col1) VALUES(...);

For the first two [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements, the **VALUES()** list must provide a value for each visible column and no invisible column. For the third [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement, the **VALUES()** list must provide the same number of values as the number of named columns.

For [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) and [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml), implicit default assignment occurs with a missing column list or a nonempty column list that does not include the invisible column. Input rows should not include a value for the invisible column.

To assign a value other than the implicit default for the preceding statements, explicitly name the invisible column in the column list and provide a value for it.

[**INSERT INTO ... SELECT \***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) and [**REPLACE INTO ... SELECT \***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) do not include invisible columns because **\*** does not include invisible columns. Implicit default assignment occurs as described previously.

For statements that insert or ignore new rows, or that replace or modify existing rows, based on values in a **PRIMARY KEY** or **UNIQUE** index, MySQL treats invisible columns the same as visible columns: Invisible columns participate in key value comparisons. Specifically, if a new row has the same value as an existing row for a unique key value, these behaviors occur whether the index columns are visible or invisible:

With the **IGNORE** modifier, [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data), and [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) ignore the new row.

[**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) replaces the existing row with the new row. With the **REPLACE** modifier, [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) and [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) do the same.

[**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) updates the existing row.

To update invisible columns for [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements, name them and assign a value, just as for visible columns.

##### Invisible Column Metadata

Information about whether a column is visible or invisible is available from the **EXTRA** column of the [**INFORMATION\_SCHEMA.COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-columns-table) table or [**SHOW COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns) output. For example:

mysql> **SELECT TABLE\_NAME, COLUMN\_NAME, EXTRA**

**FROM INFORMATION\_SCHEMA.COLUMNS**

**WHERE TABLE\_SCHEMA = 'test' AND TABLE\_NAME = 't1';**

+------------+-------------+-----------+

| TABLE\_NAME | COLUMN\_NAME | EXTRA |

+------------+-------------+-----------+

| t1 | i | |

| t1 | j | |

| t1 | k | INVISIBLE |

+------------+-------------+-----------+

Columns are visible by default, so in that case, **EXTRA** displays no visibility information. For invisible columns, **EXTRA** displays **INVISIBLE**.

**SHOW CREATE TABLE** displays invisible columns in the table definition, with the **INVISIBLE** keyword in a version-specific comment:

mysql> **SHOW CREATE TABLE t1\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t1

Create Table: CREATE TABLE `t1` (

`i` int DEFAULT NULL,

`j` int DEFAULT NULL,

`k` int DEFAULT NULL /\*!80023 INVISIBLE \*/

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

[**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) and **mysqlpump** use **SHOW CREATE TABLE**, so they include invisible columns in dumped table definitions. They also include invisible column values in dumped data.

Reloading a dump file into an older version of MySQL that does not support invisible columns causes the version-specific comment to be ignored, which creates any invisible columns as visible.

##### The Binary Log and Invisible Columns

MySQL treats invisible columns as follows with respect to events in the binary log:

Table-creation events include the **INVISIBLE** attribute for invisible columns.

Invisible columns are treated like visible columns in row events. They are included if needed according to the [**binlog\_row\_image**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_row_image) system variable setting.

When row events are applied, invisible columns are treated like visible columns in row events. In particular, the algorithm and index to use are chosen according to the [**slave\_rows\_search\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_rows_search_algorithms) system variable setting.

Invisible columns are treated like visible columns when computing writesets. In particular, writesets include indexes defined on invisible columns.

The [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) command includes visibility in column metadata.

#### 13.1.20.11 Setting NDB\_TABLE Options

In MySQL NDB Cluster, the table comment in a **CREATE TABLE** or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement can also be used to specify an **NDB\_TABLE** option, which consists of one or more name-value pairs, separated by commas if need be, following the string **NDB\_TABLE=**. Complete syntax for names and values syntax is shown here:

COMMENT="NDB\_TABLE=***ndb\_table\_option***[,***ndb\_table\_option***[,...]]"

***ndb\_table\_option***: {

NOLOGGING={1 | 0}

| READ\_BACKUP={1 | 0}

| PARTITION\_BALANCE={FOR\_RP\_BY\_NODE | FOR\_RA\_BY\_NODE | FOR\_RP\_BY\_LDM

| FOR\_RA\_BY\_LDM | FOR\_RA\_BY\_LDM\_X\_2

| FOR\_RA\_BY\_LDM\_X\_3 | FOR\_RA\_BY\_LDM\_X\_4}

| FULLY\_REPLICATED={1 | 0}

}

Spaces are not permitted within the quoted string. The string is case-insensitive.

The four **NDB** table options that can be set as part of a comment in this way are described in more detail in the next few paragraphs.

**NOLOGGING**: Using 1 corresponds to having [**ndb\_table\_no\_logging**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#sysvar_ndb_table_no_logging) enabled, but has no actual effect. Provided as a placeholder, mostly for completeness of [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements.

**READ\_BACKUP**: Setting this option to 1 has the same effect as though [**ndb\_read\_backup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#sysvar_ndb_read_backup) were enabled; enables reading from any replica. Doing so greatly improves the performance of reads from the table at a relatively small cost to write performance. Beginning with NDB 8.0.19, 1 is the default for **READ\_BACKUP**, and the default for [**ndb\_read\_backup**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#sysvar_ndb_read_backup) is **ON** (previously, read from any replica was disabled by default).

You can set **READ\_BACKUP** for an existing table online, using an **ALTER TABLE** statement similar to one of those shown here:

ALTER TABLE ... ALGORITHM=INPLACE, COMMENT="NDB\_TABLE=READ\_BACKUP=1";

ALTER TABLE ... ALGORITHM=INPLACE, COMMENT="NDB\_TABLE=READ\_BACKUP=0";

For more information about the **ALGORITHM** option for **ALTER TABLE**, see [Section 23.5.11, “Online Operations with ALTER TABLE in NDB Cluster”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-online-operations).

**PARTITION\_BALANCE**: Provides additional control over assignment and placement of partitions. The following four schemes are supported:

**FOR\_RP\_BY\_NODE**: One partition per node.

Only one LDM on each node stores a primary partition. Each partition is stored in the same LDM (same ID) on all nodes.

**FOR\_RA\_BY\_NODE**: One partition per node group.

Each node stores a single partition, which can be either a primary replica or a backup replica. Each partition is stored in the same LDM on all nodes.

**FOR\_RP\_BY\_LDM**: One partition for each LDM on each node; the default.

This is the setting used if **READ\_BACKUP** is set to 1.

**FOR\_RA\_BY\_LDM**: One partition per LDM in each node group.

These partitions can be primary or backup partitions.

**FOR\_RA\_BY\_LDM\_X\_2**: Two partitions per LDM in each node group.

These partitions can be primary or backup partitions.

**FOR\_RA\_BY\_LDM\_X\_3**: Three partitions per LDM in each node group.

These partitions can be primary or backup partitions.

**FOR\_RA\_BY\_LDM\_X\_4**: Four partitions per LDM in each node group.

These partitions can be primary or backup partitions.

**PARTITION\_BALANCE** is the preferred interface for setting the number of partitions per table. Using **MAX\_ROWS** to force the number of partitions is deprecated but continues to be supported for backward compatibility; it is subject to removal in a future release of MySQL NDB Cluster. (Bug #81759, Bug #23544301)

**FULLY\_REPLICATED** controls whether the table is fully replicated, that is, whether each data node has a complete copy of the table. To enable full replication of the table, use **FULLY\_REPLICATED=1**.

This setting can also be controlled using the **ndb\_fully\_replicated** system variable. Setting it to **ON** enables the option by default for all new **NDB** tables; the default is **OFF**. The [**ndb\_data\_node\_neighbour**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#sysvar_ndb_data_node_neighbour) system variable is also used for fully replicated tables, to ensure that when a fully replicated table is accessed, we access the data node which is local to this MySQL Server.

An example of a **CREATE TABLE** statement using such a comment when creating an **NDB** table is shown here:

mysql> **CREATE TABLE t1 (**

> **c1 INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

> **c2 VARCHAR(100),**

> **c3 VARCHAR(100) )**

> **ENGINE=NDB**

>

**COMMENT="NDB\_TABLE=READ\_BACKUP=0,PARTITION\_BALANCE=FOR\_RP\_BY\_NODE";**

The comment is displayed as part of the ouput of [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table). The text of the comment is also available from querying the MySQL Information Schema [**TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table) table, as in this example:

mysql> **SELECT TABLE\_NAME, TABLE\_SCHEMA, TABLE\_COMMENT**

> **FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME="t1"\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_NAME: t1

TABLE\_SCHEMA: test

TABLE\_COMMENT: NDB\_TABLE=READ\_BACKUP=0,PARTITION\_BALANCE=FOR\_RP\_BY\_NODE

1 row in set (0.01 sec)

This comment syntax is also supported with [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements for **NDB** tables, as shown here:

mysql> **ALTER TABLE t1 COMMENT="NDB\_TABLE=PARTITION\_BALANCE=FOR\_RA\_BY\_NODE";**

Query OK, 0 rows affected (0.40 sec)

Records: 0 Duplicates: 0 Warnings: 0

Beginning with NDB 8.0.21, the **TABLE\_COMMENT** column displays the comment that is required to re-create the table as it is following the **ALTER TABLE** statement, like this:

mysql> **SELECT TABLE\_NAME, TABLE\_SCHEMA, TABLE\_COMMENT**

-> **FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME="t1"\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TABLE\_NAME: t1

TABLE\_SCHEMA: test

TABLE\_COMMENT: NDB\_TABLE=READ\_BACKUP=0,PARTITION\_BALANCE=FOR\_RP\_BY\_NODE

1 row in set (0.01 sec)

mysql> **SELECT TABLE\_NAME, TABLE\_SCHEMA, TABLE\_COMMENT**

> **FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME="t1";**

+------------+--------------+--------------------------------------------------+

| TABLE\_NAME | TABLE\_SCHEMA | TABLE\_COMMENT |

+------------+--------------+--------------------------------------------------+

| t1 | c | NDB\_TABLE=PARTITION\_BALANCE=FOR\_RA\_BY\_NODE |

| t1 | d | |

+------------+--------------+--------------------------------------------------+

2 rows in set (0.01 sec)

Keep in mind that a table comment used with **ALTER TABLE** replaces any existing comment which the table might have.

mysql> **ALTER TABLE t1 COMMENT="NDB\_TABLE=PARTITION\_BALANCE=FOR\_RA\_BY\_NODE";**

Query OK, 0 rows affected (0.40 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> **SELECT TABLE\_NAME, TABLE\_SCHEMA, TABLE\_COMMENT**

> **FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME="t1";**

+------------+--------------+--------------------------------------------------+

| TABLE\_NAME | TABLE\_SCHEMA | TABLE\_COMMENT |

+------------+--------------+--------------------------------------------------+

| t1 | c | NDB\_TABLE=PARTITION\_BALANCE=FOR\_RA\_BY\_NODE |

| t1 | d | |

+------------+--------------+--------------------------------------------------+

2 rows in set (0.01 sec)

Prior to NDB 8.0.21, the table comment used with **ALTER TABLE** replaced any existing comment which the table might have had. This meant that (for example) the **READ\_BACKUP** value was not carried over to the new comment set by the **ALTER TABLE** statement, and that any unspecified values reverted to their defaults. (BUG#30428829) There was thus no longer any way using SQL to retrieve the value previously set for the comment. To keep comment values from reverting to their defaults, it was necessry to preserve any such values from the existing comment string and include them in the comment passed to **ALTER TABLE**.

You can also see the value of the **PARTITION\_BALANCE** option in the output of [**ndb\_desc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-programs-ndb-desc). [**ndb\_desc**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-programs-ndb-desc) also shows whether the **READ\_BACKUP** and **FULLY\_REPLICATED** options are set for the table. See the description of this program for more information.

### 13.1.21 CREATE TABLESPACE Statement

CREATE [UNDO] TABLESPACE ***tablespace\_name***

InnoDB and NDB:

[ADD DATAFILE '***file\_name***']

[AUTOEXTEND\_SIZE [=] ***value***]

InnoDB only:

[FILE\_BLOCK\_SIZE = value]

[ENCRYPTION [=] {'Y' | 'N'}]

NDB only:

USE LOGFILE GROUP ***logfile\_group***

[EXTENT\_SIZE [=] ***extent\_size***]

[INITIAL\_SIZE [=] ***initial\_size***]

[MAX\_SIZE [=] ***max\_size***]

[NODEGROUP [=] ***nodegroup\_id***]

[WAIT]

[COMMENT [=] '***string***']

InnoDB and NDB:

[ENGINE [=] ***engine\_name***]

Reserved for future use:

[ENGINE\_ATTRIBUTE [=] '***string***']

This statement is used to create a tablespace. The precise syntax and semantics depend on the storage engine used. In standard MySQL releases, this is always an [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tablespace. MySQL NDB Cluster also supports tablespaces using the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine.

[Considerations for InnoDB](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace-innodb)

[Considerations for NDB Cluster](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace-ndb)

[Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace-options)

[Notes](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace-notes)

[InnoDB Examples](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace-innodb-examples)

[NDB Example](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace-ndb-examples)

#### Considerations for InnoDB

[**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax is used to create general tablespaces or undo tablespaces. The **UNDO** keyword, introduced in MySQL 8.0.14, must be specified to create an undo tablespace.

A general tablespace is a shared tablespace. It can hold multiple tables, and supports all table row formats. General tablespaces can be created in a location relative to or independent of the data directory.

After creating an **InnoDB** general tablespace, use [**CREATE TABLE *tbl\_name* ... TABLESPACE [=] *tablespace\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE *tbl\_name* TABLESPACE [=] *tablespace\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to add tables to the tablespace. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

Undo tablespaces contain undo logs. Undo tablespaces can be created in a chosen location by specifying a fully qualified data file path. For more information, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

#### Considerations for NDB Cluster

This statement is used to create a tablespace, which can contain one or more data files, providing storage space for NDB Cluster Disk Data tables (see [Section 23.5.10, “NDB Cluster Disk Data Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data)). One data file is created and added to the tablespace using this statement. Additional data files may be added to the tablespace by using the [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement (see [Section 13.1.10, “ALTER TABLESPACE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace)).

**Note**

All NDB Cluster Disk Data objects share the same namespace. This means that each Disk Data object must be uniquely named (and not merely each Disk Data object of a given type). For example, you cannot have a tablespace and a log file group with the same name, or a tablespace and a data file with the same name.

A log file group of one or more **UNDO** log files must be assigned to the tablespace to be created with the **USE LOGFILE GROUP** clause. ***logfile\_group*** must be an existing log file group created with [**CREATE LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group) (see [Section 13.1.16, “CREATE LOGFILE GROUP Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group)). Multiple tablespaces may use the same log file group for **UNDO** logging.

When setting **EXTENT\_SIZE** or **INITIAL\_SIZE**, you may optionally follow the number with a one-letter abbreviation for an order of magnitude, similar to those used in my.cnf. Generally, this is one of the letters **M** (for megabytes) or **G** (for gigabytes).

**INITIAL\_SIZE** and **EXTENT\_SIZE** are subject to rounding as follows:

**EXTENT\_SIZE** is rounded up to the nearest whole multiple of 32K.

**INITIAL\_SIZE** is rounded down to the nearest whole multiple of 32K; this result is rounded up to the nearest whole multiple of **EXTENT\_SIZE** (after any rounding).

**Note**

[**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) reserves 4% of a tablespace for data node restart operations. This reserved space cannot be used for data storage.

The rounding just described is done explicitly, and a warning is issued by the MySQL Server when any such rounding is performed. The rounded values are also used by the NDB kernel for calculating [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) column values and other purposes. However, to avoid an unexpected result, we suggest that you always use whole multiples of 32K in specifying these options.

When [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) is used with **ENGINE [=] NDB**, a tablespace and associated data file are created on each Cluster data node. You can verify that the data files were created and obtain information about them by querying the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table. (See the example later in this section.)

(See [Section 26.3.15, “The INFORMATION\_SCHEMA FILES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table).)

#### Options

**ADD DATAFILE**: Defines the name of a tablespace data file. This option is always required when creating an **NDB** tablespace; for **InnoDB** in MySQL 8.0.14 and later, it is required only when creating an undo tablespace. The ***file\_name***, including any specified path, must be quoted with single or double quotation marks. File names (not counting the file extension) and directory names must be at least one byte in length. Zero length file names and directory names are not supported.

Because there are considerable differences in how **InnoDB** and **NDB** treat data files, the two storage engines are covered separately in the discussion that follows.

**InnoDB data files.** An **InnoDB** tablespace supports only a single data file, whose name must include a **.ibd** extension.

To place an **InnoDB** general tablespace data file in a location outside of the data directory, include a fully qualified path or a path relative to the data directory. Only a fully qualified path is permitted for undo tablespaces. If you do not specify a path, a general tablespace is created in the data directory. An undo tablespace created without specifying a path is created in the directory defined by the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable. If the [**innodb\_undo\_directory**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_undo_directory) variable is undefined, undo tablespaces are created in the data directory.

To avoid conflicts with implicitly created file-per-table tablespaces, creating an **InnoDB** general tablespace in a subdirectory under the data directory is not supported. When creating a general tablespace or undo tablespace outside of the data directory, the directory must exist and must be known to **InnoDB** prior to creating the tablespace. To make a directory known to **InnoDB**, add it to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) value or to one of the variables whose values are appended to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) value. [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) is a read-only variable. Configuring it requires restarting the server.

If the **ADD DATAFILE** clause is not specified when creating an **InnoDB** tablespace, a tablespace data file with a unique file name is created implicitly. The unique file name is a 128 bit UUID formatted into five groups of hexadecimal numbers separated by dashes (***aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee***). A file extension is added if required by the storage engine. An .ibd file extension is added for **InnoDB** general tablespace data files. In a replication environment, the data file name created on the replication source server is not the same as the data file name created on the replica.

As of MySQL 8.0.17, the **ADD DATAFILE** clause does not permit circular directory references when creating an **InnoDB** tablespace. For example, the circular directory reference (**/../**) in the following statement is not permitted:

CREATE TABLESPACE ts1 ADD DATAFILE ts1.ibd '***any\_directory***/../ts1.ibd';

An exception to this restriction exists on Linux, where a circular directory reference is permitted if the preceding directory is a symbolic link. For example, the data file path in the example above is permitted if ***any\_directory*** is a symbolic link. (It is still permitted for data file paths to begin with '**../**'.)

**NDB data files.** An **NDB** tablespace supports multiple data files which can have any legal file names; more data files can be added to an NDB Cluster tablespace following its creation by using an [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statement.

An **NDB** tablespace data file is created by default in the data node file system directory—that is, the directory named ndb\_***nodeid***\_fs/TS under the data node's data directory ([**DataDir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-datadir)), where ***nodeid*** is the data node's [**NodeId**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#ndbparam-ndbd-nodeid). To place the data file in a location other than the default, include an absolute directory path or a path relative to the default location. If the directory specified does not exist, **NDB** attempts to create it; the system user account under which the data node process is running must have the appropriate permissions to do so.

**Note**

When determining the path used for a data file, **NDB** does not expand the **~** (tilde) character.

When multiple data nodes are run on the same physical host, the following considerations apply:

You cannot specify an absolute path when creating a data file.

It is not possible to create tablespace data files outside the data node file system directory, unless each data node has a separate data directory.

If each data node has its own data directory, data files can be created anywhere within this directory.

If each data node has its own data directory, it may also be possible to create a data file outside the node's data directory using a relative path, as long as this path resolves to a unique location on the host file system for each data node running on that host.

**FILE\_BLOCK\_SIZE**: This option—which is specific to **InnoDB** general tablespaces, and is ignored by **NDB**—defines the block size for the tablespace data file. Values can be specified in bytes or kilobytes. For example, an 8 kilobyte file block size can be specified as 8192 or 8K. If you do not specify this option, **FILE\_BLOCK\_SIZE** defaults to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. **FILE\_BLOCK\_SIZE** is required when you intend to use the tablespace for storing compressed **InnoDB** tables (**ROW\_FORMAT=COMPRESSED**). In this case, you must define the tablespace **FILE\_BLOCK\_SIZE** when creating the tablespace.

If **FILE\_BLOCK\_SIZE** is equal the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value, the tablespace can contain only tables having an uncompressed row format (**COMPACT**, **REDUNDANT**, and **DYNAMIC**). Tables with a **COMPRESSED** row format have a different physical page size than uncompressed tables. Therefore, compressed tables cannot coexist in the same tablespace as uncompressed tables.

For a general tablespace to contain compressed tables, **FILE\_BLOCK\_SIZE** must be specified, and the **FILE\_BLOCK\_SIZE** value must be a valid compressed page size in relation to the [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value. Also, the physical page size of the compressed table (**KEY\_BLOCK\_SIZE**) must be equal to **FILE\_BLOCK\_SIZE/1024**. For example, if [**innodb\_page\_size=16K**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size), and **FILE\_BLOCK\_SIZE=8K**, the **KEY\_BLOCK\_SIZE** of the table must be 8. For more information, see [Section 15.6.3.3, “General Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#general-tablespaces).

**USE LOGFILE GROUP**: Required for **NDB**, this is the name of a log file group previously created using [**CREATE LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group). Not supported for **InnoDB**, where it fails with an error.

**EXTENT\_SIZE**: This option is specific to NDB, and is not supported by InnoDB, where it fails with an error. **EXTENT\_SIZE** sets the size, in bytes, of the extents used by any files belonging to the tablespace. The default value is 1M. The minimum size is 32K, and theoretical maximum is 2G, although the practical maximum size depends on a number of factors. In most cases, changing the extent size does not have any measurable effect on performance, and the default value is recommended for all but the most unusual situations.

An extent is a unit of disk space allocation. One extent is filled with as much data as that extent can contain before another extent is used. In theory, up to 65,535 (64K) extents may used per data file; however, the recommended maximum is 32,768 (32K). The recommended maximum size for a single data file is 32G—that is, 32K extents × 1 MB per extent. In addition, once an extent is allocated to a given partition, it cannot be used to store data from a different partition; an extent cannot store data from more than one partition. This means, for example that a tablespace having a single datafile whose **INITIAL\_SIZE** (described in the following item) is 256 MB and whose **EXTENT\_SIZE** is 128M has just two extents, and so can be used to store data from at most two different disk data table partitions.

You can see how many extents remain free in a given data file by querying the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table, and so derive an estimate for how much space remains free in the file. For further discussion and examples, see [Section 26.3.15, “The INFORMATION\_SCHEMA FILES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table).

**INITIAL\_SIZE**: This option is specific to **NDB**, and is not supported by **InnoDB**, where it fails with an error.

The **INITIAL\_SIZE** parameter sets the total size in bytes of the data file that was specific using **ADD DATATFILE**. Once this file has been created, its size cannot be changed; however, you can add more data files to the tablespace using [**ALTER TABLESPACE ... ADD DATAFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace).

**INITIAL\_SIZE** is optional; its default value is 134217728 (128 MB).

On 32-bit systems, the maximum supported value for **INITIAL\_SIZE** is 4294967296 (4 GB).

**AUTOEXTEND\_SIZE**: Ignored by MySQL prior to MySQL 8.0.23; From MySQL 8.0.23, defines the amount by which **InnoDB** extends the size of the tablespace when it becomes full. The setting must be a multiple of 4MB. The default setting is 0, which causes the tablespace to be extended according to the implicit default behavior. For more information, see [Section 15.6.3.9, “Tablespace AUTOEXTEND\_SIZE Configuration”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-tablespace-autoextend-size).

Has no effect in any release of MySQL NDB Cluster 8.0, regardless of the storage engine used.

**MAX\_SIZE**: Currently ignored by MySQL; reserved for possible future use. Has no effect in any release of MySQL 8.0 or MySQL NDB Cluster 8.0, regardless of the storage engine used.

**NODEGROUP**: Currently ignored by MySQL; reserved for possible future use. Has no effect in any release of MySQL 8.0 or MySQL NDB Cluster 8.0, regardless of the storage engine used.

**WAIT**: Currently ignored by MySQL; reserved for possible future use. Has no effect in any release of MySQL 8.0 or MySQL NDB Cluster 8.0, regardless of the storage engine used.

**COMMENT**: Currently ignored by MySQL; reserved for possible future use. Has no effect in any release of MySQL 8.0 or MySQL NDB Cluster 8.0, regardless of the storage engine used.

The **ENCRYPTION** clause enables or disables page-level data encryption for an **InnoDB** general tablespace. Encryption support for general tablespaces was introduced in MySQL 8.0.13.

As of MySQL 8.0.16, if the **ENCRYPTION** clause is not specified, the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting controls whether encryption is enabled. The **ENCRYPTION** clause overrides the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting. However, if the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) variable is enabled, the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege is required to use an **ENCRYPTION** clause setting that differs from the [**default\_table\_encryption**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_table_encryption) setting.

A keyring plugin must be installed and configured before an encryption-enabled tablespace can be created.

When a general tablespace is encrypted, all tables residing in the tablespace are encrypted. Likewise, a table created in an encrypted tablespace is encrypted.

For more information, see [Section 15.13, “InnoDB Data-at-Rest Encryption”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-data-encryption)

**ENGINE**: Defines the storage engine which uses the tablespace, where ***engine\_name*** is the name of the storage engine. Currently, only the **InnoDB** storage engine is supported by standard MySQL 8.0 releases. MySQL NDB Cluster supports both **NDB** and **InnoDB** tablespaces. The value of the [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) system variable is used for **ENGINE** if the option is not specified.

The **ENGINE\_ATTRIBUTE** option (available as of MySQL 8.0.21) is used to specify tablespace attributes for primary storage engines. The option is reserved for future use.

Permitted values are a string literal containing a valid **JSON** document or an empty string (''). Invalid **JSON** is rejected.

CREATE TABLESPACE ts1 ENGINE\_ATTRIBUTE='{"***key***":"***value***"}';

**ENGINE\_ATTRIBUTE** values can be repeated without error. In this case, the last specified value is used.

**ENGINE\_ATTRIBUTE** values are not checked by the server, nor are they cleared when the table's storage engine is changed.

#### Notes

For the rules covering the naming of MySQL tablespaces, see [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers). In addition to these rules, the slash character (“/”) is not permitted, nor can you use names beginning with **innodb\_**, as this prefix is reserved for system use.

Creation of temporary general tablespaces is not supported.

General tablespaces do not support temporary tables.

The **TABLESPACE** option may be used with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) to assign an **InnoDB** table partition or subpartition to a file-per-table tablespace. All partitions must belong to the same storage engine. Assigning table partitions to shared **InnoDB** tablespaces is not supported. Shared tablespaces include the **InnoDB** system tablespace and general tablespaces.

General tablespaces support the addition of tables of any row format using [**CREATE TABLE ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) does not need to be enabled.

[**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is not applicable to general tablespaces. Tablespace management rules are strictly enforced independently of [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode). If **CREATE TABLESPACE** parameters are incorrect or incompatible, the operation fails regardless of the [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) setting. When a table is added to a general tablespace using [**CREATE TABLE ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE ... TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is ignored but the statement is evaluated as if [**innodb\_strict\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_strict_mode) is enabled.

Use **DROP TABLESPACE** to remove a tablespace. All tables must be dropped from a tablespace using [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) prior to dropping the tablespace. Before dropping an NDB Cluster tablespace you must also remove all its data files using one or more [**ALTER TABLESPACE ... DROP DATATFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace) statements. See [Section 23.5.10.1, “NDB Cluster Disk Data Objects”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data-objects).

All parts of an **InnoDB** table added to an **InnoDB** general tablespace reside in the general tablespace, including indexes and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) pages.

For an **NDB** table assigned to a tablespace, only those columns which are not indexed are stored on disk, and actually use the tablespace data files. Indexes and indexed columns for all **NDB** tables are always kept in memory.

Similar to the system tablespace, truncating or dropping tables stored in a general tablespace creates free space internally in the general tablespace [.ibd data file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) which can only be used for new **InnoDB** data. Space is not released back to the operating system as it is for file-per-table tablespaces.

A general tablespace is not associated with any database or schema.

[**ALTER TABLE ... DISCARD TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**ALTER TABLE ...IMPORT TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) are not supported for tables that belong to a general tablespace.

The server uses tablespace-level metadata locking for DDL that references general tablespaces. By comparison, the server uses table-level metadata locking for DDL that references file-per-table tablespaces.

A generated or existing tablespace cannot be changed to a general tablespace.

There is no conflict between general tablespace names and file-per-table tablespace names. The “/” character, which is present in file-per-table tablespace names, is not permitted in general tablespace names.

[**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) and [**mysqlpump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlpump) do not dump **InnoDB** [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) statements.

#### InnoDB Examples

This example demonstrates creating a general tablespace and adding three uncompressed tables of different row formats.

mysql> **CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd' ENGINE=INNODB;**

mysql> **CREATE TABLE t1 (c1 INT PRIMARY KEY) TABLESPACE ts1 ROW\_FORMAT=REDUNDANT;**

mysql> **CREATE TABLE t2 (c1 INT PRIMARY KEY) TABLESPACE ts1 ROW\_FORMAT=COMPACT;**

mysql> **CREATE TABLE t3 (c1 INT PRIMARY KEY) TABLESPACE ts1 ROW\_FORMAT=DYNAMIC;**

This example demonstrates creating a general tablespace and adding a compressed table. The example assumes a default [**innodb\_page\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_page_size) value of 16K. The **FILE\_BLOCK\_SIZE** of 8192 requires that the compressed table have a **KEY\_BLOCK\_SIZE** of 8.

mysql> **CREATE TABLESPACE `ts2` ADD DATAFILE 'ts2.ibd' FILE\_BLOCK\_SIZE = 8192 Engine=InnoDB;**

mysql> **CREATE TABLE t4 (c1 INT PRIMARY KEY) TABLESPACE ts2 ROW\_FORMAT=COMPRESSED KEY\_BLOCK\_SIZE=8;**

This example demonstrates creating a general tablespace without specifying the **ADD DATAFILE** clause, which is optional as of MySQL 8.0.14.

mysql> **CREATE TABLESPACE `ts3` ENGINE=INNODB;**

This example demonstrates creating an undo tablespace.

mysql> **CREATE UNDO TABLESPACE *undo\_003* ADD DATAFILE '*undo\_003*.ibu';**

#### NDB Example

Suppose that you wish to create an NDB Cluster Disk Data tablespace named **myts** using a datafile named mydata-1.dat. An NDB tablespace always requires the use of a log file group consisting of one or more undo log files. For this example, we first create a log file group named **mylg** that contains one undo long file named myundo-1.dat, using the [**CREATE LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group) statement shown here:

mysql> **CREATE LOGFILE GROUP myg1**

-> **ADD UNDOFILE 'myundo-1.dat'**

-> **ENGINE=NDB;**

Query OK, 0 rows affected (3.29 sec)

Now you can create the tablespace previously described using the following statement:

mysql> **CREATE TABLESPACE myts**

-> **ADD DATAFILE 'mydata-1.dat'**

-> **USE LOGFILE GROUP mylg**

-> **ENGINE=NDB;**

Query OK, 0 rows affected (2.98 sec)

You can now create a Disk Data table using a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement with the **TABLESPACE** and **STORAGE DISK** options, similar to what is shown here:

mysql> **CREATE TABLE mytable (**

-> **id INT UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY,**

-> **lname VARCHAR(50) NOT NULL,**

-> **fname VARCHAR(50) NOT NULL,**

-> **dob DATE NOT NULL,**

-> **joined DATE NOT NULL,**

-> **INDEX(last\_name, first\_name)**

-> **)**

-> **TABLESPACE myts STORAGE DISK**

-> **ENGINE=NDB;**

Query OK, 0 rows affected (1.41 sec)

It is important to note that only the **dob** and **joined** columns from **mytable** are actually stored on disk, due to the fact that the **id**, **lname**, and **fname** columns are all indexed.

As mentioned previously, when **CREATE TABLESPACE** is used with **ENGINE [=] NDB**, a tablespace and associated data file are created on each NDB Cluster data node. You can verify that the data files were created and obtain information about them by querying the [**INFORMATION\_SCHEMA.FILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-files-table) table, as shown here:

mysql> **SELECT FILE\_NAME, FILE\_TYPE, LOGFILE\_GROUP\_NAME, STATUS, EXTRA**

-> **FROM INFORMATION\_SCHEMA.FILES**

-> **WHERE TABLESPACE\_NAME = 'myts';**

+--------------+------------+--------------------+--------+----------------+

| file\_name | file\_type | logfile\_group\_name | status | extra |

+--------------+------------+--------------------+--------+----------------+

| mydata-1.dat | DATAFILE | mylg | NORMAL | CLUSTER\_NODE=5 |

| mydata-1.dat | DATAFILE | mylg | NORMAL | CLUSTER\_NODE=6 |

| NULL | TABLESPACE | mylg | NORMAL | NULL |

+--------------+------------+--------------------+--------+----------------+

3 rows in set (0.01 sec)

For additional information and examples, see [Section 23.5.10.1, “NDB Cluster Disk Data Objects”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data-objects).

### 13.1.22 CREATE TRIGGER Statement

CREATE

[DEFINER = ***user***]

TRIGGER ***trigger\_name***

***trigger\_time*** ***trigger\_event***

ON ***tbl\_name*** FOR EACH ROW

[***trigger\_order***]

***trigger\_body***

***trigger\_time***: { BEFORE | AFTER }

***trigger\_event***: { INSERT | UPDATE | DELETE }

***trigger\_order***: { FOLLOWS | PRECEDES } ***other\_trigger\_name***

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.

Trigger names exist in the schema namespace, meaning that all triggers must have unique names within a schema. Triggers in different schemas can have the same name.

This section describes [**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger) syntax. For additional discussion, see [Section 25.3.1, “Trigger Syntax and Examples”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#trigger-syntax).

[**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger) requires the [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_trigger) privilege for the table associated with the trigger. If the **DEFINER** clause is present, the privileges required depend on the ***user*** value, as discussed in [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security). If binary logging is enabled, [**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger) might require the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege, as discussed in [Section 25.7, “Stored Program Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-logging).

The **DEFINER** clause determines the security context to be used when checking access privileges at trigger activation time, as described later in this section.

***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.

Basic column value checks occur prior to trigger activation, so you cannot use **BEFORE** triggers to convert values inappropriate for the column type to valid values.

***trigger\_event*** indicates the kind of operation that activates the trigger. These ***trigger\_event*** values are permitted:

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert): The trigger activates whenever a new row is inserted into the table (for example, through [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data), and [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statements).

[**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update): The trigger activates whenever a row is modified (for example, through [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements).

[**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete): The trigger activates whenever a row is deleted from the table (for example, through [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) and [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statements). [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) and [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statements on the table do not activate this trigger, because they do not use [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete). Dropping a partition does not activate [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) triggers, either.

The ***trigger\_event*** does not represent a literal type of SQL statement that activates the trigger so much as it represents a type of table operation. For example, an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) trigger activates not only for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements but also [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statements because both statements insert rows into a table.

A potentially confusing example of this is the **INSERT INTO ... ON DUPLICATE KEY UPDATE ...** syntax: a **BEFORE INSERT** trigger activates for every row, followed by either an **AFTER INSERT** trigger or both the **BEFORE UPDATE** and **AFTER UPDATE** triggers, depending on whether there was a duplicate key for the row.

**Note**

Cascaded foreign key actions do not activate triggers.

It is possible to define multiple triggers for a given table that have the same trigger event and action time. For example, you can have two **BEFORE UPDATE** triggers for a table. By default, triggers that have the same trigger event and action time activate in the order they were created. To affect trigger order, specify a ***trigger\_order*** clause that indicates **FOLLOWS** or **PRECEDES** and the name of an existing trigger that also has the same trigger event and action time. With **FOLLOWS**, the new trigger activates after the existing trigger. With **PRECEDES**, the new trigger activates before the existing trigger.

***trigger\_body*** is the statement to execute when the trigger activates. To execute multiple statements, use the [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) compound statement construct. This also enables you to use the same statements that are permitted within stored routines. See [Section 13.6.1, “BEGIN ... END Compound Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end). Some statements are not permitted in triggers; see [Section 25.8, “Restrictions on Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-program-restrictions).

Within the trigger body, you can refer to columns in the subject table (the table associated with the trigger) by using the aliases **OLD** and **NEW**. **OLD.*col\_name*** refers to a column of an existing row before it is updated or deleted. **NEW.*col\_name*** refers to the column of a new row to be inserted or an existing row after it is updated.

Triggers cannot use **NEW.*col\_name*** or use **OLD.*col\_name*** to refer to generated columns. For information about generated columns, see [Section 13.1.20.8, “CREATE TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-generated-columns).

MySQL stores the [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) system variable setting in effect when a trigger is created, and always executes the trigger body with this setting in force, regardless of the current server SQL mode when the trigger begins executing.

The **DEFINER** clause specifies the MySQL account to be used when checking access privileges at trigger activation time. If the **DEFINER** clause is present, the ***user*** value should be a MySQL account specified as **'*user\_name*'@'*host\_name*'**, [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user), or [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user). The permitted ***user*** values depend on the privileges you hold, as discussed in [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security). Also see that section for additional information about trigger security.

If the **DEFINER** clause is omitted, the default definer is the user who executes the [**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger) statement. This is the same as specifying **DEFINER = CURRENT\_USER** explicitly.

MySQL takes the **DEFINER** user into account when checking trigger privileges as follows:

At [**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger) time, the user who issues the statement must have the [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_trigger) privilege.

At trigger activation time, privileges are checked against the **DEFINER** user. This user must have these privileges:

The [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_trigger) privilege for the subject table.

The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for the subject table if references to table columns occur using **OLD.*col\_name*** or **NEW.*col\_name*** in the trigger body.

The [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the subject table if table columns are targets of **SET NEW.*col\_name* = *value*** assignments in the trigger body.

Whatever other privileges normally are required for the statements executed by the trigger.

Within a trigger body, the [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) function returns the account used to check privileges at trigger activation time. This is the **DEFINER** user, not the user whose actions caused the trigger to be activated. For information about user auditing within triggers, see [Section 6.2.22, “SQL-Based Account Activity Auditing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-activity-auditing).

If you use [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to lock a table that has triggers, the tables used within the trigger are also locked, as described in [LOCK TABLES and Triggers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables-and-triggers).

For additional discussion of trigger use, see [Section 25.3.1, “Trigger Syntax and Examples”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#trigger-syntax).

### 13.1.23 CREATE VIEW Statement

CREATE

[OR REPLACE]

[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]

[DEFINER = ***user***]

[SQL SECURITY { DEFINER | INVOKER }]

VIEW ***view\_name*** [(***column\_list***)]

AS ***select\_statement***

[WITH [CASCADED | LOCAL] CHECK OPTION]

The [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view) statement creates a new view, or replaces an existing view if the **OR REPLACE** clause is given. If the view does not exist, [**CREATE OR REPLACE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view) is the same as [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view). If the view does exist, [**CREATE OR REPLACE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view) replaces it.

For information about restrictions on view use, see [Section 25.9, “Restrictions on Views”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#view-restrictions).

The ***select\_statement*** is a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement that provides the definition of the view. (Selecting from the view selects, in effect, using the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement.) The ***select\_statement*** can select from base tables, other views. Beginning with MySQL 8.0.19, the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement can use a [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statement as its source, or can be replaced with a [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement, as with [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select).

The view definition is “frozen” at creation time and is not affected by subsequent changes to the definitions of the underlying tables. 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, and columns dropped from the table result in an error when selecting from the view.

The **ALGORITHM** clause affects how MySQL 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view) statement requires the [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-view) privilege for the view, and some privilege for each column selected by the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. For columns used elsewhere in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, you must have the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege. If the **OR REPLACE** clause is present, you must also have the [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) privilege for the view. If the **DEFINER** clause is present, the privileges required depend on the ***user*** value, as discussed in [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security).

When a view is referenced, privilege checking occurs as described later in this section.

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, use ***db\_name.view\_name*** syntax to qualify the view name with the database name:

CREATE VIEW test.v AS SELECT \* FROM t;

Unqualified table or view names in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement are also 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 appropriate database name.

Within a database, base tables and views share the same namespace, so a base table and a view cannot have the same name.

Columns retrieved by the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement can be simple references to table columns, or expressions that use functions, constant values, operators, and so forth.

A view must have unique column names with no duplicates, just like a base table. By default, the names of the columns retrieved by the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement are used for the view column names. To define explicit names for the view columns, specify the optional ***column\_list*** clause 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement.

A view can be created from many kinds of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements. It can refer to base tables or other views. It can use joins, [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), and subqueries. The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) need not even refer to any tables:

CREATE VIEW v\_today (today) AS SELECT CURRENT\_DATE;

The following example defines a view that selects two columns from another table as well as an expression calculated from those columns:

mysql> **CREATE TABLE t (qty INT, price INT);**

mysql> **INSERT INTO t VALUES(3, 50);**

mysql> **CREATE VIEW v AS SELECT qty, price, qty\*price AS value FROM t;**

mysql> **SELECT \* FROM v;**

+------+-------+-------+

| qty | price | value |

+------+-------+-------+

| 3 | 50 | 150 |

+------+-------+-------+

A view definition is subject to the following restrictions:

The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement cannot refer to system variables or user-defined variables.

Within a stored program, the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement cannot refer to program parameters or local variables.

The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement cannot refer to prepared statement parameters.

Any table or view referred to in the definition must exist. If, after the view has been created, a table or view that the definition refers to is dropped, use of the view results in an error. To check a view definition for problems of this kind, use the [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) statement.

The definition cannot refer to a **TEMPORARY** table, and you cannot create a **TEMPORARY** view.

You cannot associate a trigger with a view.

Aliases for column names in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement are checked against the maximum column length of 64 characters (not the maximum alias length of 256 characters).

**ORDER BY** is permitted 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) keyword, and to clauses such as **INTO**, **FOR UPDATE**, **FOR SHARE**, **LOCK IN SHARE MODE**, and **PROCEDURE**.

The results obtained from a view may be affected if you change the query processing environment by changing system variables:

mysql> **CREATE VIEW v (mycol) AS SELECT 'abc';**

Query OK, 0 rows affected (0.01 sec)

mysql> **SET sql\_mode = '';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT "mycol" FROM v;**

+-------+

| mycol |

+-------+

| mycol |

+-------+

1 row in set (0.01 sec)

mysql> **SET sql\_mode = 'ANSI\_QUOTES';**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT "mycol" FROM v;**

+-------+

| mycol |

+-------+

| abc |

+-------+

1 row in set (0.00 sec)

The **DEFINER** and **SQL SECURITY** clauses determine which MySQL account to use when checking access privileges for the view when a statement is executed that references the view. The valid **SQL SECURITY** characteristic values are **DEFINER** (the default) and **INVOKER**. These indicate that the required privileges must be held by the user who defined or invoked the view, respectively.

If the **DEFINER** clause is present, the ***user*** value should be a MySQL account specified as **'*user\_name*'@'*host\_name*'**, [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user), or [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user). The permitted ***user*** values depend on the privileges you hold, as discussed in [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security). Also see that section for additional information about view security.

If the **DEFINER** clause is omitted, the default definer is the user who executes the [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view) statement. This is the same as specifying **DEFINER = CURRENT\_USER** explicitly.

Within a view definition, the [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) function returns the view's **DEFINER** value by default. For views defined with the **SQL SECURITY INVOKER** characteristic, [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) returns the account for the view's invoker. For information about user auditing within views, see [Section 6.2.22, “SQL-Based Account Activity Auditing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-activity-auditing).

Within a stored routine that is defined with the **SQL SECURITY DEFINER** characteristic, [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) returns the routine's **DEFINER** value. This also affects a view defined within such a routine, if the view definition contains a **DEFINER** value of [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user).

MySQL checks view privileges 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 some privilege for each column in the select list of the definition, and the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for each column used elsewhere in the definition. If the definition refers to a stored function, only the privileges needed to invoke the function can be checked. The privileges required at function invocation time can be checked only as it executes: For different invocations, different execution paths within the function might be taken.

The user who references a view must have appropriate privileges to access it ([**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) to select from it, [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) to insert into it, and so forth.)

When a view has been referenced, privileges for objects accessed by the view are checked against the privileges held by the view **DEFINER** account 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 **SQL SECURITY** characteristic is **DEFINER** or **INVOKER**. If the security characteristic is **DEFINER**, the function runs with the privileges of the **DEFINER** account. 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 DEFINER**. The standard says that the definer of the view, which is the same as the owner of the view's schema, gets applicable privileges on the view (for example, [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select)) and may grant them. MySQL has no concept of a schema “owner”, so MySQL adds a clause to identify the definer. The **DEFINER** clause is an extension where the intent is to have what the standard has; that is, a permanent record of who defined the view. This is why the default **DEFINER** value is the account of the view creator.

The optional **ALGORITHM** clause is a MySQL extension to standard SQL. It affects how MySQL processes the view. **ALGORITHM** takes three values: **MERGE**, **TEMPTABLE**, or **UNDEFINED**. For more information, see [Section 25.5.2, “View Processing Algorithms”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#view-algorithms), as well as [Section 8.2.2.4, “Optimizing Derived Tables, View References, and Common Table Expressions with Merging or Materialization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#derived-table-optimization).

Some views are updatable. That is, you can use them in statements such as [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), or [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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 nonupdatable.

A generated column in a view is considered updatable because it is possible to assign to it. However, if such a column is updated explicitly, the only permitted value is **DEFAULT**. For information about generated columns, see [Section 13.1.20.8, “CREATE TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-generated-columns).

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 **CASCADED** 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. **CASCADED** causes the checks for underlying views to be evaluated as well. When neither keyword is given, the default is **CASCADED**.

For more information about updatable views and the **WITH CHECK OPTION** clause, see [Section 25.5.3, “Updatable and Insertable Views”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#view-updatability), and [Section 25.5.4, “The View WITH CHECK OPTION Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#view-check-option).

### 13.1.24 DROP DATABASE Statement

DROP {DATABASE | SCHEMA} [IF EXISTS] ***db\_name***

[**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) drops all tables in the database and deletes the database. Be very careful with this statement! To use [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database), you need the [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) privilege on the database. [**DROP SCHEMA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) is a synonym for [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database).

**Important**

When a database is dropped, privileges granted specifically for the database are not automatically dropped. They must be dropped manually. See [Section 13.7.1.6, “GRANT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant).

**IF EXISTS** is used to prevent an error from occurring if the database does not exist.

If the default database is dropped, the default database is unset (the [**DATABASE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_database) function returns **NULL**).

If you use [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) on a symbolically linked database, both the link and the original database are deleted.

[**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) returns the number of tables that were removed.

The [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) statement removes from the given database directory those files and directories that MySQL itself may create during normal operation. This includes all files with the extensions shown in the following list:

.BAK

.DAT

.HSH

.MRG

.MYD

.MYI

.cfg

.db

.ibd

.ndb

If other files or directories remain in the database directory after MySQL removes those just listed, the database directory cannot be removed. In this case, you must remove any remaining files or directories manually and issue the [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) statement again.

Dropping a database does not remove any **TEMPORARY** tables that were created in that database. **TEMPORARY** tables are automatically removed when the session that created them ends. See [Section 13.1.20.2, “CREATE TEMPORARY TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-temporary-table).

You can also drop databases with [**mysqladmin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin). See [Section 4.5.2, “mysqladmin — A MySQL Server Administration Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin).

### 13.1.25 DROP EVENT Statement

DROP EVENT [IF EXISTS] ***event\_name***

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 warning for nonexistent events instead using **IF EXISTS**.

This statement requires the [**EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_event) privilege for the schema to which the event to be dropped belongs.

### 13.1.26 DROP FUNCTION Statement

The [**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function) statement is used to drop stored functions and user-defined functions (UDFs):

For information about dropping stored functions, see [Section 13.1.29, “DROP PROCEDURE and DROP FUNCTION Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-procedure).

For information about dropping user-defined functions, see [Section 13.7.4.2, “DROP FUNCTION Statement for User-Defined Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf).

### 13.1.27 DROP INDEX Statement

DROP INDEX ***index\_name*** ON ***tbl\_name***

[***algorithm\_option*** | ***lock\_option***] ...

***algorithm\_option***:

ALGORITHM [=] {DEFAULT | INPLACE | COPY}

***lock\_option***:

LOCK [=] {DEFAULT | NONE | SHARED | EXCLUSIVE}

[**DROP INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-index) drops the index named ***index\_name*** from the table ***tbl\_name***. This statement is mapped to an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement to drop the index. See [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

To drop a primary key, the index name is always **PRIMARY**, which must be specified as a quoted identifier because **PRIMARY** is a reserved word:

DROP INDEX `PRIMARY` ON t;

Indexes on variable-width columns of [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables are dropped online; that is, without any table copying. The table is not locked against access from other NDB Cluster API nodes, although it is locked against other operations on the same API node for the duration of the operation. This is done automatically by the server whenever it determines that it is possible to do so; you do not have to use any special SQL syntax or server options to cause it to happen.

**ALGORITHM** and **LOCK** clauses may be given to influence the table copying method and level of concurrency for reading and writing the table while its indexes are being modified. They have the same meaning as for the [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement. For more information, see [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table)

MySQL NDB Cluster supports online operations using the same **ALGORITHM=INPLACE** syntax supported in the standard MySQL Server. See [Section 23.5.11, “Online Operations with ALTER TABLE in NDB Cluster”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-online-operations), for more information.

### 13.1.28 DROP LOGFILE GROUP Statement

DROP LOGFILE GROUP ***logfile\_group***

ENGINE [=] ***engine\_name***

This statement drops the log file group named ***logfile\_group***. The log file group must already exist or an error results. (For information on creating log file groups, see [Section 13.1.16, “CREATE LOGFILE GROUP Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group).)

**Important**

Before dropping a log file group, you must drop all tablespaces that use that log file group for **UNDO** logging.

The required **ENGINE** clause provides the name of the storage engine used by the log file group to be dropped. Currently, the only permitted values for ***engine\_name*** are [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) and [**NDBCLUSTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html).

[**DROP LOGFILE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-logfile-group) is useful only with Disk Data storage for NDB Cluster. See [Section 23.5.10, “NDB Cluster Disk Data Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-disk-data).

### 13.1.29 DROP PROCEDURE and DROP FUNCTION Statements

DROP {PROCEDURE | FUNCTION} [IF EXISTS] ***sp\_name***

These statements are used to drop a stored routine (a stored procedure or function). That is, the specified routine is removed from the server. (**DROP FUNCTION** is also used to drop user-defined functions; see [Section 13.7.4.2, “DROP FUNCTION Statement for User-Defined Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf).)

To drop a stored routine, you must have the [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine) privilege for it. (If the **automatic\_sp\_privileges** system variable is enabled, that privilege and [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute) are granted automatically to the routine creator when the routine is created and dropped from the creator when the routine is dropped. See [Section 25.2.2, “Stored Routines and MySQL Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-routines-privileges).)

The **IF EXISTS** clause is a MySQL extension. It prevents an error from occurring if the procedure or function does not exist. A warning is produced that can be viewed with [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings).

[**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function) is also used to drop user-defined functions (see [Section 13.7.4.2, “DROP FUNCTION Statement for User-Defined Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf)).

### 13.1.30 DROP SERVER Statement

DROP SERVER [ IF EXISTS ] ***server\_name***

Drops the server definition for the server named ***server\_name***. The corresponding row in the **mysql.servers** table is deleted. This statement requires the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege.

Dropping a server for a table does not affect any **FEDERATED** tables that used this connection information when they were created. See [Section 13.1.18, “CREATE SERVER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server).

**DROP SERVER** causes an implicit commit. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

**DROP SERVER** is not written to the binary log, regardless of the logging format that is in use.

### 13.1.31 DROP SPATIAL REFERENCE SYSTEM Statement

DROP SPATIAL REFERENCE SYSTEM

[IF EXISTS]

***srid***

***srid***: ***32-bit unsigned integer***

This statement removes a [spatial reference system](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#spatial-reference-systems) (SRS) definition from the data dictionary. It requires the [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege.

Example:

DROP SPATIAL REFERENCE SYSTEM 4120;

If no SRS definition with the SRID value exists, an error occurs unless **IF EXISTS** is specified. In that case, a warning occurs rather than an error.

If the SRID value is used by some column in an existing table, an error occurs. For example:

mysql> **DROP SPATIAL REFERENCE SYSTEM 4326;**

ERROR 3716 (SR005): Can't modify SRID 4326. There is at

least one column depending on it.

To identify which column or columns use the SRID, use this query:

SELECT \* FROM INFORMATION\_SCHEMA.ST\_GEOMETRY\_COLUMNS WHERE SRS\_ID=4326;

SRID values must be in the range of 32-bit unsigned integers, with these restrictions:

SRID 0 is a valid SRID but cannot be used with [**DROP SPATIAL REFERENCE SYSTEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-spatial-reference-system).

If the value is in a reserved SRID range, a warning occurs. Reserved ranges are [0, 32767] (reserved by EPSG), [60,000,000, 69,999,999] (reserved by EPSG), and [2,000,000,000, 2,147,483,647] (reserved by MySQL). EPSG stands for the [European Petroleum Survey Group](http://epsg.org).

Users should not drop SRSs with SRIDs in the reserved ranges. If system-installed SRSs are dropped, the SRS definitions may be recreated for MySQL upgrades.

### 13.1.32 DROP TABLE Statement

DROP [TEMPORARY] TABLE [IF EXISTS]

***tbl\_name*** [, ***tbl\_name***] ...

[RESTRICT | CASCADE]

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) removes one or more tables. You must have the [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) privilege for each table.

Be careful with this statement! For each table, it removes the table definition and all table data. If the table is partitioned, the statement removes the table definition, all its partitions, all data stored in those partitions, and all partition definitions associated with the dropped table.

Dropping a table also drops any triggers for the table.

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) causes an implicit commit, except when used with the **TEMPORARY** keyword. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

**Important**

When a table is dropped, privileges granted specifically for the table are not automatically dropped. They must be dropped manually. See [Section 13.7.1.6, “GRANT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant).

If any tables named in the argument list do not exist, [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) behavior depends on whether the **IF EXISTS** clause is given:

Without **IF EXISTS**, the statement fails with an error indicating which nonexisting tables it was unable to drop, and no changes are made.

With **IF EXISTS**, no error occurs for nonexisting tables. The statement drops all named tables that do exist, and generates a **NOTE** diagnostic for each nonexistent table. These notes can be displayed with [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings). See [Section 13.7.7.42, “SHOW WARNINGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings).

**IF EXISTS** can also be useful for dropping tables in unusual circumstances under which there is an entry in the data dictionary but no table managed by the storage engine. (For example, if an abnormal server exit occurs after removal of the table from the storage engine but before removal of the data dictionary entry.)

The **TEMPORARY** keyword has the following effects:

The statement drops only **TEMPORARY** tables.

The statement does not cause an implicit commit.

No access rights are checked. A **TEMPORARY** table is visible only with the session that created it, so no check is necessary.

Including the **TEMPORARY** keyword is a good way to prevent accidentally dropping non-**TEMPORARY** tables.

The **RESTRICT** and **CASCADE** keywords do nothing. They are permitted to make porting easier from other database systems.

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) is not supported with all [**innodb\_force\_recovery**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_force_recovery) settings. See [Section 15.21.2, “Forcing InnoDB Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#forcing-innodb-recovery).

### 13.1.33 DROP TABLESPACE Statement

DROP [UNDO] TABLESPACE ***tablespace\_name***

[ENGINE [=] ***engine\_name***]

This statement drops a tablespace that was previously created using [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace). It is supported by the **NDB** and **InnoDB** storage engines.

The **UNDO** keyword, introduced in MySQL 8.0.14, must be specified to drop an undo tablespace. Only undo tablespaces created using [**CREATE UNDO TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-tablespace) syntax can be dropped. An undo tablespace must be in an **empty** state before it can be dropped. For more information, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

**ENGINE** sets the storage engine that uses the tablespace, where ***engine\_name*** is the name of the storage engine. Currently, the values **InnoDB** and **NDB** are supported. If not set, the value of [**default\_storage\_engine**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_storage_engine) is used. If it is not the same as the storage engine used to create the tablespace, the **DROP TABLESPACE** statement fails.

***tablespace\_name*** is a case-sensitive identifier in MySQL.

For an **InnoDB** general tablespace, all tables must be dropped from the tablespace prior to a **DROP TABLESPACE** operation. If the tablespace is not empty, **DROP TABLESPACE** returns an error.

An **NDB** tablespace to be dropped must not contain any data files; in other words, before you can drop an **NDB** tablespace, you must first drop each of its data files using [**ALTER TABLESPACE ... DROP DATAFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace).

#### Notes

A general **InnoDB** tablespace is not deleted automatically when the last table in the tablespace is dropped. The tablespace must be dropped explicitly using **DROP TABLESPACE *tablespace\_name***.

A [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) operation can drop tables that belong to a general tablespace but it cannot drop the tablespace, even if the operation drops all tables that belong to the tablespace. The tablespace must be dropped explicitly using **DROP TABLESPACE *tablespace\_name***.

Similar to the system tablespace, truncating or dropping tables stored in a general tablespace creates free space internally in the general tablespace [.ibd data file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) which can only be used for new **InnoDB** data. Space is not released back to the operating system as it is for file-per-table tablespaces.

#### InnoDB Examples

This example demonstrates how to drop an **InnoDB** general tablespace. The general tablespace **ts1** is created with a single table. Before dropping the tablespace, the table must be dropped.

mysql> **CREATE TABLESPACE `ts1` ADD DATAFILE 'ts1.ibd' Engine=InnoDB;**

mysql> **CREATE TABLE t1 (c1 INT PRIMARY KEY) TABLESPACE ts1 Engine=InnoDB;**

mysql> **DROP TABLE t1;**

mysql> **DROP TABLESPACE ts1;**

This example demonstrates dropping an undo tablespace. An undo tablespace must be in an **empty** state before it can be dropped. For more information, see [Section 15.6.3.4, “Undo Tablespaces”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-undo-tablespaces).

mysql> **DROP UNDO TABLESPACE *undo\_003*;**

#### NDB Example

This example shows how to drop an **NDB** tablespace **myts** having a data file named mydata-1.dat after first creating the tablespace, and assumes the existence of a log file group named **mylg** (see [Section 13.1.16, “CREATE LOGFILE GROUP Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-logfile-group)).

mysql> **CREATE TABLESPACE myts**

-> **ADD DATAFILE 'mydata-1.dat'**

-> **USE LOGFILE GROUP mylg**

-> **ENGINE=NDB;**

You must remove all data files from the tablespace using [**ALTER TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-tablespace), as shown here, before it can be dropped:

mysql> **ALTER TABLESPACE myts**

-> **DROP DATAFILE 'mydata-1.dat'**

-> **ENGINE=NDB;**

mysql> **DROP TABLESPACE myts;**

### 13.1.34 DROP TRIGGER Statement

DROP TRIGGER [IF EXISTS] [***schema\_name***.]***trigger\_name***

This statement drops a trigger. The schema (database) name is optional. If the schema is omitted, the trigger is dropped from the default schema. [**DROP TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-trigger) requires the [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_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 nonexistent trigger when using **IF EXISTS**. See [Section 13.7.7.42, “SHOW WARNINGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings).

Triggers for a table are also dropped if you drop the table.

### 13.1.35 DROP VIEW Statement

DROP VIEW [IF EXISTS]

***view\_name*** [, ***view\_name***] ...

[RESTRICT | CASCADE]

[**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) removes one or more views. You must have the [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) privilege for each view.

If any views named in the argument list do not exist, the statement fails with an error indicating by name which nonexisting views it was unable to drop, and no changes are made.

**Note**

In MySQL 5.7 and earlier, [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) returns an error if any views named in the argument list do not exist, but also drops all views in the list that do exist. Due to the change in behavior in MySQL 8.0, a partially completed [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) operation on a MySQL 5.7 replication source server fails when replicated on a MySQL 8.0 replica. To avoid this failure scenario, use **IF EXISTS** syntax in [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view) statements to prevent an error from occurring for views that do not exist. For more information, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

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 nonexistent view. See [Section 13.7.7.42, “SHOW WARNINGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings).

**RESTRICT** and **CASCADE**, if given, are parsed and ignored.

### 13.1.36 RENAME TABLE Statement

RENAME TABLE

***tbl\_name*** TO ***new\_tbl\_name***

[, ***tbl\_name2*** TO ***new\_tbl\_name2***] ...

[**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table) renames one or more tables. You must have [**ALTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter) and [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) privileges for the original table, and [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create) and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privileges for the new table.

For example, to rename a table named **old\_table** to **new\_table**, use this statement:

RENAME TABLE old\_table TO new\_table;

That statement is equivalent to the following [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement:

ALTER TABLE old\_table RENAME new\_table;

**RENAME TABLE**, unlike [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), can rename multiple tables within a single statement:

RENAME TABLE old\_table1 TO new\_table1,

old\_table2 TO new\_table2,

old\_table3 TO new\_table3;

Renaming operations are performed left to right. Thus, to swap two table names, do this (assuming that a table with the intermediary name **tmp\_table** does not already exist):

RENAME TABLE old\_table TO tmp\_table,

new\_table TO old\_table,

tmp\_table TO new\_table;

Metadata locks on tables are acquired in name order, which in some cases can make a difference in operation outcome when multiple transactions execute concurrently. See [Section 8.11.4, “Metadata Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#metadata-locking).

As of MySQL 8.0.13, you can rename tables locked with a [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement, provided that they are locked with a **WRITE** lock or are the product of renaming **WRITE**-locked tables from earlier steps in a multiple-table rename operation. For example, this is permitted:

LOCK TABLE old\_table1 WRITE;

RENAME TABLE old\_table1 TO new\_table1,

new\_table1 TO new\_table2;

This is not permitted:

LOCK TABLE old\_table1 READ;

RENAME TABLE old\_table1 TO new\_table1,

new\_table1 TO new\_table2;

Prior to MySQL 8.0.13, to execute **RENAME TABLE**, there must be no tables locked with **LOCK TABLES**.

With the transaction table locking conditions satisfied, the rename operation is done atomically; no other session can access any of the tables while the rename is in progress.

If any errors occur during a **RENAME TABLE**, the statement fails and no changes are made.

You can use **RENAME TABLE** to move a table from one database to another:

RENAME TABLE ***current\_db.tbl\_name*** TO ***other\_db.tbl\_name;***

Using this method to move all tables from one database to a different one in effect renames the database (an operation for which MySQL has no single statement), except that the original database continues to exist, albeit with no tables.

Like **RENAME TABLE**, **ALTER TABLE ... RENAME** can also be used to move a table to a different database. Regardless of the statement used, if the rename operation would move the table to a database located on a different file system, the success of the outcome is platform specific and depends on the underlying operating system calls used to move table files.

If a table has triggers, attempts to rename the table into a different database fail with a Trigger in wrong schema ([**ER\_TRG\_IN\_WRONG\_SCHEMA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_trg_in_wrong_schema)) error.

An unencrypted table can be moved to an encryption-enabled database and vice versa. However, if the [**table\_encryption\_privilege\_check**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_table_encryption_privilege_check) variable is enabled, the [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) privilege is required if the table encryption setting differs from the default database encryption.

To rename **TEMPORARY** tables, **RENAME TABLE** does not work. Use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) instead.

**RENAME TABLE** works for views, except that views cannot be renamed into a different database.

Any privileges granted specifically for a renamed table or view are not migrated to the new name. They must be changed manually.

**RENAME TABLE *tbl\_name* TO *new\_tbl\_name*** changes internally generated foreign key constraint names and user-defined foreign key constraint names that begin with the string “***tbl\_name***\_ibfk\_” to reflect the new table name. **InnoDB** interprets foreign key constraint names that begin with the string “***tbl\_name***\_ibfk\_” as internally generated names.

Foreign key constraint names that point to the renamed table are automatically updated unless there is a conflict, in which case the statement fails with an error. A conflict occurs if the renamed constraint name already exists. In such cases, you must drop and re-create the foreign keys for them to function properly.

**RENAME TABLE *tbl\_name* TO *new\_tbl\_name*** changes internally generated and user-defined **CHECK** constraint names that begin with the string “***tbl\_name***\_chk\_” to reflect the new table name. MySQL interprets **CHECK** constraint names that begin with the string “***tbl\_name***\_chk\_” as internally generated names. Example:

mysql> **SHOW CREATE TABLE t1\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t1

Create Table: CREATE TABLE `t1` (

`i1` int(11) DEFAULT NULL,

`i2` int(11) DEFAULT NULL,

CONSTRAINT `t1\_chk\_1` CHECK ((`i1` > 0)),

CONSTRAINT `t1\_chk\_2` CHECK ((`i2` < 0))

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

1 row in set (0.02 sec)

mysql> **RENAME TABLE t1 TO t3;**

Query OK, 0 rows affected (0.03 sec)

mysql> **SHOW CREATE TABLE t3\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t3

Create Table: CREATE TABLE `t3` (

`i1` int(11) DEFAULT NULL,

`i2` int(11) DEFAULT NULL,

CONSTRAINT `t3\_chk\_1` CHECK ((`i1` > 0)),

CONSTRAINT `t3\_chk\_2` CHECK ((`i2` < 0))

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

1 row in set (0.01 sec)

### 13.1.37 TRUNCATE TABLE Statement

TRUNCATE [TABLE] ***tbl\_name***

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) empties a table completely. It requires the [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) privilege. Logically, [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) is similar to a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement that deletes all rows, or a sequence of [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) and [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements.

To achieve high performance, [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) bypasses the DML method of deleting data. Thus, it does not cause **ON DELETE** triggers to fire, it cannot be performed for **InnoDB** tables with parent-child foreign key relationships, and it cannot be rolled back like a DML operation. However, **TRUNCATE TABLE** operations on tables that use an atomic DDL-supported storage engine are either fully committed or rolled back if the server halts during their operation. For more information, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

Although [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) is similar to [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), it is classified as a DDL statement rather than a DML statement. It differs from [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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, and so cannot be rolled back. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

Truncation operations cannot be performed if the session holds an active table lock.

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) fails for an **InnoDB** table or [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) table if there are any **FOREIGN KEY** constraints from other tables that reference the table. Foreign key constraints between columns of the same table are permitted.

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 definition is valid, the table can be re-created as an empty table with [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table), even if the data or index files have become corrupted.

Any **AUTO\_INCREMENT** value is reset to its start value. This is true even for **MyISAM** and **InnoDB**, which normally do not reuse sequence values.

When used with partitioned tables, [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) preserves the partitioning; that is, the data and index files are dropped and re-created, while the partition definitions are unaffected.

The [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement does not invoke **ON DELETE** triggers.

Truncating a corrupted **InnoDB** table is supported.

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) is treated for purposes of binary logging and replication as DDL rather than DML, and is always logged as a statement.

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) for a table closes all handlers for the table that were opened with [**HANDLER OPEN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler).

In MySQL 5.7 and earlier, on a system with a large buffer pool and [**innodb\_adaptive\_hash\_index**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_adaptive_hash_index) enabled, a **TRUNCATE TABLE** operation could cause a temporary drop in system performance due to an LRU scan that occurred when removing the table's adaptive hash index entries (Bug #68184). The remapping of [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) to [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) and [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) in MySQL 8.0 avoids the problematic LRU scan.

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) can be used with Performance Schema summary tables, but the effect is to reset the summary columns to 0 or **NULL**, not to remove rows. See [Section 27.12.20, “Performance Schema Summary Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-summary-tables).

Truncating an **InnoDB** table that resides in a file-per-table tablespace drops the existing tablespace and creates a new one. As of MySQL 8.0.21, if the tablespace was created with an earlier version and resides in an unknown directory, **InnoDB** creates the new tablespace in the default location and writes the following warning to the error log: The DATA DIRECTORY location must be in a known directory. The DATA DIRECTORY location will be ignored and the file will be put into the default datadir location. Known directories are those defined by the [**datadir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_datadir), [**innodb\_data\_home\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_data_home_dir), and [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) variables. To have [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) create the tablespace in its current location, add the directory to the [**innodb\_directories**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_directories) setting before running [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table).

## 13.2 Data Manipulation Statements

[13.2.1 CALL Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call)

[13.2.2 DELETE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete)

[13.2.3 DO Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do)

[13.2.4 HANDLER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler)

[13.2.5 IMPORT TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table)

[13.2.6 INSERT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)

[13.2.7 LOAD DATA Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data)

[13.2.8 LOAD XML Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml)

[13.2.9 REPLACE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace)

[13.2.10 SELECT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)

[13.2.11 Subqueries](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subqueries)

[13.2.12 TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table)

[13.2.13 UPDATE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update)

[13.2.14 VALUES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values)

[13.2.15 WITH (Common Table Expressions)](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with)

### 13.2.1 CALL Statement

CALL ***sp\_name***([***parameter***[,...]])

CALL ***sp\_name***[()]

The [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statement invokes a stored procedure that was defined previously with [**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure).

Stored procedures that take no arguments can be invoked without parentheses. That is, **CALL p()** and **CALL p** are equivalent.

[**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) can pass back values to its caller using parameters that are declared as **OUT** or **INOUT** parameters. 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()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) function; from the C API, call the [**mysql\_affected\_rows()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-affected-rows.html) function.

For information about the effect of unhandled conditions on procedure parameters, see [Section 13.6.7.8, “Condition Handling and OUT or INOUT Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#conditions-and-parameters).

要从一个使用OUT或INOUT参数的存储过程中取回一个值，需要通过一个用户变量来传递该参数，然后在存储过程返回后检查该变量的值。(如果你是在另一个存储过程或函数中调用存储过程，你也可以通过一个常规参数或本地常规变量作为IN或INOUT参数。) 对于一个INOUT参数，在把它传递给存储过程之前，要先初始化它的值。下面这个存储过程有一个OUT参数，存储过程将其设置为当前的服务器版本，还有一个INOUT值，存储过程将其从当前值中增加1

CREATE PROCEDURE p (OUT ver\_param VARCHAR(25), INOUT incr\_param INT)

BEGIN

# Set value of OUT parameter

SELECT VERSION() INTO ver\_param;

# Increment value of INOUT parameter

SET incr\_param = incr\_param + 1;

END;

Before calling the procedure, initialize the variable to be passed as the **INOUT** parameter. After calling the procedure, you can see that the values of the two variables are set or modified:

mysql> **SET @increment = 10;**

mysql> **CALL p(@version, @increment);**

mysql> **SELECT @version, @increment;**

+--------------------+------------+

| @version | @increment |

+--------------------+------------+

| 8.0.3-rc-debug-log | 11 |

+--------------------+------------+

In prepared [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statements used with [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare) and [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute), placeholders can be used for **IN** parameters, **OUT**, and **INOUT** parameters. These types of parameters can be used as follows:

mysql> **SET @increment = 10;**

mysql> **PREPARE s FROM 'CALL p(?, ?)';**

mysql> **EXECUTE s USING @version, @increment;**

mysql> **SELECT @version, @increment;**

+--------------------+------------+

| @version | @increment |

+--------------------+------------+

| 8.0.3-rc-debug-log | 11 |

+--------------------+------------+

To write C programs that use the [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) SQL statement to execute stored procedures that produce result sets, the **CLIENT\_MULTI\_RESULTS** flag must be enabled. This is because each [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) returns a result to indicate the call status, in addition to any result sets that might be returned by statements executed within the procedure. **CLIENT\_MULTI\_RESULTS** must also be enabled if [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) is used to execute any stored procedure that contains prepared statements. It cannot be determined when such a procedure is loaded whether those statements produce result sets, so it is necessary to assume that they do so.

**CLIENT\_MULTI\_RESULTS** can be enabled when you call [**mysql\_real\_connect()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-connect.html), either explicitly by passing the **CLIENT\_MULTI\_RESULTS** flag itself, or implicitly by passing **CLIENT\_MULTI\_STATEMENTS** (which also enables **CLIENT\_MULTI\_RESULTS**). **CLIENT\_MULTI\_RESULTS** is enabled by default.

To process the result of a [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statement executed using [**mysql\_query()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-query.html) or [**mysql\_real\_query()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-query.html), use a loop that calls [**mysql\_next\_result()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-next-result.html) to determine whether there are more results. For an example, see [Multiple Statement Execution Support](https://dev.mysql.com/doc/c-api/8.0/en/c-api-multiple-queries.html).

C programs can use the prepared-statement interface to execute [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statements and access **OUT** and **INOUT** parameters. This is done by processing the result of a [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statement using a loop that calls [**mysql\_stmt\_next\_result()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-stmt-next-result.html) to determine whether there are more results. For an example, see [Prepared CALL Statement Support](https://dev.mysql.com/doc/c-api/8.0/en/c-api-prepared-call-statements.html). Languages that provide a MySQL interface can use prepared [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statements to directly retrieve **OUT** and **INOUT** procedure parameters.

Metadata changes to objects referred to by stored programs are detected and cause automatic reparsing of the affected statements when the program is next executed. For more information, see [Section 8.10.3, “Caching of Prepared Statements and Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#statement-caching).

### 13.2.2 DELETE Statement

**[DELETE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "delete" \o "13.2.2 DELETE Statement)** is a DML statement that removes rows from a table.

A [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement can start with a [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause to define common table expressions accessible within the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete). See [Section 13.2.15, “WITH (Common Table Expressions)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with).

#### Single-Table Syntax

DELETE [LOW\_PRIORITY] [QUICK] [IGNORE] FROM ***tbl\_name*** [[AS] ***tbl\_alias***]

[PARTITION (***partition\_name*** [, ***partition\_name***] ...)]

[WHERE ***where\_condition***]

[ORDER BY ...]

[LIMIT ***row\_count***]

The **DELETE** statement deletes rows from ***tbl\_name*** and returns the number of deleted rows. To check the number of deleted rows, call the [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) function described in [Section 12.16, “Information Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#information-functions).

#### Main Clauses

The conditions in the optional **WHERE** clause identify which rows to delete. With no **WHERE** clause, all rows are deleted.

***where\_condition*** is an expression that evaluates to true for each row to be deleted. It is specified as described in [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

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. These clauses apply to single-table deletes, but not multi-table deletes.

#### Multiple-Table Syntax

DELETE [LOW\_PRIORITY] [QUICK] [IGNORE]

***tbl\_name***[.\*] [, ***tbl\_name***[.\*]] ...

FROM ***table\_references***

[WHERE ***where\_condition***]

DELETE [LOW\_PRIORITY] [QUICK] [IGNORE]

FROM ***tbl\_name***[.\*] [, ***tbl\_name***[.\*]] ...

USING ***table\_references***

[WHERE ***where\_condition***]

#### Privileges

You need the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privilege on a table to delete rows from it. You need only the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for any columns that are only read, such as those named in the **WHERE** clause.

#### Performance

When you do not need to know the number of deleted rows, the [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) statement is a faster way to empty a table than a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement with no **WHERE** clause. Unlike [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) cannot be used within a transaction or if you have a lock on the table. See [Section 13.1.37, “TRUNCATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) and [Section 13.3.6, “LOCK TABLES and UNLOCK TABLES Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables).

The speed of delete operations may also be affected by factors discussed in [Section 8.2.5.3, “Optimizing DELETE Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#delete-optimization).

To ensure that a given [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement does not take too much time, the MySQL-specific **LIMIT *row\_count*** clause for [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) specifies the maximum number of rows to be deleted. If the number of rows to delete is larger than the limit, repeat the **DELETE** statement until the number of affected rows is less than the **LIMIT** value.

#### Subqueries

You cannot delete from a table and select from the same table in a subquery.

#### Partitioned Table Support

**DELETE** supports explicit partition selection using the **PARTITION** clause, which takes a list of the comma-separated names of one or more partitions or subpartitions (or both) from which to select rows to be dropped. Partitions not included in the list are ignored. Given a partitioned table **t** with a partition named **p0**, executing the statement **DELETE FROM t PARTITION (p0)** has the same effect on the table as executing [**ALTER TABLE t TRUNCATE PARTITION (p0)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table); in both cases, all rows in partition **p0** are dropped.

**PARTITION** can be used along with a **WHERE** condition, in which case the condition is tested only on rows in the listed partitions. For example, **DELETE FROM t PARTITION (p0) WHERE c < 5** deletes rows only from partition **p0** for which the condition **c < 5** is true; rows in any other partitions are not checked and thus not affected by the **DELETE**.

The **PARTITION** clause can also be used in multiple-table **DELETE** statements. You can use up to one such option per table named in the **FROM** option.

For more information and examples, see [Section 24.5, “Partition Selection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-selection).

#### Auto-Increment Columns

If you delete the row containing the maximum value for an **AUTO\_INCREMENT** column, the value is not reused for a **MyISAM** or **InnoDB** table. If you delete all rows in the table with **DELETE FROM *tbl\_name*** (without a **WHERE** clause) in [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) mode, the sequence starts over for all storage engines except **InnoDB** and **MyISAM**. There are some exceptions to this behavior for **InnoDB** tables, as discussed in [Section 15.6.1.6, “AUTO\_INCREMENT Handling in InnoDB”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-auto-increment-handling).

For **MyISAM** tables, you can specify an **AUTO\_INCREMENT** secondary column in a multiple-column key. In this case, reuse of values deleted from the top of the sequence occurs even for **MyISAM** tables. See [Section 3.6.9, “Using AUTO\_INCREMENT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\tutorial.html#example-auto-increment).

#### Modifiers

The [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement supports the following modifiers:

If you specify the **LOW\_PRIORITY** modifier, the server delays execution of the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) until no other clients are reading from the table. This affects only storage engines that use only table-level locking (such as **MyISAM**, **MEMORY**, and **MERGE**).

For **MyISAM** tables, if you use the **QUICK** modifier, the storage engine does not merge index leaves during delete, which may speed up some kinds of delete operations.

The **IGNORE** modifier causes MySQL to ignore ignorable errors during the process of deleting rows. (Errors encountered during the parsing stage are processed in the usual manner.) Errors that are ignored due to the use of **IGNORE** are returned as warnings. For more information, see [The Effect of IGNORE on Statement Execution](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#ignore-effect-on-execution).

#### Order of Deletion

If the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement includes an **ORDER BY** clause, rows are deleted in the order specified by the clause. This is useful primarily in conjunction with **LIMIT**. For example, the following statement finds rows matching the **WHERE** clause, sorts them by **timestamp\_column**, and deletes the first (oldest) one:

DELETE FROM somelog WHERE user = 'jcole'

ORDER BY timestamp\_column LIMIT 1;

**ORDER BY** also helps to delete rows in an order required to avoid referential integrity violations.

#### InnoDB Tables

If you are deleting many rows from a large table, you may exceed the lock table size for an **InnoDB** table. To avoid this problem, or simply to minimize the time that the table remains locked, the following strategy (which does not use [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) at all) might be helpful:

Select the rows not to be deleted into an empty table that has the same structure as the original table:

INSERT INTO t\_copy SELECT \* FROM t WHERE ... ;

Use [**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table) to atomically move the original table out of the way and rename the copy to the original name:

RENAME TABLE t TO t\_old, t\_copy TO t;

Drop the original table:

DROP TABLE t\_old;

No other sessions can access the tables involved while [**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table) executes, so the rename operation is not subject to concurrency problems. See [Section 13.1.36, “RENAME TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table).

#### MyISAM Tables

In **MyISAM** tables, deleted rows are maintained in a linked list and subsequent [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations reuse old row positions. To reclaim unused space and reduce file sizes, use the [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) statement or the [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) utility to reorganize tables. [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is easier to use, but [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) is faster. See [Section 13.7.3.4, “OPTIMIZE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table), and [Section 4.6.4, “myisamchk — MyISAM Table-Maintenance Utility”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

The **QUICK** modifier affects whether index leaves are merged for delete operations. **DELETE QUICK** is most useful for applications where index values for deleted rows are replaced by similar index values from rows inserted later. In this case, the holes left by deleted values are reused.

**DELETE QUICK** is not useful when deleted values lead to underfilled index blocks spanning a range of index values for which new inserts occur again. In this case, use of **QUICK** can lead to wasted space in the index that remains unreclaimed. Here is an example of such a scenario:

Create a table that contains an indexed **AUTO\_INCREMENT** column.

Insert many rows into the table. Each insert results in an index value that is added to the high end of the index.

Delete a block of rows at the low end of the column range using **DELETE QUICK**.

In this scenario, the index blocks associated with the deleted index values become underfilled but are not merged with other index blocks due to the use of **QUICK**. They remain underfilled when new inserts occur, because new rows do not have index values in the deleted range. Furthermore, they remain underfilled even if you later use [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) without **QUICK**, unless some of the deleted index values happen to lie in index blocks within or adjacent to the underfilled blocks. To reclaim unused index space under these circumstances, use [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table).

If you are going to delete many rows from a table, it might be faster to use **DELETE QUICK** followed by [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table). This rebuilds the index rather than performing many index block merge operations.

#### Multi-Table Deletes

You can specify multiple tables in a [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement to delete rows from one or more tables depending on the condition in the **WHERE** clause. You cannot use **ORDER BY** or **LIMIT** in a multiple-table **DELETE**. The ***table\_references*** clause lists the tables involved in the join, as described in [Section 13.2.10.2, “JOIN Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join).

For the first multiple-table syntax, only matching rows from the tables listed before the **FROM** clause are deleted. For the second multiple-table syntax, only matching rows from the tables listed in the **FROM** clause (before the **USING** clause) are deleted. The effect is that you can delete rows from many tables at the same time and have additional tables that are used only for searching:

DELETE t1, t2 FROM t1 INNER JOIN t2 INNER JOIN t3

WHERE t1.id=t2.id AND t2.id=t3.id;

Or:

DELETE FROM t1, t2 USING t1 INNER JOIN t2 INNER JOIN t3

WHERE t1.id=t2.id AND t2.id=t3.id;

These statements use all three tables when searching for rows to delete, but delete matching rows only from tables **t1** and **t2**.

The preceding examples use **INNER JOIN**, but multiple-table [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements can use other types of join permitted in [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, such as **LEFT JOIN**. For example, to delete rows that exist in **t1** that have no match in **t2**, use a **LEFT JOIN**:

DELETE t1 FROM t1 LEFT JOIN t2 ON t1.id=t2.id WHERE t2.id IS NULL;

The syntax permits **.\*** after each ***tbl\_name*** for compatibility with **Access**.

If you use a multiple-table [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statement involving **InnoDB** tables for which there are foreign key constraints, the MySQL optimizer might process tables in an order that differs from that of their parent/child relationship. In this case, the statement fails and rolls back. Instead, you should delete from a single table and rely on the **ON DELETE** capabilities that **InnoDB** provides to cause the other tables to be modified accordingly.

**Note**

If you declare an alias for a table, you must use the alias when referring to the table:

DELETE t1 FROM test AS t1, test2 WHERE ...

Table aliases in a multiple-table [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) should be declared only in the ***table\_references*** part of the statement. Elsewhere, alias references are permitted but not alias declarations.

Correct:

DELETE a1, a2 FROM t1 AS a1 INNER JOIN t2 AS a2

WHERE a1.id=a2.id;

DELETE FROM a1, a2 USING t1 AS a1 INNER JOIN t2 AS a2

WHERE a1.id=a2.id;

Incorrect:

DELETE t1 AS a1, t2 AS a2 FROM t1 INNER JOIN t2

WHERE a1.id=a2.id;

DELETE FROM t1 AS a1, t2 AS a2 USING t1 INNER JOIN t2

WHERE a1.id=a2.id;

Table aliases are also supported for single-table **DELETE** statements beginning with MySQL 8.0.16. (Bug #89410,Bug #27455809)

### 13.2.3 DO Statement

DO ***expr*** [, ***expr***] ...

[**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) executes the expressions but does not return any results. In most respects, [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) is shorthand for **SELECT *expr*, ...**, but has the advantage that it is slightly faster when you do not care about the result.

[**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) is useful primarily with functions that have side effects, such as [**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock).

Example: This [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement pauses, but also produces a result set:

mysql> **SELECT SLEEP(5);**

+----------+

| SLEEP(5) |

+----------+

| 0 |

+----------+

1 row in set (5.02 sec)

[**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do), on the other hand, pauses without producing a result set.:

mysql> **DO SLEEP(5);**

Query OK, 0 rows affected (4.99 sec)

This could be useful, for example in a stored function or trigger, which prohibit statements that produce result sets.

[**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do) only executes expressions. It cannot be used in all cases where **SELECT** can be used. For example, **DO id FROM t1** is invalid because it references a table.

### 13.2.4 HANDLER Statement

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

The **HANDLER** statement provides direct access to table storage engine interfaces. It is available for **InnoDB** and **MyISAM** tables.

The **HANDLER ... OPEN** statement opens a table, making it accessible using subsequent **HANDLER ... READ** statements. This table object is not shared by other sessions and is not closed until the session calls **HANDLER ... CLOSE** or the session terminates.

If you open the table using an alias, further references to the open table with other **HANDLER** statements must use the alias rather than the table name. If you do not use an alias, but open the table using a table name qualified by the database name, further references must use the unqualified table name. For example, for a table opened using **mydb.mytable**, further references must use **mytable**.

The first **HANDLER ... READ** syntax fetches a row where the index specified satisfies the given values and the **WHERE** condition is met. If you have a multiple-column index, specify the index column values as a comma-separated list. Either specify values for all the columns in the index, or specify values for a leftmost prefix of the index columns. Suppose that an index **my\_idx** includes three columns named **col\_a**, **col\_b**, and **col\_c**, in that order. The **HANDLER** statement can specify values for all three columns in the index, or for the columns in a leftmost prefix. For example:

HANDLER ... READ my\_idx = (col\_a\_val,col\_b\_val,col\_c\_val) ...

HANDLER ... READ my\_idx = (col\_a\_val,col\_b\_val) ...

HANDLER ... READ my\_idx = (col\_a\_val) ...

To employ the **HANDLER** interface to refer to a table's **PRIMARY KEY**, use the quoted identifier **`PRIMARY`**:

HANDLER ***tbl\_name*** READ `PRIMARY` ...

The second **HANDLER ... READ** syntax fetches a row from the table in index order that matches the **WHERE** condition.

The third **HANDLER ... READ** syntax fetches a row from the table in natural row order that matches the **WHERE** condition. It is faster than **HANDLER *tbl\_name* READ *index\_name*** when a full table scan is desired. Natural row order is the order in which rows are stored in a **MyISAM** table data file. This statement works for **InnoDB** tables as well, but there is no such concept because there is no separate data file.

Without a **LIMIT** clause, all forms of **HANDLER ... READ** fetch a single row if one is available. To return a specific number of rows, include a **LIMIT** clause. It has the same syntax as for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. See [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

**HANDLER ... CLOSE** closes a table that was opened with **HANDLER ... OPEN**.

There are several reasons to use the **HANDLER** interface instead of normal [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements:

**HANDLER** is faster than [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select):

A designated storage engine handler object is allocated for the **HANDLER ... OPEN**. The object is reused for subsequent **HANDLER** statements for that table; it need not be reinitialized for each one.

There is less parsing involved.

There is no optimizer or query-checking overhead.

The handler interface does not have to provide a consistent look of the data (for example, [dirty reads](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_read) are permitted), so the storage engine can use optimizations that [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) does not normally permit.

**HANDLER** makes it easier to port to MySQL applications that use a low-level **ISAM**-like interface. (See [Section 15.20, “InnoDB memcached Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-memcached) for an alternative way to adapt applications that use the key-value store paradigm.)

**HANDLER** enables you to traverse a database in a manner that is difficult (or even impossible) to accomplish with [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). The **HANDLER** interface is a more natural way to look at data when working with applications that provide an interactive user interface to the database.

**HANDLER** is a somewhat low-level statement. For example, it does not provide consistency. That is, **HANDLER ... OPEN** does not take a snapshot of the table, and does not lock the table. This means that after a **HANDLER ... OPEN** statement is issued, table data can be modified (by the current session or other sessions) and these modifications might be only partially visible to **HANDLER ... NEXT** or **HANDLER ... PREV** scans.

An open handler can be closed and marked for reopen, in which case the handler loses its position in the table. This occurs when both of the following circumstances are true:

Any session executes [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) or DDL statements on the handler's table.

The session in which the handler is open executes non-**HANDLER** statements that use tables.

[**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) for a table closes all handlers for the table that were opened with [**HANDLER OPEN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler).

If a table is flushed with [**FLUSH TABLES *tbl\_name* WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock-with-list) was opened with **HANDLER**, the handler is implicitly flushed and loses its position.

### 13.2.5 IMPORT TABLE Statement

IMPORT TABLE FROM ***sdi\_file*** [, ***sdi\_file***] ...

The [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) statement imports **MyISAM** tables based on information contained in .sdi (serialized dictionary information) metadata files. [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) requires the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege to read the .sdi and table content files, and the [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create) privilege for the table to be created.

Tables can be exported from one server using [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) to write a file of SQL statements and imported into another server using [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) to process the dump file. [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) provides a faster alternative using the “raw” table files.

Prior to import, the files that provide the table content must be placed in the appropriate schema directory for the import server, and the .sdi file must be located in a directory accessible to the server. For example, the .sdi file can be placed in the directory named by the [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv) system variable, or (if [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv) is empty) in a directory under the server data directory.

The following example describes how to export **MyISAM** tables named **employees** and **managers** from the **hr** schema of one server and import them into the **hr** schema of another server. The example uses these assumptions (to perform a similar operation on your own system, modify the path names as appropriate):

For the export server, ***export\_basedir*** represents its base directory, and its data directory is ***export\_basedir***/data.

For the import server, ***import\_basedir*** represents its base directory, and its data directory is ***import\_basedir***/data.

Table files are exported from the export server into the /tmp/export directory and this directory is secure (not accessible to other users).

The import server uses /tmp/mysql-files as the directory named by its [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv) system variable.

To export tables from the export server, use this procedure:

Ensure a consistent snapshot by executing this statement to lock the tables so that they cannot be modified during export:

mysql> **FLUSH TABLES hr.employees, hr.managers WITH READ LOCK;**

While the lock is in effect, the tables can still be used, but only for read access.

At the file system level, copy the .sdi and table content files from the **hr** schema directory to the secure export directory:

The **.sdi** file is located in the **hr** schema directory, but might not have exactly the same basename as the table name. For example, the .sdi files for the **employees** and **managers** tables might be named employees\_125.sdi and managers\_238.sdi.

For a **MyISAM** table, the content files are its .MYD data file and .MYI index file.

Given those file names, the copy commands look like this:

shell> **cd *export\_basedir*/data/hr**

shell> **cp employees\_125.sdi /tmp/export**

shell> **cp managers\_238.sdi /tmp/export**

shell> **cp employees.{MYD,MYI} /tmp/export**

shell> **cp managers.{MYD,MYI} /tmp/export**

Unlock the tables:

mysql> **UNLOCK TABLES;**

To import tables into the import server, use this procedure:

The import schema must exist. If necessary, execute this statement to create it:

mysql> **CREATE SCHEMA hr;**

At the file system level, copy the .sdi files to the import server [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv) directory, /tmp/mysql-files. Also, copy the table content files to the **hr** schema directory:

shell> **cd /tmp/export**

shell> **cp employees\_125.sdi /tmp/mysql-files**

shell> **cp managers\_238.sdi /tmp/mysql-files**

shell> **cp employees.{MYD,MYI} *import\_basedir*/data/hr**

shell> **cp managers.{MYD,MYI} *import\_basedir*/data/hr**

Import the tables by executing an [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) statement that names the .sdi files:

mysql> **IMPORT TABLE FROM**

**'/tmp/mysql-files/employees.sdi',**

**'/tmp/mysql-files/managers.sdi';**

The .sdi file need not be placed in the import server directory named by the [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv) system variable if that variable is empty; it can be in any directory accessible to the server, including the schema directory for the imported table. If the .sdi file is placed in that directory, however, it may be rewritten; the import operation creates a new .sdi file for the table, which overwrites the old .sdi file if the operation uses the same file name for the new file.

Each ***sdi\_file*** value must be a string literal that names the .sdi file for a table or is a pattern that matches .sdi files. If the string is a pattern, any leading directory path and the .sdi file name suffix must be given literally. Pattern characters are permitted only in the base name part of the file name:

**?** matches any single character

**\*** matches any sequence of characters, including no characters

Using a pattern, the previous [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) statement could have been written like this (assuming that the /tmp/mysql-files directory contains no other .sdi files matching the pattern):

IMPORT TABLE FROM '/tmp/mysql-files/\*.sdi';

To interpret the location of .sdi file path names, the server uses the same rules for [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) as the server-side rules for [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) (that is, the non-**LOCAL** rules). See [Section 13.2.7, “LOAD DATA Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data), paying particular attention to the rules used to interpret relative path names.

[**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) fails if the **.sdi** or table files cannot be located. After importing a table, the server attempts to open it and reports as warnings any problems detected. To attempt a repair to correct any reported issues, use [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table).

[**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) is not written to the binary log.

#### Restrictions and Limitations

[**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) applies only to non-**TEMPORARY** **MyISAM** tables. It does not apply to tables created with a transactional storage engine, tables created with [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), or views.

An .sdi file used in an import operation must be generated on a server with the same data dictionary version and sdi version as the import server. The version information of the generating server is found in the .sdi file:

{

"mysqld\_version\_id":80019,

"dd\_version":80017,

"sdi\_version":80016,

...

}

To determine the data dictionary and sdi version of the import server, you can check the .sdi file of a recently created table on the import server.

The table data and index files must be placed in the schema directory for the import server prior to the import operation, unless the table as defined on the export server uses the **DATA DIRECTORY** or **INDEX DIRECTORY** table options. In that case, modify the import procedure using one of these alternatives before executing the [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) statement:

Put the data and index files into the same directory on the import server host as on the export server host, and create symlinks in the import server schema directory to those files.

Put the data and index files into an import server host directory different from that on the export server host, and create symlinks in the import server schema directory to those files. In addition, modify the .sdi file to reflect the different file locations.

Put the data and index files into the schema directory on the import server host, and modify the .sdi file to remove the data and index directory table options.

Any collation IDs stored in the .sdi file must refer to the same collations on the export and import servers.

Trigger information for a table is not serialized into the table .sdi file, so triggers are not restored by the import operation.

Some edits to an .sdi file are permissible prior to executing the [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) statement, whereas others are problematic or may even cause the import operation to fail:

Changing the data directory and index directory table options is required if the locations of the data and index files differ between the export and import servers.

Changing the schema name is required to import the table into a different schema on the import server than on the export server.

Changing schema and table names may be required to accommodate differences between file system case-sensitivity semantics on the export and import servers or differences in [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) settings. Changing the table names in the .sdi file may require renaming the table files as well.

In some cases, changes to column definitions are permitted. Changing data types is likely to cause problems.

### 13.2.6 INSERT Statement

[13.2.6.1 INSERT ... SELECT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select)

[13.2.6.2 INSERT ... ON DUPLICATE KEY UPDATE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate)

[13.2.6.3 INSERT DELAYED Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-delayed)

INSERT [LOW\_PRIORITY | DELAYED | HIGH\_PRIORITY] [IGNORE]

[INTO] ***tbl\_name***

[PARTITION (***partition\_name*** [, ***partition\_name***] ...)]

[(***col\_name*** [, ***col\_name***] ...)]

{ {VALUES | VALUE} (***value\_list***) [, (***value\_list***)] ...

|

VALUES ***row\_constructor\_list***

}

[AS ***row\_alias***[(***col\_alias*** [, ***col\_alias***] ...)]]

[ON DUPLICATE KEY UPDATE ***assignment\_list***]

INSERT [LOW\_PRIORITY | DELAYED | HIGH\_PRIORITY] [IGNORE]

[INTO] ***tbl\_name***

[PARTITION (***partition\_name*** [, ***partition\_name***] ...)]

[AS ***row\_alias***[(***col\_alias*** [, ***col\_alias***] ...)]]

SET ***assignment\_list***

[ON DUPLICATE KEY UPDATE ***assignment\_list***]

INSERT [LOW\_PRIORITY | HIGH\_PRIORITY] [IGNORE]

[INTO] ***tbl\_name***

[PARTITION (***partition\_name*** [, ***partition\_name***] ...)]

[(***col\_name*** [, ***col\_name***] ...)]

[AS ***row\_alias***[(***col\_alias*** [, ***col\_alias***] ...)]]

{SELECT ... | TABLE ***table\_name***}

[ON DUPLICATE KEY UPDATE ***assignment\_list***]

***value***:

{***expr*** | DEFAULT}

***value\_list***:

***value*** [, ***value***] ...

***row\_constructor\_list***:

ROW(***value\_list***)[, ROW(***value\_list***)][, ...]

***assignment***:

***col\_name*** = [***row\_alias***.]***value***

***assignment\_list***:

***assignment*** [, ***assignment***] ...

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) inserts new rows into an existing table. The [**INSERT ... VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**INSERT ... VALUES ROW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values), and [**INSERT ... SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) forms of the statement insert rows based on explicitly specified values. The [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) form inserts rows selected from another table or tables. You can also use [**INSERT ... TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) in MySQL 8.0.19 and later to insert rows from a single table. [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) with an **ON DUPLICATE KEY UPDATE** clause enables existing rows to be updated if a row to be inserted would cause a duplicate value in a **UNIQUE** index or **PRIMARY KEY**. In MySQL 8.0.19 and later, a row alias with one or more optional column alises can be used with **ON DUPLICATE KEY UPDATE** to refer to the row to be inserted.

For additional information about [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) and [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate), see [Section 13.2.6.1, “INSERT ... SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select), and [Section 13.2.6.2, “INSERT ... ON DUPLICATE KEY UPDATE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate).

In MySQL 8.0, the **DELAYED** keyword is accepted but ignored by the server. For the reasons for this, see [Section 13.2.6.3, “INSERT DELAYED Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-delayed),

Inserting into a table requires the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege for the table. If the **ON DUPLICATE KEY UPDATE** clause is used and a duplicate key causes an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) to be performed instead, the statement requires the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the columns to be updated. For columns that are read but not modified you need only the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege (such as for a column referenced only on the right hand side of an ***col\_name***=***expr*** assignment in an **ON DUPLICATE KEY UPDATE** clause).

When inserting into a partitioned table, you can control which partitions and subpartitions accept new rows. The **PARTITION** clause takes a list of the comma-separated names of one or more partitions or subpartitions (or both) of the table. If any of the rows to be inserted by a given [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement do not match one of the partitions listed, the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement fails with the error Found a row not matching the given partition set. For more information and examples, see [Section 24.5, “Partition Selection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-selection).

***tbl\_name*** is the table into which rows should be inserted. Specify the columns for which the statement provides values as follows:

Provide a parenthesized list of comma-separated column names following the table name. In this case, a value for each named column must be provided by the **VALUES** list, [**VALUES ROW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) list, or [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. For the **INSERT TABLE** form, the number of columns in the source table must match the number of columns to be inserted.

If you do not specify a list of column names for [**INSERT ... VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select), values for every column in the table must be provided by the **VALUES** list, [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, or [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement. If you do not know the order of the columns in the table, use **DESCRIBE *tbl\_name*** to find out.

A **SET** clause indicates columns explicitly by name, together with the value to assign each one.

Column values can be given in several ways:

If strict SQL mode is not enabled, any column not explicitly given a value is set to its default (explicit or implicit) value. For example, if you specify a column list that does not name all the columns in the table, unnamed columns are set to their default values. Default value assignment is described in [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults). See also [Section 1.7.3.3, “Enforced Constraints on Invalid Data”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\introduction.html#constraint-invalid-data).

If strict SQL mode is enabled, an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement generates an error if it does not specify an explicit value for every column that has no default value. See [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

If both the column list and the **VALUES** list are empty, [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) creates a row with each column set to its default value:

INSERT INTO ***tbl\_name*** () VALUES();

If strict mode is not enabled, MySQL uses the implicit default value for any column that has no explicitly defined default. If strict mode is enabled, an error occurs if any column has no default value.

Use the keyword **DEFAULT** to set a column explicitly to its default value. This makes it easier to write [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements that assign values to all but a few columns, because it enables you to avoid writing an incomplete **VALUES** list that does not include a value for each column in the table. Otherwise, you must provide the list of column names corresponding to each value in the **VALUES** list.

If a generated column is inserted into explicitly, the only permitted value is **DEFAULT**. For information about generated columns, see [Section 13.1.20.8, “CREATE TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-generated-columns).

In expressions, you can use [**DEFAULT(*col\_name*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_default) to produce the default value for column ***col\_name***.

Type conversion of an expression ***expr*** that provides a column value might occur if the expression data type does not match the column data type. Conversion of a given value can result in different inserted values depending on the column type. For example, inserting the string **'1999.0e-2'** into an [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types), [**DECIMAL(10,6)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types), or [**YEAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year) column inserts the value **1999**, **19.9921**, **19.992100**, or **1999**, respectively. The value stored in the [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) and [**YEAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year) columns is **1999** because the string-to-number conversion looks only at as much of the initial part of the string as may be considered a valid integer or year. For the [**FLOAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) and [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) columns, the string-to-number conversion considers the entire string a valid numeric value.

An expression ***expr*** can refer to any column that was set earlier in a value list. For example, you can do this because the value for **col2** refers to **col1**, which has previously been assigned:

INSERT INTO ***tbl\_name*** (col1,col2) VALUES(15,col1\*2);

But the following is not legal, because the value for **col1** refers to **col2**, which is assigned after **col1**:

INSERT INTO ***tbl\_name*** (col1,col2) VALUES(col2\*2,15);

An exception occurs for columns that contain **AUTO\_INCREMENT** values. Because **AUTO\_INCREMENT** values are generated after other value assignments, any reference to an **AUTO\_INCREMENT** column in the assignment returns a **0**.

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements that use **VALUES** syntax can insert multiple rows. To do this, include multiple lists of comma-separated column values, with lists enclosed within parentheses and separated by commas. Example:

INSERT INTO ***tbl\_name*** (a,b,c)

VALUES(1,2,3), (4,5,6), (7,8,9);

Each values list must contain exactly as many values as are to be inserted per row. The following statement is invalid because it contains one list of nine values, rather than three lists of three values each:

INSERT INTO ***tbl\_name*** (a,b,c) VALUES(1,2,3,4,5,6,7,8,9);

**VALUE** is a synonym for **VALUES** in this context. Neither implies anything about the number of values lists, nor about the number of values per list. Either may be used whether there is a single values list or multiple lists, and regardless of the number of values per list.

[**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements using [**VALUES ROW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) syntax can also insert multiple rows. In this case, each value list must be contained within a **ROW()** (row constructor), like this:

INSERT INTO ***tbl\_name*** (a,b,c)

VALUES ROW(1,2,3), ROW(4,5,6), ROW(7,8,9);

The affected-rows value for an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) can be obtained using the [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) SQL function or the [**mysql\_affected\_rows()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-affected-rows.html) C API function. See [Section 12.16, “Information Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#information-functions), and [mysql\_affected\_rows()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-affected-rows.html).

If you use [**INSERT ... VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or **INSERT ... VALUES ROW()** with multiple value lists, or [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) or **INSERT ... TABLE**, the statement returns an information string in this format:

Records: ***N1*** Duplicates: ***N2*** Warnings: ***N3***

If you are using the C API, the information string can be obtained by invoking the [**mysql\_info()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-info.html) function. See [mysql\_info()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-info.html).

**Records** indicates the number of rows processed by the statement. (This is not necessarily the number of rows actually inserted because **Duplicates** can be nonzero.) **Duplicates** indicates the number of rows that could not be inserted because they would duplicate some existing unique index value. **Warnings** indicates the number of attempts to insert column values that were problematic in some way. Warnings can occur under any of the following conditions:

Inserting **NULL** into a column that has been declared **NOT NULL**. For multiple-row [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements or [**INSERT INTO ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) statements, the column is set to the implicit default value for the column data type. This is **0** for numeric types, the empty string (**''**) for string types, and the “zero” value for date and time types. [**INSERT INTO ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) statements are handled the same way as multiple-row inserts because the server does not examine the result set from the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) to see whether it returns a single row. (For a single-row [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), no warning occurs when **NULL** is inserted into a **NOT NULL** column. Instead, the statement fails with an error.)

Setting a numeric column to a value that lies outside the column range. The value is clipped to the closest endpoint of the range.

Assigning a value such as **'10.34 a'** to a numeric column. The trailing nonnumeric text is stripped off and the remaining numeric part is inserted. If the string value has no leading numeric part, the column is set to **0**.

Inserting a string into a string column ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), or [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)) that exceeds the column maximum length. The value is truncated to the column maximum length.

Inserting a value into a date or time column that is illegal for the data type. The column is set to the appropriate zero value for the type.

For [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) examples involving **AUTO\_INCREMENT** column values, see [Section 3.6.9, “Using AUTO\_INCREMENT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\tutorial.html#example-auto-increment).

If [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) inserts a row into a table that has an **AUTO\_INCREMENT** column, you can find the value used for that column by using the [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) SQL function or the [**mysql\_insert\_id()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-insert-id.html) C API function.

**Note**

These two functions do not always behave identically. The behavior of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements with respect to **AUTO\_INCREMENT** columns is discussed further in [Section 12.16, “Information Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#information-functions), and [mysql\_insert\_id()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-insert-id.html).

The [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement supports the following modifiers:

If you use the **LOW\_PRIORITY** modifier, execution of the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) is delayed until no other clients are reading from the table. This includes other clients that began reading while existing clients are reading, and while the **INSERT LOW\_PRIORITY** statement is waiting. It is possible, therefore, for a client that issues an **INSERT LOW\_PRIORITY** statement to wait for a very long time.

**LOW\_PRIORITY** affects only storage engines that use only table-level locking (such as **MyISAM**, **MEMORY**, and **MERGE**).

**Note**

**LOW\_PRIORITY** should normally not be used with **MyISAM** tables because doing so disables concurrent inserts. See [Section 8.11.3, “Concurrent Inserts”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#concurrent-inserts).

If you specify **HIGH\_PRIORITY**, it overrides the effect of the [--low-priority-updates](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_low_priority_updates) option if the server was started with that option. It also causes concurrent inserts not to be used. See [Section 8.11.3, “Concurrent Inserts”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#concurrent-inserts).

**HIGH\_PRIORITY** affects only storage engines that use only table-level locking (such as **MyISAM**, **MEMORY**, and **MERGE**).

If you use the **IGNORE** modifier, ignorable errors that occur while executing the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement are ignored. For example, without **IGNORE**, a row that duplicates an existing **UNIQUE** index or **PRIMARY KEY** value in the table causes a duplicate-key error and the statement is aborted. With **IGNORE**, the row is discarded and no error occurs. Ignored errors generate warnings instead.

**IGNORE** has a similar effect on inserts into partitioned tables where no partition matching a given value is found. Without **IGNORE**, such [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements are aborted with an error. When [**INSERT IGNORE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) is used, the insert operation fails silently for rows containing the unmatched value, but inserts rows that are matched. For an example, see [Section 24.2.2, “LIST Partitioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-list).

Data conversions that would trigger errors abort the statement if **IGNORE** is not specified. With **IGNORE**, invalid values are adjusted to the closest values and inserted; warnings are produced but the statement does not abort. You can determine with the [**mysql\_info()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-info.html) C API function how many rows were actually inserted into the table.

For more information, see [The Effect of IGNORE on Statement Execution](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#ignore-effect-on-execution).

You can use [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) instead of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) to overwrite old rows. [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) is the counterpart to [**INSERT IGNORE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) in the treatment of new rows that contain unique key values that duplicate old rows: The new rows replace the old rows rather than being discarded. See [Section 13.2.9, “REPLACE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace).

If you specify **ON DUPLICATE KEY UPDATE**, and a row is inserted that would cause a duplicate value in a **UNIQUE** index or **PRIMARY KEY**, an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) of the old row occurs. The affected-rows value per row is 1 if the row is inserted as a new row, 2 if an existing row is updated, and 0 if an existing row is set to its current values. If you specify the **CLIENT\_FOUND\_ROWS** flag to the [**mysql\_real\_connect()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-connect.html) C API function when connecting to [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld), the affected-rows value is 1 (not 0) if an existing row is set to its current values. See [Section 13.2.6.2, “INSERT ... ON DUPLICATE KEY UPDATE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate).

**[INSERT DELAYED](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "insert-delayed" \o "13.2.6.3 INSERT DELAYED Statement)** was deprecated in MySQL 5.6, and is scheduled for eventual removal. In MySQL 8.0, the **DELAYED** modifier is accepted but ignored. Use **INSERT** (without **DELAYED**) instead. See [Section 13.2.6.3, “INSERT DELAYED Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-delayed).

#### 13.2.6.1 INSERT ... SELECT Statement

INSERT [LOW\_PRIORITY | HIGH\_PRIORITY] [IGNORE]

[INTO] ***tbl\_name***

[PARTITION (***partition\_name*** [, ***partition\_name***] ...)]

[(***col\_name*** [, ***col\_name***] ...)]

{SELECT ... | TABLE ***table\_name***}

[ON DUPLICATE KEY UPDATE ***assignment\_list***]

***value***:

{***expr*** | DEFAULT}

***assignment***:

***col\_name*** = ***value***

***assignment\_list***:

***assignment*** [, ***assignment***] ...

With [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select), you can quickly insert many rows into a table from the result of a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, which can select from one or many 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;

Beginning with MySQL 8.0.19, you can use a [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement in place of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), as shown here:

INSERT INTO ta TABLE tb;

**TABLE tb** is equivalent to **SELECT \* FROM tb**. It can be useful when inserting all columns from the source table into the target table, and no filtering with WHERE is required. In addition, the rows from [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) can be ordered by one or more columns using **ORDER BY**, and the number of rows inserted can be limited using a **LIMIT** clause. For more information, see [Section 13.2.12, “TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table).

The following conditions hold for [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) statements, and, except where noted, for **INSERT ... TABLE** as well:

Specify **IGNORE** to ignore rows that would cause duplicate-key violations.

The target table of the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement may appear in the **FROM** clause of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part of the query, or as the table named by [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table). However, you cannot insert into a table and select from the same table in a subquery.

When selecting from and inserting into the same table, MySQL creates an internal temporary table to hold the rows from the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and then inserts those rows into the target table. However, you cannot use **INSERT INTO t ... SELECT ... FROM t** when **t** is a **TEMPORARY** table, because **TEMPORARY** tables cannot be referred to twice in the same statement. For the same reason, you cannot use **INSERT INTO t ... TABLE t** when **t** is a temporary table. See [Section 8.4.4, “Internal Temporary Table Use in MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#internal-temporary-tables), and [Section B.3.6.2, “TEMPORARY Table Problems”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#temporary-table-problems).

**AUTO\_INCREMENT** columns work as usual.

To ensure that the binary log can be used to re-create the original tables, MySQL does not permit concurrent inserts for [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) or **INSERT ... TABLE** statements (see [Section 8.11.3, “Concurrent Inserts”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#concurrent-inserts)).

To avoid ambiguous column reference problems when the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) refer to the same table, provide a unique alias for each table used in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part, and qualify column names in that part with the appropriate alias.

The [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement does not support aliases.

You can explicitly select which partitions or subpartitions (or both) of the source or target table (or both) are to be used with a **PARTITION** clause following the name of the table. When **PARTITION** is used with the name of the source table in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) portion of the statement, rows are selected only from the partitions or subpartitions named in its partition list. When **PARTITION** is used with the name of the target table for the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) portion of the statement, it must be possible to insert all rows selected into the partitions or subpartitions named in the partition list following the option. Otherwise, the **INSERT ... SELECT** statement fails. For more information and examples, see [Section 24.5, “Partition Selection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-selection).

[**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) does not support a **PARTITION** clause.

For [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statements, see [Section 13.2.6.2, “INSERT ... ON DUPLICATE KEY UPDATE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) for conditions under which the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) columns can be referred to in an **ON DUPLICATE KEY UPDATE** clause. This also works for **INSERT ... TABLE**.

The order in which a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement with no **ORDER BY** clause returns rows is nondeterministic. This means that, when using replication, there is no guarantee that such a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) returns rows in the same order on the source and the replica, which can lead to inconsistencies between them. To prevent this from occurring, always write **INSERT ... SELECT** or **INSERT ... TABLE** statements that are to be replicated using an **ORDER BY** clause that produces the same row order on the source and the replica. See also [Section 17.5.1.18, “Replication and LIMIT”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-limit).

Due to this issue, [**INSERT ... SELECT ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) and [**INSERT IGNORE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) statements are flagged as unsafe for statement-based replication. Such statements produce a warning in the error log when using statement-based mode and are written to the binary log using the row-based format when using **MIXED** mode. (Bug #11758262, Bug #50439)

See also [Section 17.2.1.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr).

#### 13.2.6.2 INSERT ... ON DUPLICATE KEY UPDATE Statement

If you specify an **ON DUPLICATE KEY UPDATE** clause and a row to be inserted would cause a duplicate value in a **UNIQUE** index or **PRIMARY KEY**, an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) of the old row occurs. For example, if column **a** is declared as **UNIQUE** and contains the value **1**, the following two statements have similar effect:

INSERT INTO t1 (a,b,c) VALUES (1,2,3)

ON DUPLICATE KEY UPDATE c=c+1;

UPDATE t1 SET c=c+1 WHERE a=1;

The effects are not quite identical: For an **InnoDB** table where **a** is an auto-increment column, the **INSERT** statement increases the auto-increment value but the **UPDATE** does not.

If column **b** is also unique, the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) is equivalent to this [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement instead:

UPDATE t1 SET c=c+1 WHERE a=1 OR b=2 LIMIT 1;

If **a=1 OR b=2** matches several rows, only one row is updated. In general, you should try to avoid using an **ON DUPLICATE KEY UPDATE** clause on tables with multiple unique indexes.

With **ON DUPLICATE KEY UPDATE**, the affected-rows value per row is 1 if the row is inserted as a new row, 2 if an existing row is updated, and 0 if an existing row is set to its current values. If you specify the **CLIENT\_FOUND\_ROWS** flag to the [**mysql\_real\_connect()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-connect.html) C API function when connecting to [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld), the affected-rows value is 1 (not 0) if an existing row is set to its current values.

If a table contains an **AUTO\_INCREMENT** column and [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) inserts or updates a row, the [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) function returns the **AUTO\_INCREMENT** value.

The **ON DUPLICATE KEY UPDATE** clause can contain multiple column assignments, separated by commas.

In assignment value expressions in the **ON DUPLICATE KEY UPDATE** clause, you can use the [**VALUES(*col\_name*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) function to refer to column values from the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) portion of the [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statement. In other words, [**VALUES(*col\_name*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) in the **ON DUPLICATE KEY 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()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) function is meaningful only in the **ON DUPLICATE KEY UPDATE** clause or [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements and returns **NULL** otherwise. Example:

INSERT INTO t1 (a,b,c) VALUES (1,2,3),(4,5,6)

ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);

That statement is identical to the following two statements:

INSERT INTO t1 (a,b,c) VALUES (1,2,3)

ON DUPLICATE KEY UPDATE c=3;

INSERT INTO t1 (a,b,c) VALUES (4,5,6)

ON DUPLICATE KEY UPDATE c=9;

**Note**

The use of [**VALUES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) to refer to the new row and columns is deprecated beginning with MySQL 8.0.20, and is subject to removal in a future version of MySQL. Instead, use row and column aliases, as described in the next few paragraphs of this section.

Beginning with MySQL 8.0.19, it is possible to use an alias for the row, with, optionally, one or more of its columns to be inserted, following the **VALUES** or **SET** clause, and preceded by the **AS** keyword. Using the row alias **new**, the statement shown previously using **VALUES()** to access the new column values can be written in the form shown here:

INSERT INTO t1 (a,b,c) VALUES (1,2,3),(4,5,6) AS new

ON DUPLICATE KEY UPDATE c = new.a+new.b;

If, in addition, you use the column aliases **m**, **n**, and **p**, you can omit the row alias in the assignment clause and write the same statement like this:

INSERT INTO t1 (a,b,c) VALUES (1,2,3),(4,5,6) AS new(m,n,p)

ON DUPLICATE KEY UPDATE c = m+n;

When using column aliases in this fashion, you must still use a row alias following the **VALUES** clause, even if you do not make direct use of it in the assignment clause.

You can also use row and column aliases with a **SET** clause, as mentioned previously. Employing **SET** instead of **VALUES** in the two **INSERT ... ON DUPLICATE KEY UPDATE** statements just shown can be done as shown here:

INSERT INTO t1 SET a=1,b=2,c=3 AS new

ON DUPLICATE KEY UPDATE c = new.a+new.b;

INSERT INTO t1 SET a=1,b=2,c=3 AS new(m,n,p)

ON DUPLICATE KEY UPDATE c = m+n;

The row alias must not be the same as the name of the table. If column aliases are not used, or if they are the same as the column names, they must be distinguished using the row alias in the **ON DUPLICATE KEY UPDATE** clause. Column aliases must be unique with regard to the row alias to which they apply (that is, no column aliases referring to columns of the same row may be the same).

For [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statements, these rules apply regarding acceptable forms of **SELECT** query expressions that you can refer to in an **ON DUPLICATE KEY UPDATE** clause:

References to columns from queries on a single table, which may be a derived table.

References to columns from queries on a join over multiple tables.

References to columns from **DISTINCT** queries.

References to columns in other tables, as long as the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) does not use **GROUP BY**. One side effect is that you must qualify references to nonunique column names.

References to columns from a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) are not supported. To work around this restriction, rewrite the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) as a derived table so that its rows can be treated as a single-table result set. For example, this statement produces an error:

INSERT INTO t1 (a, b)

SELECT c, d FROM t2

UNION

SELECT e, f FROM t3

ON DUPLICATE KEY UPDATE b = b + c;

Instead, use an equivalent statement that rewrites the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) as a derived table:

INSERT INTO t1 (a, b)

SELECT \* FROM

(SELECT c, d FROM t2

UNION

SELECT e, f FROM t3) AS dt

ON DUPLICATE KEY UPDATE b = b + c;

The technique of rewriting a query as a derived table also enables references to columns from **GROUP BY** queries.

Because the results of [**INSERT ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select) statements depend on the ordering of rows from the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and this order cannot always be guaranteed, it is possible when logging [**INSERT ... SELECT ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statements for the source and the replica to diverge. Thus, [**INSERT ... SELECT ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statements are flagged as unsafe for statement-based replication. Such statements produce a warning in the error log when using statement-based mode and are written to the binary log using the row-based format when using **MIXED** mode. An [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate) statement against a table having more than one unique or primary key is also marked as unsafe. (Bug #11765650, Bug #58637)

See also [Section 17.2.1.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr).

#### 13.2.6.3 INSERT DELAYED Statement

INSERT DELAYED ...

The **DELAYED** option for the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement is a MySQL extension to standard SQL. In previous versions of MySQL, it can be used for certain kinds of tables (such as **MyISAM**), such that when a client uses [**INSERT DELAYED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-delayed), it gets an okay from the server at once, and the row is queued to be inserted when the table is not in use by any other thread.

**DELAYED** inserts and replaces were deprecated in MySQL 5.6. In MySQL 8.0, **DELAYED** is not supported. The server recognizes but ignores the **DELAYED** keyword, handles the insert as a nondelayed insert, and generates an **ER\_WARN\_LEGACY\_SYNTAX\_CONVERTED** warning: INSERT DELAYED is no longer supported. The statement was converted to INSERT. The **DELAYED** keyword is scheduled for removal in a future release.

### 13.2.7 LOAD DATA Statement

LOAD DATA

[LOW\_PRIORITY | CONCURRENT] [LOCAL]

INFILE '***file\_name***'

[REPLACE | IGNORE]

INTO TABLE ***tbl\_name***

[PARTITION (***partition\_name*** [, ***partition\_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 | ROWS}]

[(***col\_name\_or\_user\_var***

[, ***col\_name\_or\_user\_var***] ...)]

[SET ***col\_name***={***expr*** | DEFAULT}

[, ***col\_name***={***expr*** | DEFAULT}] ...]

The [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement reads rows from a text file into a table at a very high speed. The file can be read from the server host or the client host, depending on whether the **LOCAL** modifier is given. **LOCAL** also affects data interpretation and error handling.

[**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) is the complement of [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into). (See [Section 13.2.10.1, “SELECT ... INTO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into).) To write data from a table to a file, use [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into). To read the file back into a table, use [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data). The syntax of the **FIELDS** and **LINES** clauses is the same for both statements.

The [**mysqlimport**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlimport) utility provides another way to load data files; it operates by sending a [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement to the server. See [Section 4.5.5, “mysqlimport — A Data Import Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlimport).

For information about the efficiency of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) versus [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) and speeding up [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data), see [Section 8.2.5.1, “Optimizing INSERT Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#insert-optimization).

[Non-LOCAL Versus LOCAL Operation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-local)

[Input File Character Set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-character-set)

[Input File Location](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-file-location)

[Security Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-security-requirements)

[Duplicate-Key and Error Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-error-handling)

[Index Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-index-handling)

[Field and Line Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-field-line-handling)

[Column List Specification](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-column-list)

[Input Preprocessing](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-input-preprocessing)

[Column Value Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-column-assignments)

[Partitioned Table Support](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-partitioning-support)

[Concurrency Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-concurrency)

[Statement Result Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-statement-result-information)

[Replication Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-replication)

[Miscellaneous Topics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-miscellaneous)

#### Non-LOCAL Versus LOCAL Operation

The **LOCAL** modifier affects these aspects of [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data), compared to non-**LOCAL** operation:

It changes the expected location of the input file; see [Input File Location](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-file-location).

It changes the statement security requirements; see [Security Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-security-requirements).

It has the same effect as the **IGNORE** modifier on the interpretation of input file contents and error handling; see [Duplicate-Key and Error Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-error-handling), and [Column Value Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-column-assignments).

**LOCAL** works only if the server and your client both have been configured to permit it. For example, if [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) was started with the [**local\_infile**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_local_infile) system variable disabled, **LOCAL** produces an error. See [Section 6.1.6, “Security Considerations for LOAD DATA LOCAL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#load-data-local-security).

#### Input File Character Set

The file name must be given as a literal string. On Windows, specify backslashes in path names as forward slashes or doubled backslashes. The server interprets the file name using the character set indicated by the [**character\_set\_filesystem**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_filesystem) system variable.

By default, the server interprets the file contents using the character set indicated by the [**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database) system variable. If the file contents use a character set different from this default, it is a good idea to specify that character set by using the **CHARACTER SET** clause. A character set of **binary** specifies “no conversion.”

[**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) and the setting of [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) do not affect interpretation of file contents.

[**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) interprets all fields in the file as having the same character set, regardless of the data types of the columns into which field values are loaded. For proper interpretation of the file, you must ensure that it was written with the correct character set. For example, if you write a data file with [**mysqldump -T**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) or by issuing a [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) statement in [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql), be sure to use a [--default-character-set](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_default-character-set) option to write output in the character set to be used when the file is loaded with [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data).

**Note**

It is not possible to load data files that use the **ucs2**, **utf16**, **utf16le**, or **utf32** character set.

#### Input File Location

These rules determine the [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) input file location:

If **LOCAL** is not specified, the file must be located on the server host. The server reads the file directly, locating it as follows:

If the file name is an absolute path name, the server uses it as given.

If the file name is a relative path name with leading components, the server looks for the file relative to its data directory.

If the file name has no leading components, the server looks for the file in the database directory of the default database.

If **LOCAL** is specified, the file must be located on the client host. The client program reads the file, locating it as follows:

If the file name is an absolute path name, the client program uses it as given.

If the file name is a relative path name, the client program looks for the file relative to its invocation directory.

When **LOCAL** is used, the client program reads the file and sends its contents to the server. The server creates a copy of the file in the directory where it stores temporary files. See [Section B.3.3.5, “Where MySQL Stores Temporary Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#temporary-files). Lack of sufficient space for the copy in this directory can cause the [**LOAD DATA LOCAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement to fail.

The non-**LOCAL** rules mean that the server reads a file named as ./myfile.txt relative to its data directory, whereas it reads a file named as myfile.txt from the database directory of the default database. For example, if the following [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement is executed while **db1** is the default database, the server reads the file data.txt from the database directory for **db1**, even though the statement explicitly loads the file into a table in the **db2** database:

LOAD DATA INFILE 'data.txt' INTO TABLE db2.my\_table;

**Note**

The server also uses the non-**LOCAL** rules to locate .sdi files for the [**IMPORT TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#import-table) statement.

#### Security Requirements

For a non-**LOCAL** load operation, the server reads a text file located on the server host, so these security requirements must be satisified:

You must have the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege. See [Section 6.2.2, “Privileges Provided by MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#privileges-provided).

The operation is subject to the [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv) system variable setting:

If the variable value is a nonempty directory name, the file must be located in that directory.

If the variable value is empty (which is insecure), the file need only be readable by the server.

For a **LOCAL** load operation, the client program reads a text file located on the client host. Because the file contents are sent over the connection by the client to the server, using **LOCAL** is a bit slower than when the server accesses the file directly. On the other hand, you do not need the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege, and the file can be located in any directory the client program can access.

#### Duplicate-Key and Error Handling

The **REPLACE** and **IGNORE** modifiers control handling of new (input) rows that duplicate existing table rows on unique key values (**PRIMARY KEY** or **UNIQUE** index values):

With **REPLACE**, new rows that have the same value as a unique key value in an existing row replace the existing row. See [Section 13.2.9, “REPLACE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace).

With **IGNORE**, new rows that duplicate an existing row on a unique key value are discarded. For more information, see [The Effect of IGNORE on Statement Execution](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#ignore-effect-on-execution).

The **LOCAL** modifier has the same effect as **IGNORE**. This occurs because the server has no way to stop transmission of the file in the middle of the operation.

If none of **REPLACE**, **IGNORE**, or **LOCAL** is specified, an error occurs when a duplicate key value is found, and the rest of the text file is ignored.

In addition to affecting duplicate-key handling as just described, **IGNORE** and **LOCAL** also affect error handling:

With neither **IGNORE** nor **LOCAL**, data-interpretation errors terminate the operation.

With **IGNORE** or **LOCAL**, data-interpretation errors become warnings and the load operation continues, even if the SQL mode is restrictive. For examples, see [Column Value Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-column-assignments).

#### Index Handling

To ignore foreign key constraints during the load operation, execute a **SET foreign\_key\_checks = 0** statement before executing [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data).

If you use [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) on an empty **MyISAM** table, all nonunique indexes are created in a separate batch (as for [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table)). Normally, this makes [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) much faster when you have many indexes. In some extreme cases, you can create the indexes even faster by turning them off with [**ALTER TABLE ... DISABLE KEYS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) before loading the file into the table and re-creating the indexes with [**ALTER TABLE ... ENABLE KEYS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) after loading the file. See [Section 8.2.5.1, “Optimizing INSERT Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#insert-optimization).

#### Field and Line Handling

For both the [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) and [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) statements, the syntax of the **FIELDS** and **LINES** clauses is the same. Both clauses are optional, but **FIELDS** must precede **LINES** if both are specified.

If you specify a **FIELDS** clause, each of its subclauses (**TERMINATED BY**, **[OPTIONALLY] ENCLOSED BY**, and **ESCAPED BY**) is also optional, except that you must specify at least one of them. Arguments to these clauses are permitted to contain only ASCII characters.

If you specify no **FIELDS** or **LINES** clause, the defaults are the same as if you had written this:

FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'

LINES TERMINATED BY '\n' STARTING BY ''

Backslash is the MySQL escape character within strings in SQL statements. Thus, to specify a literal backslash, you must specify two backslashes for the value to be interpreted as a single backslash. The escape sequences **'\t'** and **'\n'** specify tab and newline characters, respectively.

In other words, the defaults cause [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) to act as follows when reading input:

Look for line boundaries at newlines.

Do not skip any line prefix.

Break lines into fields at tabs.

Do not expect fields to be enclosed within any quoting characters.

Interpret characters preceded by the escape character **\** as escape sequences. For example, **\t**, **\n**, and **\\** signify tab, newline, and backslash, respectively. See the discussion of **FIELDS ESCAPED BY** later for the full list of escape sequences.

Conversely, the defaults cause [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) to act as follows when writing output:

Write tabs between fields.

Do not enclose fields within any quoting characters.

Use **\** to escape instances of tab, newline, or **\** that occur within field values.

Write newlines at the ends of lines.

**Note**

For a text file generated on a Windows system, proper file reading might require **LINES TERMINATED BY '\r\n'** because Windows programs typically use two characters as a line terminator. Some programs, such as **WordPad**, might use **\r** as a line terminator when writing files. To read such files, use **LINES TERMINATED BY '\r'**.

If all the input lines have a common prefix that you want to ignore, you can use **LINES STARTING BY '*prefix\_string*'** to skip the prefix and anything before it. If a line does not include the prefix, the entire line is skipped. Suppose that you issue the following statement:

LOAD DATA INFILE '/tmp/test.txt' INTO TABLE test

FIELDS TERMINATED BY ',' LINES STARTING BY 'xxx';

If the data file looks like this:

xxx"abc",1

something xxx"def",2

"ghi",3

The resulting rows are **("abc",1)** and **("def",2)**. The third row in the file is skipped because it does not contain the prefix.

The **IGNORE *number* LINES** clause can be used to ignore lines at the start of the file. For example, you can use **IGNORE 1 LINES** to skip an initial header line containing column names:

LOAD DATA INFILE '/tmp/test.txt' INTO TABLE test IGNORE 1 LINES;

When you use [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) in tandem with [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) to write data from a database into a file and then read the file back into the database later, the field- and line-handling options for both statements must match. Otherwise, [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) does not interpret the contents of the file properly. Suppose that you use [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) to write a file with fields delimited by commas:

SELECT \* INTO OUTFILE 'data.txt'

FIELDS TERMINATED BY ','

FROM table2;

To read the comma-delimited file, the correct statement is:

LOAD DATA INFILE 'data.txt' INTO TABLE table2

FIELDS TERMINATED BY ',';

If instead you tried to read the file with the statement shown following, it would not work because it instructs [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) to look for tabs between fields:

LOAD DATA INFILE 'data.txt' INTO TABLE table2

FIELDS TERMINATED BY '\t';

The likely result is that each input line would be interpreted as a single field.

**[LOAD DATA](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "load-data" \o "13.2.7 LOAD DATA Statement)** can be used to read files obtained from external sources. For example, many programs can export data in comma-separated values (CSV) format, such that lines have fields separated by commas and enclosed within double quotation marks, with an initial line of column names. If the lines in such a file are terminated by carriage return/newline pairs, the statement shown here illustrates the field- and line-handling options you would use to load the file:

LOAD DATA INFILE 'data.txt' INTO TABLE ***tbl\_name***

FIELDS TERMINATED BY ',' ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 LINES;

If the input values are not necessarily enclosed within quotation marks, use **OPTIONALLY** before the **ENCLOSED BY** option.

Any of the field- or line-handling options can specify an empty string (**''**). If not empty, the **FIELDS [OPTIONALLY] ENCLOSED BY** and **FIELDS ESCAPED BY** values must be a single character. The **FIELDS TERMINATED BY**, **LINES STARTING BY**, and **LINES TERMINATED BY** values can be more than one character. For example, to write lines that are terminated by carriage return/linefeed pairs, or to read a file containing such lines, specify a **LINES TERMINATED BY '\r\n'** clause.

To read a file containing jokes that are separated by lines consisting of **%%**, you can do this

CREATE TABLE jokes

(a INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

joke TEXT NOT NULL);

LOAD DATA INFILE '/tmp/jokes.txt' INTO TABLE jokes

FIELDS TERMINATED BY ''

LINES TERMINATED BY '\n%%\n' (joke);

**FIELDS [OPTIONALLY] ENCLOSED BY** controls quoting of fields. For output ([**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into)), if you omit the word **OPTIONALLY**, all fields are enclosed by the **ENCLOSED BY** character. An example of such output (using a comma as the field delimiter) is shown here:

"1","a string","100.20"

"2","a string containing a , comma","102.20"

"3","a string containing a \" quote","102.20"

"4","a string containing a \", quote and comma","102.20"

If you specify **OPTIONALLY**, the **ENCLOSED BY** character is used only to enclose values from columns that have a string data type (such as [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), or [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum)):

1,"a string",100.20

2,"a string containing a , comma",102.20

3,"a string containing a \" quote",102.20

4,"a string containing a \", quote and comma",102.20

Occurrences of the **ENCLOSED BY** character within a field value are escaped by prefixing them with the **ESCAPED BY** character. Also, if you specify an empty **ESCAPED BY** value, it is possible to inadvertently generate output that cannot be read properly by [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data). For example, the preceding output just shown would appear as follows if the escape character is empty. Observe that the second field in the fourth line contains a comma following the quote, which (erroneously) appears to terminate the field:

1,"a string",100.20

2,"a string containing a , comma",102.20

3,"a string containing a " quote",102.20

4,"a string containing a ", quote and comma",102.20

For input, the **ENCLOSED BY** character, if present, is stripped from the ends of field values. (This is true regardless of whether **OPTIONALLY** is specified; **OPTIONALLY** has no effect on input interpretation.) Occurrences of the **ENCLOSED BY** character preceded by the **ESCAPED BY** character are interpreted as part of the current field value.

If the field begins with the **ENCLOSED BY** character, instances of that character are recognized as terminating a field value only if followed by the field or line **TERMINATED BY** sequence. To avoid ambiguity, occurrences of the **ENCLOSED BY** character within a field value can be doubled and are interpreted as a single instance of the character. For example, if **ENCLOSED BY '"'** is specified, quotation marks are handled as shown here:

"The ""BIG"" boss" -> The "BIG" boss

The "BIG" boss -> The "BIG" boss

The ""BIG"" boss -> The ""BIG"" boss

**FIELDS ESCAPED BY** controls how to read or write special characters:

For input, if the **FIELDS ESCAPED BY** character is not empty, occurrences of that character are stripped and the following character is taken literally as part of a field value. Some two-character sequences that are exceptions, where the first character is the escape character. These sequences are shown in the following table (using **\** for the escape character). The rules for **NULL** handling are described later in this section.

| **Character** | **Escape Sequence** |
| --- | --- |
| **\0** | An ASCII NUL (**X'00'**) character |
| **\b** | A backspace character |
| **\n** | A newline (linefeed) character |
| **\r** | A carriage return character |
| **\t** | A tab character. |
| **\Z** | ASCII 26 (Control+Z) |
| **\N** | NULL |

For more information about **\**-escape syntax, see [Section 9.1.1, “String Literals”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#string-literals).

If the **FIELDS ESCAPED BY** character is empty, escape-sequence interpretation does not occur.

For output, if the **FIELDS ESCAPED BY** character is not empty, it is used to prefix the following characters on output:

The **FIELDS ESCAPED BY** character.

The **FIELDS [OPTIONALLY] ENCLOSED BY** character.

The first character of the **FIELDS TERMINATED BY** and **LINES TERMINATED BY** values, if the **ENCLOSED BY** character is empty or unspecified.

ASCII **0** (what is actually written following the escape character is ASCII **0**, not a zero-valued byte).

If the **FIELDS ESCAPED BY** character is empty, no characters are escaped and **NULL** is output as **NULL**, not **\N**. It is probably not a good idea to specify an empty escape character, particularly if field values in your data contain any of the characters in the list just given.

In certain cases, field- and line-handling options interact:

If **LINES TERMINATED BY** is an empty string and **FIELDS TERMINATED BY** is nonempty, lines are also terminated with **FIELDS TERMINATED BY**.

If the **FIELDS TERMINATED BY** and **FIELDS ENCLOSED BY** values are both empty (**''**), a fixed-row (nondelimited) format is used. With fixed-row format, no delimiters are used between fields (but you can still have a line terminator). Instead, column values are read and written using a field width wide enough to hold all values in the field. For [**TINYINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**SMALLINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**MEDIUMINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), [**INT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), and [**BIGINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types), the field widths are 4, 6, 8, 11, and 20, respectively, no matter what the declared display width is.

**LINES TERMINATED BY** is still used to separate lines. If a line does not contain all fields, the rest of the columns are set to their default values. If you do not have a line terminator, you should set this to **''**. In this case, the text file must contain all fields for each row.

Fixed-row format also affects handling of **NULL** values, as described later.

**Note**

Fixed-size format does not work if you are using a multibyte character set.

Handling of **NULL** values varies according to the **FIELDS** and **LINES** options in use:

For the default **FIELDS** and **LINES** values, **NULL** is written as a field value of **\N** for output, and a field value of **\N** is read as **NULL** for input (assuming that the **ESCAPED BY** character is **\**).

If **FIELDS ENCLOSED BY** is not empty, a field containing the literal word **NULL** as its value is read as a **NULL** value. This differs from the word **NULL** enclosed within **FIELDS ENCLOSED BY** characters, which is read as the string **'NULL'**.

If **FIELDS ESCAPED BY** is empty, **NULL** is written as the word **NULL**.

With fixed-row format (which is used when **FIELDS TERMINATED BY** and **FIELDS ENCLOSED BY** are both empty), **NULL** is written as an empty string. This causes both **NULL** values and empty strings in the table to be indistinguishable when written to the file because both are written as empty strings. If you need to be able to tell the two apart when reading the file back in, you should not use fixed-row format.

An attempt to load **NULL** into a **NOT NULL** column produces either a warning or an error according to the rules described in [Column Value Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data-column-assignments).

Some cases are not supported by [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data):

Fixed-size rows (**FIELDS TERMINATED BY** and **FIELDS ENCLOSED BY** both empty) and [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns.

If you specify one separator that is the same as or a prefix of another, [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) cannot interpret the input properly. For example, the following **FIELDS** clause would cause problems:

FIELDS TERMINATED BY '"' ENCLOSED BY '"'

If **FIELDS ESCAPED BY** is empty, a field value that contains an occurrence of **FIELDS ENCLOSED BY** or **LINES TERMINATED BY** followed by the **FIELDS TERMINATED BY** value causes [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) to stop reading a field or line too early. This happens because [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) cannot properly determine where the field or line value ends.

#### Column List Specification

The following example loads all columns of the **persondata** table:

LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata;

By default, when no column list is provided at the end of the [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement, input lines are expected to contain a field for each table column. If you want to load only some of a table's columns, specify a column list:

LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata

(***col\_name\_or\_user\_var*** [, ***col\_name\_or\_user\_var***] ...);

You must also specify a column list if the order of the fields in the input file differs from the order of the columns in the table. Otherwise, MySQL cannot tell how to match input fields with table columns.

#### Input Preprocessing

Each instance of ***col\_name\_or\_user\_var*** in [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) syntax is either a column name or a user variable. With user variables, the **SET** clause enables you to perform preprocessing transformations on their values before assigning the result to columns.

User variables in the **SET** clause can be used in several ways. The following example uses the first input column directly for the value of **t1.column1**, and assigns the second input column to a user variable that is subjected to a division operation before being used for the value of **t1.column2**:

LOAD DATA INFILE 'file.txt'

INTO TABLE t1

(column1, @var1)

SET column2 = @var1/100;

The **SET** clause can be used to supply values not derived from the input file. The following statement sets **column3** to the current date and time:

LOAD DATA INFILE 'file.txt'

INTO TABLE t1

(column1, column2)

SET column3 = CURRENT\_TIMESTAMP;

You can also discard an input value by assigning it to a user variable and not assigning the variable to any table column:

LOAD DATA INFILE 'file.txt'

INTO TABLE t1

(column1, @dummy, column2, @dummy, column3);

Use of the column/variable list and **SET** clause is subject to the following restrictions:

Assignments in the **SET** clause should have only column names on the left hand side of assignment operators.

You can use subqueries in the right hand side of **SET** assignments. A subquery that returns a value to be assigned to a column may be a scalar subquery only. Also, you cannot use a subquery to select from the table that is being loaded.

Lines ignored by an **IGNORE *number* LINES** clause are not processed for the column/variable list or **SET** clause.

User variables cannot be used when loading data with fixed-row format because user variables do not have a display width.

#### Column Value Assignment

To process an input line, [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) splits it into fields and uses the values according to the column/variable list and the **SET** clause, if they are present. Then the resulting row is inserted into the table. If there are **BEFORE INSERT** or **AFTER INSERT** triggers for the table, they are activated before or after inserting the row, respectively.

Interpretation of field values and assignment to table columns depends on these factors:

The SQL mode (the value of the [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) system variable). The mode can be nonstrictive, or restrictive in various ways. For example, strict SQL mode can be enabled, or the mode can include values such as [**NO\_ZERO\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_zero_date) or [**NO\_ZERO\_IN\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_zero_in_date).

Presence or absence of the **IGNORE** and **LOCAL** modifiers.

Those factors combine to produce restrictive or nonrestrictive data interpretation by [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data):

Data interpretation is restrictive if the SQL mode is restrictive and neither the **IGNORE** nor the **LOCAL** modifier is specified. Errors terminate the load operation.

Data interpretation is nonrestrictive if the SQL mode is nonrestrictive or the **IGNORE** or **LOCAL** modifier is specified. (In particular, either modifier if specified overrides a restrictive SQL mode.) Errors become warnings and the load operation continues.

Restrictive data interpretation uses these rules:

Too many or too few fields results an error.

Assigning **NULL** (that is, **\N**) to a non-**NULL** column results in an error.

A value that is out of range for the column data type results in an error.

Invalid values produce errors. For example, a value such as **'x'** for a numeric column results in an error, not conversion to 0.

By contrast, nonrestrictive data interpretation uses these rules:

If an input line has too many fields, the extra fields are ignored and the number of warnings is incremented.

If an input line has too few fields, the columns for which input fields are missing are assigned their default values. Default value assignment is described in [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults).

Assigning **NULL** (that is, **\N**) to a non-**NULL** column results in assignment of the implicit default value for the column data type. Implicit default values are described in [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults).

Invalid values produce warnings rather than errors, and are converted to the “closest” valid value for the column data type. Examples:

A value such as **'x'** for a numeric column results in conversion to 0.

An out-of-range numeric or temporal value is clipped to the closest endpoint of the range for the column data type.

An invalid value for a **DATETIME**, **DATE**, or **TIME** column is inserted as the implicit default value, regardless of the SQL mode [**NO\_ZERO\_DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_no_zero_date) setting. The implicit default is the appropriate “zero” value for the type (**'0000-00-00 00:00:00'**, **'0000-00-00'**, or **'00:00:00'**). See [Section 11.2, “Date and Time Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#date-and-time-types).

[**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) interprets an empty field value differently from a missing field:

For string types, the column is set to the empty string.

For numeric types, the column is set to **0**.

For date and time types, the column is set to the appropriate “zero” value for the type. See [Section 11.2, “Date and Time Data Types”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#date-and-time-types).

These are the same values that result if you assign an empty string explicitly to a string, numeric, or date or time type explicitly in an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement.

[**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) columns are set to the current date and time only if there is a **NULL** value for the column (that is, **\N**) and the column is not declared to permit **NULL** values, or if the [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) column default value is the current timestamp and it is omitted from the field list when a field list is specified.

[**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) regards all input as strings, so you cannot use numeric values for [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) or [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) columns the way you can with [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements. All [**ENUM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#enum) and [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#set) values must be specified as strings.

[**BIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#bit-type) values cannot be loaded directly using binary notation (for example, **b'011010'**). To work around this, use the **SET** clause to strip off the leading **b'** and trailing **'** and perform a base-2 to base-10 conversion so that MySQL loads the values into the [**BIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#bit-type) column properly:

shell> **cat /tmp/bit\_test.txt**

b'10'

b'1111111'

shell> **mysql test**

mysql> **LOAD DATA INFILE '/tmp/bit\_test.txt'**

**INTO TABLE bit\_test (@var1)**

**SET b = CAST(CONV(MID(@var1, 3, LENGTH(@var1)-3), 2, 10) AS UNSIGNED);**

Query OK, 2 rows affected (0.00 sec)

Records: 2 Deleted: 0 Skipped: 0 Warnings: 0

mysql> **SELECT BIN(b+0) FROM bit\_test;**

+----------+

| BIN(b+0) |

+----------+

| 10 |

| 1111111 |

+----------+

2 rows in set (0.00 sec)

For [**BIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#bit-type) values in **0b** binary notation (for example, **0b011010**), use this **SET** clause instead to strip off the leading **0b**:

SET b = CAST(CONV(MID(@var1, 3, LENGTH(@var1)-2), 2, 10) AS UNSIGNED)

#### Partitioned Table Support

[**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) supports explicit partition selection using the **PARTITION** clause with a list of one or more comma-separated names of partitions, subpartitions, or both. When this clause is used, if any rows from the file cannot be inserted into any of the partitions or subpartitions named in the list, the statement fails with the error Found a row not matching the given partition set. For more information and examples, see [Section 24.5, “Partition Selection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-selection).

#### Concurrency Considerations

With the **LOW\_PRIORITY** modifier, execution of the [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement is delayed until no other clients are reading from the table. This affects only storage engines that use only table-level locking (such as **MyISAM**, **MEMORY**, and **MERGE**).

With the **CONCURRENT** modifier and a **MyISAM** table that satisfies the condition for concurrent inserts (that is, it contains no free blocks in the middle), other threads can retrieve data from the table while [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) is executing. This modifier affects the performance of [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) a bit, even if no other thread is using the table at the same time.

#### Statement Result Information

When the [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement finishes, it returns an information string in the following format:

Records: 1 Deleted: 0 Skipped: 0 Warnings: 0

Warnings occur under the same circumstances as when values are inserted using the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement (see [Section 13.2.6, “INSERT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert)), except that [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) also generates warnings when there are too few or too many fields in the input row.

You can use [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) to get a list of the first [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) warnings as information about what went wrong. See [Section 13.7.7.42, “SHOW WARNINGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings).

If you are using the C API, you can get information about the statement by calling the [**mysql\_info()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-info.html) function. See [mysql\_info()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-info.html).

#### Replication Considerations

[**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) is considered unsafe for statement-based replication. If you use [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) with [**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format), each replica on which the changes are to be applied creates a temporary file containing the data. This temporary file is not encrypted, even if binary log encryption is active on the source, If encryption is required, use row-based or mixed binary logging format instead, for which replicas do not create the temporary file. For more information on the interaction between [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) and replication, see [Section 17.5.1.19, “Replication and LOAD DATA”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-load-data).

#### Miscellaneous Topics

On Unix, if you need [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) to read from a pipe, you can use the following technique (the example loads a listing of the / directory into the table **db1.t1**):

mkfifo /mysql/data/db1/ls.dat

chmod 666 /mysql/data/db1/ls.dat

find / -ls > /mysql/data/db1/ls.dat &

mysql -e "LOAD DATA INFILE 'ls.dat' INTO TABLE t1" db1

Here you must run the command that generates the data to be loaded and the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) commands either on separate terminals, or run the data generation process in the background (as shown in the preceding example). If you do not do this, the pipe blocks until data is read by the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) process.

### 13.2.8 LOAD XML Statement

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}]

[(***field\_name\_or\_user\_var***

[, ***field\_name\_or\_user\_var***] ...)]

[SET ***col\_name***={***expr*** | DEFAULT}

[, ***col\_name***={***expr*** | DEFAULT}] ...]

The [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) acts as the complement of running the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client in XML output mode (that is, starting the client with the [--xml](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_xml) option). To write data from a table to an XML file, you can invoke the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client with the [--xml](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_xml) and [-e](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_execute) options from the system shell, as shown here:

shell> **mysql --xml -e 'SELECT \* FROM mydb.mytable' > file.xml**

To read the file back into a table, use [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml). 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 MySQL tools, such as [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump).

All three 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.

Prior to MySQL 8.0.21, **LOAD XML** did not support **CDATA** sections in the source XML. (Bug #30753708, Bug #98199)

The following clauses work essentially the same way for [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) as they do for [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data):

**LOW\_PRIORITY** or **CONCURRENT**

**LOCAL**

**REPLACE** or **IGNORE**

**CHARACTER SET**

**SET**

See [Section 13.2.7, “LOAD DATA Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data), for more information about these clauses.

**(*field\_name\_or\_user\_var*, ...)** is a list of one or more comma-separated XML fields or user variables. The name of a user variable used for this purpose must match the name of a field from the XML file, prefixed with **@**. You can use field names to select only desired fields. User variables can be employed to store the corresponding field values for subsequent re-use.

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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement's **IGNORE ... LINES** clause.

Suppose that we have a table named **person**, created as shown here:

USE test;

CREATE TABLE person (

person\_id INT NOT NULL PRIMARY KEY,

fname VARCHAR(40) NULL,

lname VARCHAR(40) NULL,

created TIMESTAMP

);

Suppose further that this table is initially empty.

Now suppose that we have a simple XML file person.xml, whose contents are as shown here:

<list>

<person person\_id="1" fname="Kapek" lname="Sainnouine"/>

<person person\_id="2" fname="Sajon" lname="Rondela"/>

<person person\_id="3"><fname>Likame</fname><lname>Örrtmons</lname></person>

<person person\_id="4"><fname>Slar</fname><lname>Manlanth</lname></person>

<person><field name="person\_id">5</field><field name="fname">Stoma</field>

<field name="lname">Milu</field></person>

<person><field name="person\_id">6</field><field name="fname">Nirtam</field>

<field name="lname">Sklöd</field></person>

<person person\_id="7"><fname>Sungam</fname><lname>Dulbåd</lname></person>

<person person\_id="8" fname="Sraref" lname="Encmelt"/>

</list>

Each of the permissible XML formats discussed previously is represented in this example file.

To import the data in person.xml into the **person** table, you can use this statement:

mysql> **LOAD XML LOCAL INFILE 'person.xml'**

-> **INTO TABLE person**

-> **ROWS IDENTIFIED BY '<person>';**

Query OK, 8 rows affected (0.00 sec)

Records: 8 Deleted: 0 Skipped: 0 Warnings: 0

Here, we assume that person.xml is located in the MySQL data directory. If the file cannot be found, the following error results:

ERROR 2 (HY000): File '/person.xml' not found (Errcode: 2)

The **ROWS IDENTIFIED BY '<person>'** clause means that each **<person>** element in the XML file is considered equivalent to a row in the table into which the data is to be imported. In this case, this is the **person** table in the **test** database.

As can be seen by the response from the server, 8 rows were imported into the **test.person** table. This can be verified by a simple [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement:

mysql> **SELECT \* FROM person;**

+-----------+--------+------------+---------------------+

| person\_id | fname | lname | created |

+-----------+--------+------------+---------------------+

| 1 | Kapek | Sainnouine | 2007-07-13 16:18:47 |

| 2 | Sajon | Rondela | 2007-07-13 16:18:47 |

| 3 | Likame | Örrtmons | 2007-07-13 16:18:47 |

| 4 | Slar | Manlanth | 2007-07-13 16:18:47 |

| 5 | Stoma | Nilu | 2007-07-13 16:18:47 |

| 6 | Nirtam | Sklöd | 2007-07-13 16:18:47 |

| 7 | Sungam | Dulbåd | 2007-07-13 16:18:47 |

| 8 | Sreraf | Encmelt | 2007-07-13 16:18:47 |

+-----------+--------+------------+---------------------+

8 rows in set (0.00 sec)

This shows, as stated earlier in this section, that any or all of the 3 permitted XML formats may appear in a single file and be read using [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml).

The inverse of the import operation just shown—that is, dumping MySQL table data into an XML file—can be accomplished using the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client from the system shell, as shown here:

shell> **mysql --xml -e "SELECT \* FROM test.person" > person-dump.xml**

shell> **cat person-dump.xml**

<?xml version="1.0"?>

<resultset statement="SELECT \* FROM test.person" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<row>

<field name="person\_id">1</field>

<field name="fname">Kapek</field>

<field name="lname">Sainnouine</field>

</row>

<row>

<field name="person\_id">2</field>

<field name="fname">Sajon</field>

<field name="lname">Rondela</field>

</row>

<row>

<field name="person\_id">3</field>

<field name="fname">Likema</field>

<field name="lname">Örrtmons</field>

</row>

<row>

<field name="person\_id">4</field>

<field name="fname">Slar</field>

<field name="lname">Manlanth</field>

</row>

<row>

<field name="person\_id">5</field>

<field name="fname">Stoma</field>

<field name="lname">Nilu</field>

</row>

<row>

<field name="person\_id">6</field>

<field name="fname">Nirtam</field>

<field name="lname">Sklöd</field>

</row>

<row>

<field name="person\_id">7</field>

<field name="fname">Sungam</field>

<field name="lname">Dulbåd</field>

</row>

<row>

<field name="person\_id">8</field>

<field name="fname">Sreraf</field>

<field name="lname">Encmelt</field>

</row>

</resultset>

**Note**

The [--xml](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_xml) option causes the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to use XML formatting for its output; the [-e](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_mysql_execute) option causes the client to execute the SQL statement immediately following the option. See [Section 4.5.1, “mysql — The MySQL Command-Line Client”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql).

You can verify that the dump is valid by creating a copy of the **person** table and importing the dump file into the new table, like this:

mysql> **USE test;**

mysql> **CREATE TABLE person2 LIKE person;**

Query OK, 0 rows affected (0.00 sec)

mysql> **LOAD XML LOCAL INFILE 'person-dump.xml'**

-> **INTO TABLE person2;**

Query OK, 8 rows affected (0.01 sec)

Records: 8 Deleted: 0 Skipped: 0 Warnings: 0

mysql> **SELECT \* FROM person2;**

+-----------+--------+------------+---------------------+

| person\_id | fname | lname | created |

+-----------+--------+------------+---------------------+

| 1 | Kapek | Sainnouine | 2007-07-13 16:18:47 |

| 2 | Sajon | Rondela | 2007-07-13 16:18:47 |

| 3 | Likema | Örrtmons | 2007-07-13 16:18:47 |

| 4 | Slar | Manlanth | 2007-07-13 16:18:47 |

| 5 | Stoma | Nilu | 2007-07-13 16:18:47 |

| 6 | Nirtam | Sklöd | 2007-07-13 16:18:47 |

| 7 | Sungam | Dulbåd | 2007-07-13 16:18:47 |

| 8 | Sreraf | Encmelt | 2007-07-13 16:18:47 |

+-----------+--------+------------+---------------------+

8 rows in set (0.00 sec)

There is no requirement that every field in the XML file be matched with a column in the corresponding table. Fields which have no corresponding columns are skipped. You can see this by first emptying the **person2** table and dropping the **created** column, then using the same [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) statement we just employed previously, like this:

mysql> **TRUNCATE person2;**

Query OK, 8 rows affected (0.26 sec)

mysql> **ALTER TABLE person2 DROP COLUMN created;**

Query OK, 0 rows affected (0.52 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> **SHOW CREATE TABLE person2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: person2

Create Table: CREATE TABLE `person2` (

`person\_id` int(11) NOT NULL,

`fname` varchar(40) DEFAULT NULL,

`lname` varchar(40) DEFAULT NULL,

PRIMARY KEY (`person\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

1 row in set (0.00 sec)

mysql> **LOAD XML LOCAL INFILE 'person-dump.xml'**

-> **INTO TABLE person2;**

Query OK, 8 rows affected (0.01 sec)

Records: 8 Deleted: 0 Skipped: 0 Warnings: 0

mysql> **SELECT \* FROM person2;**

+-----------+--------+------------+

| person\_id | fname | lname |

+-----------+--------+------------+

| 1 | Kapek | Sainnouine |

| 2 | Sajon | Rondela |

| 3 | Likema | Örrtmons |

| 4 | Slar | Manlanth |

| 5 | Stoma | Nilu |

| 6 | Nirtam | Sklöd |

| 7 | Sungam | Dulbåd |

| 8 | Sreraf | Encmelt |

+-----------+--------+------------+

8 rows in set (0.00 sec)

The order in which the fields are given within each row of the XML file does not affect the operation of [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml); the field order can vary from row to row, and is not required to be in the same order as the corresponding columns in the table.

As mentioned previously, you can use a **(*field\_name\_or\_user\_var*, ...)** list of one or more XML fields (to select desired fields only) or user variables (to store the corresponding field values for later use). User variables can be especially useful when you want to insert data from an XML file into table columns whose names do not match those of the XML fields. To see how this works, we first create a table named **individual** whose structure matches that of the **person** table, but whose columns are named differently:

mysql> **CREATE TABLE individual (**

-> **individual\_id INT NOT NULL PRIMARY KEY,**

-> **name1 VARCHAR(40) NULL,**

-> **name2 VARCHAR(40) NULL,**

-> **made TIMESTAMP**

-> );

Query OK, 0 rows affected (0.42 sec)

In this case, you cannot simply load the XML file directly into the table, because the field and column names do not match:

mysql> **LOAD XML INFILE '../bin/person-dump.xml' INTO TABLE test.individual;**

ERROR 1263 (22004): Column set to default value; NULL supplied to NOT NULL column 'individual\_id' at row 1

This happens because the MySQL server looks for field names matching the column names of the target table. You can work around this problem by selecting the field values into user variables, then setting the target table's columns equal to the values of those variables using **SET**. You can perform both of these operations in a single statement, as shown here:

mysql> **LOAD XML INFILE '../bin/person-dump.xml'**

-> **INTO TABLE test.individual (@person\_id, @fname, @lname, @created)**

-> **SET individual\_id=@person\_id, name1=@fname, name2=@lname, made=@created;**

Query OK, 8 rows affected (0.05 sec)

Records: 8 Deleted: 0 Skipped: 0 Warnings: 0

mysql> **SELECT \* FROM individual;**

+---------------+--------+------------+---------------------+

| individual\_id | name1 | name2 | made |

+---------------+--------+------------+---------------------+

| 1 | Kapek | Sainnouine | 2007-07-13 16:18:47 |

| 2 | Sajon | Rondela | 2007-07-13 16:18:47 |

| 3 | Likema | Örrtmons | 2007-07-13 16:18:47 |

| 4 | Slar | Manlanth | 2007-07-13 16:18:47 |

| 5 | Stoma | Nilu | 2007-07-13 16:18:47 |

| 6 | Nirtam | Sklöd | 2007-07-13 16:18:47 |

| 7 | Sungam | Dulbåd | 2007-07-13 16:18:47 |

| 8 | Srraf | Encmelt | 2007-07-13 16:18:47 |

+---------------+--------+------------+---------------------+

8 rows in set (0.00 sec)

The names of the user variables must match those of the corresponding fields from the XML file, with the addition of the required **@** prefix to indicate that they are variables. The user variables need not be listed or assigned in the same order as the corresponding fields.

Using a **ROWS IDENTIFIED BY '<*tagname*>'** clause, it is possible to import data from the same XML file into database tables with different definitions. For this example, suppose that you have a file named address.xml which contains the following XML:

<?xml version="1.0"?>

<list>

<person person\_id="1">

<fname>Robert</fname>

<lname>Jones</lname>

<address address\_id="1" street="Mill Creek Road" zip="45365" city="Sidney"/>

<address address\_id="2" street="Main Street" zip="28681" city="Taylorsville"/>

</person>

<person person\_id="2">

<fname>Mary</fname>

<lname>Smith</lname>

<address address\_id="3" street="River Road" zip="80239" city="Denver"/>

<!-- <address address\_id="4" street="North Street" zip="37920" city="Knoxville"/> -->

</person>

</list>

You can again use the **test.person** table as defined previously in this section, after clearing all the existing records from the table and then showing its structure as shown here:

mysql< **TRUNCATE person;**

Query OK, 0 rows affected (0.04 sec)

mysql< **SHOW CREATE TABLE person\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: person

Create Table: CREATE TABLE `person` (

`person\_id` int(11) NOT NULL,

`fname` varchar(40) DEFAULT NULL,

`lname` varchar(40) DEFAULT NULL,

`created` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (`person\_id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4

1 row in set (0.00 sec)

Now create an **address** table in the **test** database using the following [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement:

CREATE TABLE address (

address\_id INT NOT NULL PRIMARY KEY,

person\_id INT NULL,

street VARCHAR(40) NULL,

zip INT NULL,

city VARCHAR(40) NULL,

created TIMESTAMP

);

To import the data from the XML file into the **person** table, execute the following [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) statement, which specifies that rows are to be specified by the **<person>** element, as shown here;

mysql> **LOAD XML LOCAL INFILE 'address.xml'**

-> **INTO TABLE person**

-> **ROWS IDENTIFIED BY '<person>';**

Query OK, 2 rows affected (0.00 sec)

Records: 2 Deleted: 0 Skipped: 0 Warnings: 0

You can verify that the records were imported using a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement:

mysql> **SELECT \* FROM person;**

+-----------+--------+-------+---------------------+

| person\_id | fname | lname | created |

+-----------+--------+-------+---------------------+

| 1 | Robert | Jones | 2007-07-24 17:37:06 |

| 2 | Mary | Smith | 2007-07-24 17:37:06 |

+-----------+--------+-------+---------------------+

2 rows in set (0.00 sec)

Since the **<address>** elements in the XML file have no corresponding columns in the **person** table, they are skipped.

To import the data from the **<address>** elements into the **address** table, use the [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) statement shown here:

mysql> **LOAD XML LOCAL INFILE 'address.xml'**

-> **INTO TABLE address**

-> **ROWS IDENTIFIED BY '<address>';**

Query OK, 3 rows affected (0.00 sec)

Records: 3 Deleted: 0 Skipped: 0 Warnings: 0

You can see that the data was imported using a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement such as this one:

mysql> **SELECT \* FROM address;**

+------------+-----------+-----------------+-------+--------------+---------------------+

| address\_id | person\_id | street | zip | city | created |

+------------+-----------+-----------------+-------+--------------+---------------------+

| 1 | 1 | Mill Creek Road | 45365 | Sidney | 2007-07-24 17:37:37 |

| 2 | 1 | Main Street | 28681 | Taylorsville | 2007-07-24 17:37:37 |

| 3 | 2 | River Road | 80239 | Denver | 2007-07-24 17:37:37 |

+------------+-----------+-----------------+-------+--------------+---------------------+

3 rows in set (0.00 sec)

The data from the **<address>** element that is enclosed in XML comments is not imported. However, since there is a **person\_id** column in the **address** table, the value of the **person\_id** attribute from the parent **<person>** element for each **<address>** is imported into the **address** table.

**Security Considerations.** As with the [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement, the transfer of the XML file from the client host to the server host is initiated by the MySQL server. In theory, a patched server could be built that would tell the client program to transfer a file of the server's choosing rather than the file named by the client in the [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) statement. Such a server could access any file on the client host to which the client user has read access.

In a Web environment, clients usually connect to MySQL from a Web server. A user that can run any command against the MySQL server can use [**LOAD XML LOCAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) to read any files to which the Web server process has read access. In this environment, the client with respect to the MySQL server is actually the Web server, not the remote program being run by the user who connects to the Web server.

You can disable loading of XML files from clients by starting the server with [**--local-infile=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_local_infile) or [**--local-infile=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_local_infile). This option can also be used when starting the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client to disable [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) for the duration of the client session.

To prevent a client from loading XML files from the server, do not grant the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege to the corresponding MySQL user account, or revoke this privilege if the client user account already has it.

**Important**

Revoking the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege (or not granting it in the first place) keeps the user only from executing the [**LOAD XML**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml) statement (as well as the [**LOAD\_FILE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_load-file) function; it does not prevent the user from executing [**LOAD XML LOCAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-xml). To disallow this statement, you must start the server or the client with --local-infile=OFF.

In other words, the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege affects only whether the client can read files on the server; it has no bearing on whether the client can read files on the local file system.

### 13.2.9 REPLACE Statement

REPLACE [LOW\_PRIORITY | DELAYED]

[INTO] ***tbl\_name***

[PARTITION (***partition\_name*** [, ***partition\_name***] ...)]

[(***col\_name*** [, ***col\_name***] ...)]

{ {VALUES | VALUE} (***value\_list***) [, (***value\_list***)] ...

|

VALUES ***row\_constructor\_list***

}

REPLACE [LOW\_PRIORITY | DELAYED]

[INTO] ***tbl\_name***

[PARTITION (***partition\_name*** [, ***partition\_name***] ...)]

SET ***assignment\_list***

REPLACE [LOW\_PRIORITY | DELAYED]

[INTO] ***tbl\_name***

[PARTITION (***partition\_name*** [, ***partition\_name***] ...)]

[(***col\_name*** [, ***col\_name***] ...)]

{SELECT ... | TABLE ***table\_name***}

***value***:

{***expr*** | DEFAULT}

***value\_list***:

***value*** [, ***value***] ...

***row\_constructor\_list***:

ROW(***value\_list***)[, ROW(***value\_list***)][, ...]

***assignment***:

***col\_name*** = ***value***

***assignment\_list***:

***assignment*** [, ***assignment***] ...

[**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) works exactly like [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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. See [Section 13.2.6, “INSERT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert).

[**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) is a MySQL extension to the SQL standard. It either inserts, or deletes and inserts. For another MySQL extension to standard SQL—that either inserts or updates—see [Section 13.2.6.2, “INSERT ... ON DUPLICATE KEY UPDATE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate).

**DELAYED** inserts and replaces were deprecated in MySQL 5.6. In MySQL 8.0, **DELAYED** is not supported. The server recognizes but ignores the **DELAYED** keyword, handles the replace as a nondelayed replace, and generates an **ER\_WARN\_LEGACY\_SYNTAX\_CONVERTED** warning: REPLACE DELAYED is no longer supported. The statement was converted to REPLACE. The **DELAYED** keyword is scheduled for removal in a future release. release.

**Note**

[**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) makes sense only if a table has a **PRIMARY KEY** or **UNIQUE** index. Otherwise, it becomes equivalent to [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statement. Any missing columns are set to their default values, just as happens for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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\_name* = *col\_name* + 1**, the reference to the column name on the right hand side is treated as [**DEFAULT(*col\_name*)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_default), so the assignment is equivalent to **SET *col\_name* = DEFAULT(*col\_name*) + 1**.

In MySQL 8.0.19 and later, you can specify the column values that **REPLACE** attempts to insert using [**VALUES ROW()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values).

To use [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), you must have both the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privileges for the table.

If a generated column is replaced explicitly, the only permitted value is **DEFAULT**. For information about generated columns, see [Section 13.1.20.8, “CREATE TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-generated-columns).

**REPLACE** supports explicit partition selection using the **PARTITION** clause with a list of comma-separated names of partitions, subpartitions, or both. As with [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), if it is not possible to insert the new row into any of these partitions or subpartitions, the **REPLACE** statement fails with the error Found a row not matching the given partition set. For more information and examples, see [Section 24.5, “Partition Selection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-selection).

The [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statement returns a count to indicate the number of rows affected. This is the sum of the rows deleted and inserted. If the count is 1 for a single-row [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), a row was inserted and no rows were deleted. If the count is greater than 1, one or more old rows were deleted before the new row was inserted. It is possible for a single row to replace more than one old row if the table contains multiple unique indexes and the new row duplicates values for different old rows in different unique indexes.

The affected-rows count makes it easy to determine whether [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) only added a row or whether it also replaced any rows: Check whether the count is 1 (added) or greater (replaced).

If you are using the C API, the affected-rows count can be obtained using the [**mysql\_affected\_rows()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-affected-rows.html) function.

You cannot replace into a table and select from the same table in a subquery.

MySQL uses the following algorithm for [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) (and [**LOAD DATA ... REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data)):

Try to insert the new row into the table

While the insertion fails because a duplicate-key error occurs for a primary key or unique index:

Delete from the table the conflicting row that has the duplicate key value

Try again to insert the new row into the table

It is possible that in the case of a duplicate-key error, a storage engine may perform the **REPLACE** as an update rather than a delete plus insert, but the semantics are the same. There are no user-visible effects other than a possible difference in how the storage engine increments **Handler\_*xxx*** status variables.

Because the results of **REPLACE ... SELECT** statements depend on the ordering of rows from the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and this order cannot always be guaranteed, it is possible when logging these statements for the source and the replica to diverge. For this reason, **REPLACE ... SELECT** statements are flagged as unsafe for statement-based replication. such statements produce a warning in the error log when using statement-based mode and are written to the binary log using the row-based format when using **MIXED** mode. See also [Section 17.2.1.1, “Advantages and Disadvantages of Statement-Based and Row-Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-sbr-rbr).

MySQL 8.0.19 and later supports [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) as well as [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) with **REPLACE**, just as it does with [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert). See [Section 13.2.6.1, “INSERT ... SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select), for more information and examples.

When modifying an existing table that is not partitioned to accommodate partitioning, or, when modifying the partitioning of an already partitioned table, you may consider altering the table's primary key (see [Section 24.6.1, “Partitioning Keys, Primary Keys, and Unique Keys”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-limitations-partitioning-keys-unique-keys)). You should be aware that, if you do this, the results of **REPLACE** statements may be affected, just as they would be if you modified the primary key of a nonpartitioned table. Consider the table created by the following [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement:

CREATE TABLE test (

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

data VARCHAR(64) DEFAULT NULL,

ts TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (id)

);

When we create this table and run the statements shown in the mysql client, the result is as follows:

mysql> **REPLACE INTO test VALUES (1, 'Old', '2014-08-20 18:47:00');**

Query OK, 1 row affected (0.04 sec)

mysql> **REPLACE INTO test VALUES (1, 'New', '2014-08-20 18:47:42');**

Query OK, 2 rows affected (0.04 sec)

mysql> **SELECT \* FROM test;**

+----+------+---------------------+

| id | data | ts |

+----+------+---------------------+

| 1 | New | 2014-08-20 18:47:42 |

+----+------+---------------------+

1 row in set (0.00 sec)

Now we create a second table almost identical to the first, except that the primary key now covers 2 columns, as shown here (emphasized text):

CREATE TABLE test2 (

id INT UNSIGNED NOT NULL AUTO\_INCREMENT,

data VARCHAR(64) DEFAULT NULL,

ts TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (id, ts)

);

When we run on **test2** the same two **REPLACE** statements as we did on the original **test** table, we obtain a different result:

mysql> **REPLACE INTO test2 VALUES (1, 'Old', '2014-08-20 18:47:00');**

Query OK, 1 row affected (0.05 sec)

mysql> **REPLACE INTO test2 VALUES (1, 'New', '2014-08-20 18:47:42');**

Query OK, 1 row affected (0.06 sec)

mysql> **SELECT \* FROM test2;**

+----+------+---------------------+

| id | data | ts |

+----+------+---------------------+

| 1 | Old | 2014-08-20 18:47:00 |

| 1 | New | 2014-08-20 18:47:42 |

+----+------+---------------------+

2 rows in set (0.00 sec)

This is due to the fact that, when run on **test2**, both the **id** and **ts** column values must match those of an existing row for the row to be replaced; otherwise, a row is inserted.

### 13.2.10 SELECT Statement

[13.2.10.1 SELECT ... INTO Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into)

[13.2.10.2 JOIN Clause](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join)

[13.2.10.3 UNION Clause](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union)

[13.2.10.4 Parenthesized Query Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#parenthesized-query-expressions)

SELECT

[ALL | DISTINCT | DISTINCTROW ]

[HIGH\_PRIORITY]

[STRAIGHT\_JOIN]

[SQL\_SMALL\_RESULT] [SQL\_BIG\_RESULT] [SQL\_BUFFER\_RESULT]

[SQL\_NO\_CACHE] [SQL\_CALC\_FOUND\_ROWS]

***select\_expr*** [, ***select\_expr***] ...

[***into\_option***]

[FROM ***table\_references***

[PARTITION ***partition\_list***]]

[WHERE ***where\_condition***]

[GROUP BY {***col\_name*** | ***expr*** | ***position***}, ... [WITH ROLLUP]]

[HAVING ***where\_condition***]

[WINDOW ***window\_name*** AS (***window\_spec***)

[, ***window\_name*** AS (***window\_spec***)] ...]

[ORDER BY {***col\_name*** | ***expr*** | ***position***}

[ASC | DESC], ... [WITH ROLLUP]]

[LIMIT {[***offset***,] ***row\_count*** | ***row\_count*** OFFSET ***offset***}]

[***into\_option***]

[FOR {UPDATE | SHARE}

[OF ***tbl\_name*** [, ***tbl\_name***] ...]

[NOWAIT | SKIP LOCKED]

| LOCK IN SHARE MODE]

[***into\_option***]

***into\_option***: {

INTO OUTFILE '***file\_name***'

[CHARACTER SET ***charset\_name***]

***export\_options***

| INTO DUMPFILE '***file\_name***'

| INTO ***var\_name*** [, ***var\_name***] ...

}

[**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) is used to retrieve rows selected from one or more tables, and can include [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) statements and subqueries. See [Section 13.2.10.3, “UNION Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), and [Section 13.2.11, “Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subqueries). A [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement can start with a [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause to define common table expressions accessible within the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). See [Section 13.2.15, “WITH (Common Table Expressions)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with).

The most commonly used clauses of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements are these:

Each ***select\_expr*** indicates a column that you want to retrieve. There must be at least one ***select\_expr***.

***table\_references*** indicates the table or tables from which to retrieve rows. Its syntax is described in [Section 13.2.10.2, “JOIN Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join).

**SELECT** supports explicit partition selection using the **PARTITION** clause with a list of partitions or subpartitions (or both) following the name of the table in a ***table\_reference*** (see [Section 13.2.10.2, “JOIN Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join)). In this case, rows are selected only from the partitions listed, and any other partitions of the table are ignored. For more information and examples, see [Section 24.5, “Partition Selection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-selection).

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** expression, you can use any of the functions and operators that MySQL supports, except for aggregate (group) functions. See [Section 9.5, “Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#expressions), and [Chapter 12, *Functions and Operators*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html).

[**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) can also be used to retrieve rows computed without reference to any table.

For example:

mysql> **SELECT 1 + 1;**

-> 2

You are permitted to specify **DUAL** as a dummy table name in situations where no tables are referenced:

mysql> **SELECT 1 + 1 FROM DUAL;**

-> 2

**DUAL** is purely for the convenience of people who require that all [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements should have **FROM** and possibly other clauses. MySQL may ignore the clauses. MySQL does not require **FROM DUAL** if no tables are referenced.

In general, clauses used must be given in exactly the order shown in the syntax description. For example, a **HAVING** clause must come after any **GROUP BY** clause and before any **ORDER BY** clause. The **INTO** clause, if present, can appear in any position indicated by the syntax description, but within a given statement can appear only once, not in multiple positions. For more information about **INTO**, see [Section 13.2.10.1, “SELECT ... INTO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into).

The list of ***select\_expr*** terms comprises the select list that indicates which columns to retrieve. Terms specify a column or expression or can use **\***-shorthand:

A select list consisting only of a single unqualified **\*** can be used as shorthand to select all columns from all tables:

SELECT \* FROM t1 INNER JOIN t2 ...

***tbl\_name*.\*** can be used as a qualified shorthand to select all columns from the named table:

SELECT t1.\*, t2.\* FROM t1 INNER JOIN t2 ...

If a table has invisible columns, **\*** and ***tbl\_name*.\*** do not include them. To be included, invisible columns must be referenced explicitly.

Use of an unqualified **\*** with other items in the select list may produce a parse error. To avoid this problem, use a qualified ***tbl\_name*.\*** reference:

SELECT AVG(score), t1.\* FROM t1 ...

The following list provides additional information about other **SELECT** clauses:

A ***select\_expr*** can be given an alias using **AS *alias\_name***. The alias is used as the expression's column name and can be used in **GROUP BY**, **ORDER BY**, or **HAVING** clauses. For example:

SELECT CONCAT(last\_name,', ',first\_name) AS full\_name

FROM mytable ORDER BY full\_name;

The **AS** keyword is optional when aliasing a ***select\_expr*** with an identifier. The preceding example could have been written like this:

SELECT CONCAT(last\_name,', ',first\_name) full\_name

FROM mytable ORDER BY full\_name;

However, because the **AS** is optional, a subtle problem can occur if you forget the comma between two ***select\_expr*** expressions: MySQL interprets the second as an alias name. For example, in the following statement, **columnb** is treated as an alias name:

SELECT columna columnb FROM mytable;

For this reason, it is good practice to be in the habit of using **AS** explicitly when specifying column aliases.

It is not permissible to refer to a column alias in a **WHERE** clause, because the column value might not yet be determined when the **WHERE** clause is executed. See [Section B.3.4.4, “Problems with Column Aliases”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#problems-with-alias).

The **FROM *table\_references*** clause indicates the table or tables from which to retrieve rows. If you name more than one table, you are performing a join. For information on join syntax, see [Section 13.2.10.2, “JOIN Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join). For each table specified, you can optionally specify an alias.

***tbl\_name*** [[AS] ***alias***] [***index\_hint***]

The use of index hints provides the optimizer with information about how to choose indexes during query processing. For a description of the syntax for specifying these hints, see [Section 8.9.4, “Index Hints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#index-hints).

You can use **SET max\_seeks\_for\_key=*value*** as an alternative way to force MySQL to prefer key scans instead of table scans. See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

You can refer to a table within the default database as ***tbl\_name***, or as ***db\_name***.***tbl\_name*** to specify a database explicitly. You can refer to a column as ***col\_name***, ***tbl\_name***.***col\_name***, or ***db\_name***.***tbl\_name***.***col\_name***. You need not specify a ***tbl\_name*** or ***db\_name***.***tbl\_name*** prefix for a column reference unless the reference would be ambiguous. See [Section 9.2.2, “Identifier Qualifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifier-qualifiers), for examples of ambiguity that require the more explicit column reference forms.

A table reference can be aliased using ***tbl\_name* AS *alias\_name*** or ***tbl\_name alias\_name***. These statements are equivalent:

SELECT t1.name, t2.salary FROM employee AS t1, info AS t2

WHERE t1.name = t2.name;

SELECT t1.name, t2.salary FROM employee t1, info t2

WHERE t1.name = t2.name;

Columns selected for output can be referred to in **ORDER BY** and **GROUP BY** clauses using column names, column aliases, or column positions. Column positions are integers and begin with 1:

SELECT college, region, seed FROM tournament

ORDER BY region, seed;

SELECT college, region AS r, seed AS s FROM tournament

ORDER BY r, s;

SELECT college, region, seed FROM tournament

ORDER BY 2, 3;

To sort in reverse order, add the **DESC** (descending) keyword to the name of the column in the **ORDER BY** clause that you are sorting by. The default is ascending order; this can be specified explicitly using the **ASC** keyword.

If **ORDER BY** occurs within a parenthesized query expression and also is applied in the outer query, the results are undefined and may change in a future version of MySQL.

Use of column positions is deprecated because the syntax has been removed from the SQL standard.

Prior to MySQL 8.0.13, MySQL supported a nonstandard syntax extension that permitted explicit **ASC** or **DESC** designators for **GROUP BY** columns. MySQL 8.0.12 and later supports **ORDER BY** with grouping functions so that use of this extension is no longer necessary. (Bug #86312, Bug #26073525) This also means you can sort on an arbitrary column or columns when using **GROUP BY**, like this:

SELECT a, b, COUNT(c) AS t FROM test\_table GROUP BY a,b ORDER BY a,t DESC;

As of MySQL 8.0.13, the **GROUP BY** extension is no longer supported: **ASC** or **DESC** designators for **GROUP BY** columns are not permitted.

When you use **ORDER BY** or **GROUP BY** to sort a column in a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), the server sorts values using only the initial number of bytes indicated by the [**max\_sort\_length**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_sort_length) system variable.

MySQL extends the use of **GROUP BY** to permit selecting fields that are not mentioned in the **GROUP BY** clause. If you are not getting the results that you expect from your query, please read the description of **GROUP BY** found in [Section 12.20, “Aggregate Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#aggregate-functions-and-modifiers).

**GROUP BY** permits a **WITH ROLLUP** modifier. See [Section 12.20.2, “GROUP BY Modifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-modifiers).

Previously, it was not permitted to use **ORDER BY** in a query having a **WITH ROLLUP** modifier. This restriction is lifted as of MySQL 8.0.12. See [Section 12.20.2, “GROUP BY Modifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#group-by-modifiers).

The **HAVING** clause is applied nearly last, just before items are sent to the client, with no optimization. (**LIMIT** is applied after **HAVING**.)

The SQL standard requires that **HAVING** must reference only columns in the **GROUP BY** clause or columns used in aggregate functions. However, MySQL supports an extension to this behavior, and permits **HAVING** to refer to columns in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) list and columns in outer subqueries as well.

If the **HAVING** clause refers to a column that is ambiguous, a warning occurs. In the following statement, **col2** is ambiguous because it is used as both an alias and a column name:

SELECT COUNT(col1) AS col2 FROM t GROUP BY col2 HAVING col2 = 2;

Preference is given to standard SQL behavior, so if a **HAVING** column name is used both in **GROUP BY** and as an aliased column in the output column list, preference is given to the column in the **GROUP BY** column.

Do not use **HAVING** for items that should be in the **WHERE** clause. For example, do not write the following:

SELECT ***col\_name*** FROM ***tbl\_name*** HAVING ***col\_name*** > 0;

Write this instead:

SELECT ***col\_name*** FROM ***tbl\_name*** WHERE ***col\_name*** > 0;

The **HAVING** clause can refer to aggregate functions, which the **WHERE** clause cannot:

SELECT user, MAX(salary) FROM users

GROUP BY user HAVING MAX(salary) > 10;

(This did not work in some older versions of MySQL.)

MySQL permits duplicate column names. That is, there can be more than one ***select\_expr*** with the same name. This is an extension to standard SQL. Because MySQL also permits **GROUP BY** and **HAVING** to refer to ***select\_expr*** values, this can result in an ambiguity:

SELECT 12 AS a, a FROM t GROUP BY a;

In that statement, both columns have the name **a**. To ensure that the correct column is used for grouping, use different names for each ***select\_expr***.

The **WINDOW** clause, if present, defines named windows that can be referred to by window functions. For details, see [Section 12.21.4, “Named Windows”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#window-functions-named-windows).

MySQL resolves unqualified column or alias references in **ORDER BY** clauses by searching in the ***select\_expr*** values, then in the columns of the tables in the **FROM** clause. For **GROUP BY** or **HAVING** clauses, it searches the **FROM** clause before searching in the ***select\_expr*** values. (For **GROUP BY** and **HAVING**, this differs from the pre-MySQL 5.0 behavior that used the same rules as for **ORDER BY**.)

The **LIMIT** clause can be used to constrain the number of rows returned by the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. **LIMIT** takes one or two numeric arguments, which must both be nonnegative integer constants, with these exceptions:

Within prepared statements, **LIMIT** parameters can be specified using **?** placeholder markers.

Within stored programs, **LIMIT** parameters can be specified using integer-valued routine parameters or local variables.

With two arguments, the first argument specifies the offset of the first row to return, and the second specifies the maximum number of rows to return. The offset of the initial row is 0 (not 1):

SELECT \* FROM tbl LIMIT 5,10; # Retrieve rows 6-15

To retrieve all rows from a certain offset up to the end of the result set, you can use some large number for the second parameter. This statement retrieves all rows from the 96th row to the last:

SELECT \* FROM tbl LIMIT 95,18446744073709551615;

With one argument, the value specifies the number of rows to return from the beginning of the result set:

SELECT \* FROM tbl LIMIT 5; # Retrieve first 5 rows

In other words, **LIMIT *row\_count*** is equivalent to **LIMIT 0, *row\_count***.

For prepared statements, you can use placeholders. The following statements return one row from the **tbl** table:

SET @a=1;

PREPARE STMT FROM 'SELECT \* FROM tbl LIMIT ?';

EXECUTE STMT USING @a;

The following statements return the second to sixth rows from the **tbl** table:

SET @skip=1; SET @numrows=5;

PREPARE STMT FROM 'SELECT \* FROM tbl LIMIT ?, ?';

EXECUTE STMT USING @skip, @numrows;

For compatibility with PostgreSQL, MySQL also supports the **LIMIT *row\_count* OFFSET *offset*** syntax.

If **LIMIT** occurs within a parenthesized query expression and also is applied in the outer query, the results are undefined and may change in a future version of MySQL.

The [**SELECT ... INTO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) form of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) enables the query result to be written to a file or stored in variables. For more information, see [Section 13.2.10.1, “SELECT ... INTO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into).

If you use **FOR UPDATE** with a storage engine that uses page or row locks, rows examined by the query are write-locked until the end of the current transaction.

You cannot use **FOR UPDATE** as part of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) in a statement such as [**CREATE TABLE *new\_table* SELECT ... FROM *old\_table* ...**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select). (If you attempt to do so, the statement is rejected with the error Can't update table '***old\_table***' while '***new\_table***' is being created.)

**FOR SHARE** and **LOCK IN SHARE MODE** set shared locks that permit other transactions to read the examined rows but not to update or delete them. **FOR SHARE** and **LOCK IN SHARE MODE** are equivalent. However, **FOR SHARE**, like **FOR UPDATE**, supports **NOWAIT**, **SKIP LOCKED**, and **OF *tbl\_name*** options. **FOR SHARE** is a replacement for **LOCK IN SHARE MODE**, but **LOCK IN SHARE MODE** remains available for backward compatibility.

**NOWAIT** causes a **FOR UPDATE** or **FOR SHARE** query to execute immediately, returning an error if a row lock cannot be obtained due to a lock held by another transaction.

**SKIP LOCKED** causes a **FOR UPDATE** or **FOR SHARE** query to execute immediately, excluding rows from the result set that are locked by another transaction.

**NOWAIT** and **SKIP LOCKED** options are unsafe for statement-based replication.

**Note**

Queries that skip locked rows return an inconsistent view of the data. **SKIP LOCKED** is therefore not suitable for general transactional work. However, it may be used to avoid lock contention when multiple sessions access the same queue-like table.

**OF *tbl\_name*** applies **FOR UPDATE** and **FOR SHARE** queries to named tables. For example:

SELECT \* FROM t1, t2 FOR SHARE OF t1 FOR UPDATE OF t2;

All tables referenced by the query block are locked when **OF *tbl\_name*** is omitted. Consequently, using a locking clause without **OF *tbl\_name*** in combination with another locking clause returns an error. Specifying the same table in multiple locking clauses returns an error. If an alias is specified as the table name in the **SELECT** statement, a locking clause may only use the alias. If the **SELECT** statement does not specify an alias explicitly, the locking clause may only specify the actual table name.

For more information about **FOR UPDATE** and **FOR SHARE**, see [Section 15.7.2.4, “Locking Reads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-reads). For additional information about **NOWAIT** and **SKIP LOCKED** options, see [Locking Read Concurrency with NOWAIT and SKIP LOCKED](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-locking-reads-nowait-skip-locked).

Following the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) keyword, you can use a number of modifiers that affect the operation of the statement. **HIGH\_PRIORITY**, **STRAIGHT\_JOIN**, and modifiers beginning with **SQL\_** are MySQL extensions to standard SQL.

The **ALL** and **DISTINCT** modifiers specify whether duplicate rows should be returned. **ALL** (the default) specifies that all matching rows should be returned, including duplicates. **DISTINCT** specifies removal of duplicate rows from the result set. It is an error to specify both modifiers. **DISTINCTROW** is a synonym for **DISTINCT**.

In MySQL 8.0.12 and later, **DISTINCT** can be used with a query that also uses **WITH ROLLUP**. (Bug #87450, Bug #26640100)

**HIGH\_PRIORITY** gives the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) higher priority than a statement that updates a table. You should use this only for queries that are very fast and must be done at once. A **SELECT HIGH\_PRIORITY** query that is issued while the table is locked for reading runs even if there is an update statement waiting for the table to be free. This affects only storage engines that use only table-level locking (such as **MyISAM**, **MEMORY**, and **MERGE**).

**HIGH\_PRIORITY** cannot be used with [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements that are part of a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union).

**STRAIGHT\_JOIN** forces the optimizer to join the tables in the order in which they are listed in the **FROM** clause. You can use this to speed up a query if the optimizer joins the tables in nonoptimal order. **STRAIGHT\_JOIN** also can be used in the ***table\_references*** list. See [Section 13.2.10.2, “JOIN Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join).

**STRAIGHT\_JOIN** does not apply to any table that the optimizer treats as a [**const**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#jointype_const) or [**system**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#jointype_system) table. Such a table produces a single row, is read during the optimization phase of query execution, and references to its columns are replaced with the appropriate column values before query execution proceeds. These tables appear first in the query plan displayed by [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain). See [Section 8.8.1, “Optimizing Queries with EXPLAIN”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#using-explain). This exception may not apply to [**const**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#jointype_const) or [**system**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#jointype_system) tables that are used on the **NULL**-complemented side of an outer join (that is, the right-side table of a **LEFT JOIN** or the left-side table of a **RIGHT JOIN**.

**SQL\_BIG\_RESULT** or **SQL\_SMALL\_RESULT** can be used with **GROUP BY** or **DISTINCT** to tell the optimizer that the result set has many rows or is small, respectively. For **SQL\_BIG\_RESULT**, MySQL directly uses disk-based temporary tables if they are created, and prefers sorting to using a temporary table with a key on the **GROUP BY** elements. For **SQL\_SMALL\_RESULT**, MySQL uses in-memory temporary tables to store the resulting table instead of using sorting. This should not normally be needed.

**SQL\_BUFFER\_RESULT** forces the result to be put into a temporary table. This helps MySQL free the table locks early and helps in cases where it takes a long time to send the result set to the client. This modifier can be used only for top-level [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, not for subqueries or following [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union).

**SQL\_CALC\_FOUND\_ROWS** tells MySQL to calculate how many rows there would be in the result set, disregarding any **LIMIT** clause. The number of rows can then be retrieved with **SELECT FOUND\_ROWS()**. See [Section 12.16, “Information Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#information-functions).

**Note**

The **SQL\_CALC\_FOUND\_ROWS** query modifier and accompanying [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) function are deprecated as of MySQL 8.0.17; expect them to be removed in a future version of MySQL. See the [**FOUND\_ROWS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_found-rows) description for information about an alternative strategy.

The **SQL\_CACHE** and **SQL\_NO\_CACHE** modifiers were used with the query cache prior to MySQL 8.0. The query cache was removed in MySQL 8.0. The **SQL\_CACHE** modifier was removed as well. **SQL\_NO\_CACHE** is deprecated, and has no effect; expect it to be removed in a future MySQL release.

#### 13.2.10.1 SELECT ... INTO Statement

The [**SELECT ... INTO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) form of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) enables a query result to be stored in variables or written to a file:

**SELECT ... INTO *var\_list*** selects column values and stores them into variables.

**SELECT ... INTO OUTFILE** writes the selected rows to a file. Column and line terminators can be specified to produce a specific output format.

**SELECT ... INTO DUMPFILE** writes a single row to a file without any formatting.

A given [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement can contain at most one **INTO** clause, although as shown by the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) syntax description (see [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)), the **INTO** can appear in different positions:

Before **FROM**. Example:

SELECT \* INTO @myvar FROM t1;

Before a trailing locking clause. Example:

SELECT \* FROM t1 INTO @myvar FOR UPDATE;

At the end of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). Example:

SELECT \* FROM t1 FOR UPDATE INTO @myvar;

The **INTO** position at the end of the statement is supported as of MySQL 8.0.20, and is the preferred position. The position before a locking clause is deprecated as of MySQL 8.0.20; expect support for it to be removed in a future version of MySQL. In other words, **INTO** after **FROM** but not at the end of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) produces a warning.

An **INTO** clause should not be used in a nested [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) because such a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) must return its result to the outer context. There are also constraints on the use of **INTO** within [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) statements; see [Section 13.2.10.3, “UNION Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union).

For the **INTO *var\_list*** variant:

***var\_list*** names a list of one or more variables, each of which can be a user-defined variable, stored procedure or function parameter, or stored program local variable. (Within a prepared **SELECT ... INTO *var\_list*** statement, only user-defined variables are permitted; see [Section 13.6.4.2, “Local Variable Scope and Resolution”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#local-variable-scope).)

The selected values are assigned to the variables. The number of variables must match the number of columns. 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.

SELECT id, data INTO @x, @y FROM test.t1 LIMIT 1;

**INTO *var\_list*** can also be used with a [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement, subject to these restrictions:

The number of variables must match the number of columns in the table.

If the table contains more than one row, you must use **LIMIT 1** to limit the result set to a single row. **LIMIT 1** must precede the **INTO** keyword.

An example of such a statement is shown here:

TABLE employees ORDER BY lname DESC LIMIT 1

INTO @id, @fname, @lname, @hired, @separated, @job\_code, @store\_id;

You can also select values from a [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statement that generates a single row into a set of user variables. In this case, you must employ a table alias, and you must assign each value from the value list to a variable. Each of the two statements shown here is equivalent to [**SET @x=2, @y=4, @z=8**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable):

SELECT \* FROM (VALUES ROW(2,4,8)) AS t INTO @x,@y,@z;

SELECT \* FROM (VALUES ROW(2,4,8)) AS t(a,b,c) INTO @x,@y,@z;

User variable names are not case-sensitive. See [Section 9.4, “User-Defined Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#user-variables).

The [**SELECT ... INTO OUTFILE '*file\_name*'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) form of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) writes the selected rows to a file. The file is created on the server host, so you must have the [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) privilege to use this syntax. ***file\_name*** cannot be an existing file, which among other things prevents files such as /etc/passwd and database tables from being modified. The [**character\_set\_filesystem**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_filesystem) system variable controls the interpretation of the file name.

The [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) statement is intended to enable dumping a table to a text file on the server host. To create the resulting file on some other host, [**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) normally is unsuitable because there is no way to write a path to the file relative to the server host file system, unless the location of the file on the remote host can be accessed using a network-mapped path on the server host file system.

Alternatively, if the MySQL client software is installed on the remote host, you can use a client command such as **mysql -e "SELECT ..." > *file\_name*** to generate the file on that host.

[**SELECT ... INTO OUTFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) is the complement of [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data). Column values are written converted to the character set specified in the **CHARACTER SET** clause. If no such clause is present, values are dumped using the **binary** character set. In effect, there is no character set conversion. If a result set contains columns in several character sets, so is the output data file, and it may not be possible to reload the file correctly.

The syntax for the ***export\_options*** part of the statement consists of the same **FIELDS** and **LINES** clauses that are used with the [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) statement. For information about the **FIELDS** and **LINES** clauses, including their default values and permissible values, see [Section 13.2.7, “LOAD DATA Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data).

**FIELDS ESCAPED BY** controls how to write special characters. If the **FIELDS ESCAPED BY** character is not empty, it is used when necessary to avoid ambiguity as a prefix that precedes following characters on output:

The **FIELDS ESCAPED BY** character

The **FIELDS [OPTIONALLY] ENCLOSED BY** character

The first character of the **FIELDS TERMINATED BY** and **LINES TERMINATED BY** values

ASCII **NUL** (the zero-valued byte; what is actually written following the escape character is ASCII **0**, not a zero-valued byte)

The **FIELDS TERMINATED BY**, **ENCLOSED BY**, **ESCAPED BY**, or **LINES TERMINATED BY** characters must be escaped so that you can read the file back in reliably. ASCII **NUL** is escaped to make it easier to view with some pagers.

The resulting file need not conform to SQL syntax, so nothing else need be escaped.

If the **FIELDS ESCAPED BY** character is empty, no characters are escaped and **NULL** is output as **NULL**, not **\N**. It is probably not a good idea to specify an empty escape character, particularly if field values in your data contain any of the characters in the list just given.

**INTO OUTFILE** can also be used with a [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement when you want to dump all columns of a table into a text file. In this case, the ordering and number of rows can be controlled using **ORDER BY** and **LIMIT**; these clauses must precede **INTO OUTFILE**. **TABLE ... INTO OUTFILE** supports the same ***export\_options*** as does **SELECT ... INTO OUTFILE**, and it is subject to the same restrictions on writing to the file system. An example of such a statement is shown here:

TABLE employees ORDER BY lname LIMIT 1000

INTO OUTFILE '/tmp/employee\_data\_1.txt'

FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"', ESCAPED BY '\'

LINES TERMINATED BY '\n';

You can also use **SELECT ... INTO OUTFILE** with a [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statement to write values directly into a file. An example is shown here:

SELECT \* FROM (VALUES ROW(1,2,3),ROW(4,5,6),ROW(7,8,9)) AS t

INTO OUTFILE '/tmp/select-values.txt';

You must use a table alias; column aliases are also supported, and can optionally be used to write values only from desired columns. You can also use any or all of the export options supported by **SELECT ... INTO OUTFILE** to format the output to the file.

Here is an example that produces a file in the comma-separated values (CSV) format used by many programs:

SELECT a,b,a+b INTO OUTFILE '/tmp/result.txt'

FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

FROM test\_table;

If you use **INTO DUMPFILE** instead of **INTO OUTFILE**, MySQL writes only one row into the file, without any column or line termination and without performing any escape processing. This is useful for selecting a [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) value and storing it in a file.

[**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) also supports **INTO DUMPFILE**. If the table contains more than one row, you must also use **LIMIT 1** to limit the output to a single row. **INTO DUMPFILE** can also be used with **SELECT \* FROM (VALUES ROW()[, ...]) AS *table\_alias* [LIMIT 1]**. See [Section 13.2.14, “VALUES Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values).

**Note**

Any file created by **INTO OUTFILE** or **INTO DUMPFILE** is owned by the operating system user under whose account [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) runs. (You should never run [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) as **root** for this and other reasons.) As of MySQL 8.0.17, the umask for file creation is 0640; you must have sufficient access privileges to manipulate the file contents. Prior to MySQL 8.0.17, the umask is 0666 and the file is writable by all users on the server host.

If the [**secure\_file\_priv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_secure_file_priv) system variable is set to a nonempty directory name, the file to be written must be located in that directory.

In the context of [**SELECT ... INTO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) 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. For additional information, see [Section 25.4.5, “Event Scheduler Status”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#events-status-info).

As of MySQL 8.0.22, support is provided for periodic synchronization of output files written to by **SELECT INTO OUTFILE** and **SELECT INTO DUMPFILE**, enabled by setting the [**select\_into\_disk\_sync**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_select_into_disk_sync) server system variable introduced in that version. Output buffer size and optional delay can be set using, respectively, [**select\_into\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_select_into_buffer_size) and [**select\_into\_disk\_sync\_delay**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_select_into_disk_sync_delay). For more information, see the descriptions of these system variables.

#### 13.2.10.2 JOIN Clause

MySQL supports the following **JOIN** syntax for the ***table\_references*** part of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements and multiple-table [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements:

***table\_references:***

***escaped\_table\_reference*** [, ***escaped\_table\_reference***] ...

***escaped\_table\_reference***: {

***table\_reference***

| { OJ ***table\_reference*** }

}

***table\_reference***: {

***table\_factor***

| ***joined\_table***

}

***table\_factor***: {

***tbl\_name*** [PARTITION (***partition\_names***)]

[[AS] ***alias***] [***index\_hint\_list***]

| [LATERAL] ***table\_subquery*** [AS] ***alias*** [(***col\_list***)]

| ( ***table\_references*** )

}

***joined\_table***: {

***table\_reference*** {[INNER | CROSS] JOIN | STRAIGHT\_JOIN} ***table\_factor*** [***join\_specification***]

| ***table\_reference*** {LEFT|RIGHT} [OUTER] JOIN ***table\_reference*** ***join\_specification***

| ***table\_reference*** NATURAL [INNER | {LEFT|RIGHT} [OUTER]] JOIN ***table\_factor***

}

***join\_specification***: {

ON ***search\_condition***

| USING (***join\_column\_list***)

}

***join\_column\_list***:

***column\_name*** [, ***column\_name***] ...

***index\_hint\_list***:

***index\_hint*** [, ***index\_hint***] ...

***index\_hint***: {

USE {INDEX|KEY}

[FOR {JOIN|ORDER BY|GROUP BY}] ([***index\_list***])

| {IGNORE|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.

A table reference (when it refers to a partitioned table) may contain a **PARTITION** clause, including a list of comma-separated partitions, subpartitions, or both. This option follows the name of the table and precedes any alias declaration. The effect of this option is that rows are selected only from the listed partitions or subpartitions. Any partitions or subpartitions not named in the list are ignored. For more information and examples, see [Section 24.5, “Partition Selection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-selection).

The syntax of ***table\_factor*** is extended in MySQL in comparison with standard SQL. The standard accepts only ***table\_reference***, not a list of them inside a pair of parentheses.

This is a conservative extension if each comma in a list of ***table\_reference*** items is considered 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 MySQL, **JOIN**, **CROSS JOIN**, and **INNER JOIN** are syntactic equivalents (they 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. MySQL also supports nested joins. See [Section 8.2.1.8, “Nested Join Optimization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#nested-join-optimization).

Index hints can be specified to affect how the MySQL optimizer makes use of indexes. For more information, see [Section 8.9.4, “Index Hints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#index-hints). Optimizer hints and the **optimizer\_switch** system variable are other ways to influence optimizer use of indexes. See [Section 8.9.3, “Optimizer Hints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints), and [Section 8.9.2, “Switchable Optimizations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#switchable-optimizations).

The following list describes general factors to take into account when writing joins:

A table reference can be aliased using ***tbl\_name* AS *alias\_name*** or ***tbl\_name alias\_name***:

SELECT t1.name, t2.salary

FROM employee AS t1 INNER JOIN info AS t2 ON t1.name = t2.name;

SELECT t1.name, t2.salary

FROM employee t1 INNER JOIN info t2 ON t1.name = t2.name;

A ***table\_subquery*** is also known as a derived table or subquery in the **FROM** clause. See [Section 13.2.11.8, “Derived Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#derived-tables). Such subqueries must include an alias to give the subquery result a table name, and may optionally include a list of table column names in parentheses. A trivial example follows:

SELECT \* FROM (SELECT 1, 2, 3) AS t1;

The maximum number of tables that can be referenced in a single join is 61. This includes a join handled by merging derived tables and views in the **FROM** clause into the outer query block (see [Section 8.2.2.4, “Optimizing Derived Tables, View References, and Common Table Expressions with Merging or Materialization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#derived-table-optimization)).

**INNER JOIN** and **,** (comma) are semantically equivalent in the absence of a join condition: both produce a Cartesian product between the specified tables (that is, each and every row in the first table is joined to each and every row in the second table).

However, the precedence of the comma operator is less than that of **INNER JOIN**, **CROSS JOIN**, **LEFT JOIN**, and so on. If you mix comma joins with the other join types when there is a join condition, an error of the form **Unknown column '*col\_name*' in 'on clause'** may occur. Information about dealing with this problem is given later in this section.

The ***search\_condition*** used with **ON** is any conditional expression of the form that can be used in a **WHERE** clause. Generally, the **ON** clause serves for conditions that specify how to join tables, and the **WHERE** clause restricts which rows to include in the result set.

If there is no matching row for the right table in the **ON** or **USING** part in a **LEFT JOIN**, a row with all columns set to **NULL** is used for the right table. You can use this fact to find rows in a table that have no counterpart in another table:

SELECT left\_tbl.\*

FROM left\_tbl LEFT JOIN right\_tbl ON left\_tbl.id = right\_tbl.id

WHERE right\_tbl.id IS NULL;

This example finds all rows in **left\_tbl** with an **id** value that is not present in **right\_tbl** (that is, all rows in **left\_tbl** with no corresponding row in **right\_tbl**). See [Section 8.2.1.9, “Outer Join Optimization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#outer-join-optimization).

The **USING(*join\_column\_list*)** clause names a list of columns that must exist in both tables. If tables **a** and **b** both contain columns **c1**, **c2**, and **c3**, the following join compares corresponding columns from the two tables:

a LEFT JOIN b USING (c1, c2, c3)

The **NATURAL [LEFT] JOIN** of two tables is defined to be semantically equivalent to an **INNER JOIN** or a **LEFT JOIN** with a **USING** clause that names all columns that exist in both tables.

**RIGHT JOIN** works analogously to **LEFT JOIN**. To keep code portable across databases, it is recommended that you use **LEFT JOIN** instead of **RIGHT JOIN**.

The **{ OJ ... }** syntax shown in the join syntax description exists only for compatibility with ODBC. The curly braces in the syntax should be written literally; they are not metasyntax as used elsewhere in syntax descriptions.

SELECT left\_tbl.\*

FROM { OJ left\_tbl LEFT OUTER JOIN right\_tbl

ON left\_tbl.id = right\_tbl.id }

WHERE right\_tbl.id IS NULL;

You can use other types of joins within **{ OJ ... }**, such as **INNER JOIN** or **RIGHT OUTER JOIN**. This helps with compatibility with some third-party applications, but is not official ODBC syntax.

**STRAIGHT\_JOIN** is similar to **JOIN**, except that the left table is always read before the right table. This can be used for those (few) cases for which the join optimizer processes the tables in a suboptimal order.

Some join examples:

SELECT \* FROM table1, table2;

SELECT \* FROM table1 INNER JOIN table2 ON table1.id = table2.id;

SELECT \* FROM table1 LEFT JOIN table2 ON table1.id = table2.id;

SELECT \* FROM table1 LEFT JOIN table2 USING (id);

SELECT \* FROM table1 LEFT JOIN table2 ON table1.id = table2.id

LEFT JOIN table3 ON table2.id = table3.id;

Natural joins and joins with **USING**, including outer join variants, are processed according to the SQL:2003 standard:

Redundant columns of a **NATURAL** join do not appear. Consider this set of statements:

CREATE TABLE t1 (i INT, j INT);

CREATE TABLE t2 (k INT, j INT);

INSERT INTO t1 VALUES(1, 1);

INSERT INTO t2 VALUES(1, 1);

SELECT \* FROM t1 NATURAL JOIN t2;

SELECT \* FROM t1 JOIN t2 USING (j);

In the first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, column **j** appears in both tables and thus becomes a join column, so, according to standard SQL, it should appear only once in the output, not twice. Similarly, in the second SELECT statement, column **j** is named in the **USING** clause and should appear only once in the output, not twice.

Thus, the statements produce this output:

+------+------+------+

| j | i | k |

+------+------+------+

| 1 | 1 | 1 |

+------+------+------+

+------+------+------+

| j | i | k |

+------+------+------+

| 1 | 1 | 1 |

+------+------+------+

Redundant column elimination and column ordering occurs according to standard SQL, producing this display order:

First, coalesced common columns of the two joined tables, in the order in which they occur in the first table

Second, columns unique to the first table, in order in which they occur in that table

Third, columns unique to the second table, in order in which they occur in that table

The single result column that replaces two common columns is defined using the coalesce operation. That is, for two **t1.a** and **t2.a** the resulting single join column **a** is defined as **a = COALESCE(t1.a, t2.a)**, where:

COALESCE(x, y) = (CASE WHEN x IS NOT NULL THEN x ELSE y END)

If the join operation is any other join, the result columns of the join consist of the concatenation of all columns of the joined tables.

A consequence of the definition of coalesced columns is that, for outer joins, the coalesced column contains the value of the non-**NULL** column if one of the two columns is always **NULL**. If neither or both columns are **NULL**, both common columns have the same value, so it doesn't matter which one is chosen as the value of the coalesced column. A simple way to interpret this is to consider that a coalesced column of an outer join is represented by the common column of the inner table of a **JOIN**. Suppose that the tables **t1(a, b)** and **t2(a, c)** have the following contents:

t1 t2

---- ----

1 x 2 z

2 y 3 w

Then, for this join, column **a** contains the values of **t1.a**:

mysql> **SELECT \* FROM t1 NATURAL LEFT JOIN t2;**

+------+------+------+

| a | b | c |

+------+------+------+

| 1 | x | NULL |

| 2 | y | z |

+------+------+------+

By contrast, for this join, column **a** contains the values of **t2.a**.

mysql> **SELECT \* FROM t1 NATURAL RIGHT JOIN t2;**

+------+------+------+

| a | c | b |

+------+------+------+

| 2 | z | y |

| 3 | w | NULL |

+------+------+------+

Compare those results to the otherwise equivalent queries with **JOIN ... ON**:

mysql> **SELECT \* FROM t1 LEFT JOIN t2 ON (t1.a = t2.a);**

+------+------+------+------+

| a | b | a | c |

+------+------+------+------+

| 1 | x | NULL | NULL |

| 2 | y | 2 | z |

+------+------+------+------+

mysql> **SELECT \* FROM t1 RIGHT JOIN t2 ON (t1.a = t2.a);**

+------+------+------+------+

| a | b | a | c |

+------+------+------+------+

| 2 | y | 2 | z |

| NULL | NULL | 3 | w |

+------+------+------+------+

A **USING** clause can be rewritten as an **ON** clause that compares corresponding columns. However, although **USING** and **ON** are similar, they are not quite the same. Consider the following two queries:

a LEFT JOIN b USING (c1, c2, c3)

a LEFT JOIN b ON a.c1 = b.c1 AND a.c2 = b.c2 AND a.c3 = b.c3

With respect to determining which rows satisfy the join condition, both joins are semantically identical.

With respect to determining which columns to display for **SELECT \*** expansion, the two joins are not semantically identical. The **USING** join selects the coalesced value of corresponding columns, whereas the **ON** join selects all columns from all tables. For the **USING** join, **SELECT \*** selects these values:

COALESCE(a.c1, b.c1), COALESCE(a.c2, b.c2), COALESCE(a.c3, b.c3)

For the **ON** join, **SELECT \*** selects these values:

a.c1, a.c2, a.c3, b.c1, b.c2, b.c3

With an inner join, [**COALESCE(a.c1, b.c1)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coalesce) is the same as either **a.c1** or **b.c1** because both columns have the same value. With an outer join (such as **LEFT JOIN**), one of the two columns can be **NULL**. That column is omitted from the result.

An **ON** clause can refer only to its operands.

Example:

CREATE TABLE t1 (i1 INT);

CREATE TABLE t2 (i2 INT);

CREATE TABLE t3 (i3 INT);

SELECT \* FROM t1 JOIN t2 ON (i1 = i3) JOIN t3;

The statement fails with an **Unknown column 'i3' in 'on clause'** error because **i3** is a column in **t3**, which is not an operand of the **ON** clause. To enable the join to be processed, rewrite the statement as follows:

SELECT \* FROM t1 JOIN t2 JOIN t3 ON (i1 = i3);

**JOIN** has higher precedence than the comma operator (**,**), so the join expression **t1, t2 JOIN t3** is interpreted as **(t1, (t2 JOIN t3))**, not as **((t1, t2) JOIN t3)**. This affects statements that use an **ON** clause because that clause can refer only to columns in the operands of the join, and the precedence affects interpretation of what those operands are.

Example:

CREATE TABLE t1 (i1 INT, j1 INT);

CREATE TABLE t2 (i2 INT, j2 INT);

CREATE TABLE t3 (i3 INT, j3 INT);

INSERT INTO t1 VALUES(1, 1);

INSERT INTO t2 VALUES(1, 1);

INSERT INTO t3 VALUES(1, 1);

SELECT \* FROM t1, t2 JOIN t3 ON (t1.i1 = t3.i3);

The **JOIN** takes precedence over the comma operator, so the operands for the **ON** clause are **t2** and **t3**. Because **t1.i1** is not a column in either of the operands, the result is an **Unknown column 't1.i1' in 'on clause'** error.

To enable the join to be processed, use either of these strategies:

Group the first two tables explicitly with parentheses so that the operands for the **ON** clause are **(t1, t2)** and **t3**:

SELECT \* FROM (t1, t2) JOIN t3 ON (t1.i1 = t3.i3);

Avoid the use of the comma operator and use **JOIN** instead:

SELECT \* FROM t1 JOIN t2 JOIN t3 ON (t1.i1 = t3.i3);

The same precedence interpretation also applies to statements that mix the comma operator with **INNER JOIN**, **CROSS JOIN**, **LEFT JOIN**, and **RIGHT JOIN**, all of which have higher precedence than the comma operator.

A MySQL extension compared to the SQL:2003 standard is that MySQL permits you to qualify the common (coalesced) columns of **NATURAL** or **USING** joins, whereas the standard disallows that.

#### 13.2.10.3 UNION Clause

SELECT ...

UNION [ALL | DISTINCT] SELECT ...

[UNION [ALL | DISTINCT] SELECT ...]

[**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) combines the result from multiple [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements into a single result set. Example:

mysql> **SELECT 1, 2;**

+---+---+

| 1 | 2 |

+---+---+

| 1 | 2 |

+---+---+

mysql> **SELECT 'a', 'b';**

+---+---+

| a | b |

+---+---+

| a | b |

+---+---+

mysql> **SELECT 1, 2 UNION SELECT 'a', 'b';**

+---+---+

| 1 | 2 |

+---+---+

| 1 | 2 |

| a | b |

+---+---+

[Result Set Column Names and Data Types](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union-result-set)

[TABLE in Unions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union-table)

[UNION DISTINCT and UNION ALL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union-distinct-all)

[ORDER BY and LIMIT in Unions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union-order-by-limit)

[UNION Restrictions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union-restrictions)

[UNION Handing in MySQL 8.0 Compared to MySQL 5.7](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union-8-0-versus-5-7)

##### Result Set Column Names and Data Types

The column names for a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) result set are taken from the column names of the first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement.

Selected columns listed in corresponding positions of each [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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 the data types of corresponding [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) columns do not match, the types and lengths of the columns in the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) result take into account the values retrieved by all the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements. For example, consider the following, where the column length is not constrained to the length of the value from the first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select):

mysql> **SELECT REPEAT('a',1) UNION SELECT REPEAT('b',20);**

+----------------------+

| REPEAT('a',1) |

+----------------------+

| a |

| bbbbbbbbbbbbbbbbbbbb |

+----------------------+

##### TABLE in Unions

Beginning with MySQL 8.0.19, you can also use a [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement or [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statement in a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) wherever you can employ the equivalent [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. Assume that tables **t1** and **t2** are created and populated as shown here:

CREATE TABLE t1 (x INT, y INT);

INSERT INTO t1 VALUES ROW(4,-2),ROW(5,9);

CREATE TABLE t2 (a INT, b INT);

INSERT INTO t2 VALUES ROW(1,2),ROW(3,4);

The preceding being the case, and disregarding the column names in the output of the queries beginning with [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values), all of the following **UNION** queries yield the same result:

SELECT \* FROM t1 UNION SELECT \* FROM t2;

TABLE t1 UNION SELECT \* FROM t2;

VALUES ROW(4,-2), ROW(5,9) UNION SELECT \* FROM t2;

SELECT \* FROM t1 UNION TABLE t2;

TABLE t1 UNION TABLE t2;

VALUES ROW(4,-2), ROW(5,9) UNION TABLE t2;

SELECT \* FROM t1 UNION VALUES ROW(4,-2),ROW(5,9);

TABLE t1 UNION VALUES ROW(4,-2),ROW(5,9);

VALUES ROW(4,-2), ROW(5,9) UNION VALUES ROW(4,-2),ROW(5,9);

To force the column names to be the same, wrap the [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) on the left hand side in a **SELECT** and use aliases, like this:

SELECT \* FROM (VALUES ROW(4,-2), ROW(5,9)) AS t(x,y)

UNION TABLE t2;

SELECT \* FROM (VALUES ROW(4,-2), ROW(5,9)) AS t(x,y)

UNION VALUES ROW(4,-2),ROW(5,9);

##### UNION DISTINCT and UNION ALL

By default, duplicate rows are removed from [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) results. The optional **DISTINCT** keyword has the same effect but makes it explicit. With the optional **ALL** keyword, duplicate-row removal does not occur and the result includes all matching rows from all the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements.

You can mix [**UNION ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) and [**UNION DISTINCT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) in the same query. Mixed [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) types are treated such that a **DISTINCT** union overrides any **ALL** union to its left. A **DISTINCT** union can be produced explicitly by using [**UNION DISTINCT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) or implicitly by using [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) with no following **DISTINCT** or **ALL** keyword.

In MySQL 8.0.19 and later, **UNION ALL** and **UNION DISTINCT** work the same way when one or more [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statements are used in the union.

##### ORDER BY and LIMIT in Unions

To apply an **ORDER BY** or **LIMIT** clause to an individual [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), parenthesize the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and place the clause inside the parentheses:

(SELECT a FROM t1 WHERE a=10 AND B=1 ORDER BY a LIMIT 10)

UNION

(SELECT a FROM t2 WHERE a=11 AND B=2 ORDER BY a LIMIT 10);

Use of **ORDER BY** for individual [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements implies nothing about the order in which the rows appear in the final result because [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) by default produces an unordered set of rows. Therefore, **ORDER BY** in this context typically is used in conjunction with **LIMIT**, to determine the subset of the selected rows to retrieve for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), even though it does not necessarily affect the order of those rows in the final [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) result. If **ORDER BY** appears without **LIMIT** in a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), it is optimized away because it has no effect in any case.

To use an **ORDER BY** or **LIMIT** clause to sort or limit the entire [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) result, parenthesize the individual [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements and place the **ORDER BY** or **LIMIT** after the last one:

(SELECT a FROM t1 WHERE a=10 AND B=1)

UNION

(SELECT a FROM t2 WHERE a=11 AND B=2)

ORDER BY a LIMIT 10;

A statement without parentheses is equivalent to one parenthesized as just shown.

Beginning with MySQL 8.0.19, you can use **ORDER BY** and **LIMIT** with [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) in unions in the same way as just shown, bearing in mind that **TABLE** does not support a **WHERE** clause.

This kind of **ORDER BY** cannot use column references that include a table name (that is, names in ***tbl\_name***.***col\_name*** format). Instead, provide a column alias in the first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement and refer to the alias in the **ORDER BY**. (Alternatively, refer to the column in the **ORDER BY** using its column position. However, use of column positions is deprecated.)

Also, if a column to be sorted is aliased, the **ORDER BY** clause must refer to the alias, not the column name. The first of the following statements is permitted, but the second fails with an **Unknown column 'a' in 'order clause'** error:

(SELECT a AS b FROM t) UNION (SELECT ...) ORDER BY b;

(SELECT a AS b FROM t) UNION (SELECT ...) ORDER BY a;

To cause rows in a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) result to consist of the sets of rows retrieved by each [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) one after the other, select an additional column in each [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) to use as a sort column and add an **ORDER BY** that sorts on that column following the last [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select):

(SELECT 1 AS sort\_col, col1a, col1b, ... FROM t1)

UNION

(SELECT 2, col2a, col2b, ... FROM t2) ORDER BY sort\_col;

To additionally maintain sort order within individual [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) results, add a secondary column to the **ORDER BY** clause:

(SELECT 1 AS sort\_col, col1a, col1b, ... FROM t1)

UNION

(SELECT 2, col2a, col2b, ... FROM t2) ORDER BY sort\_col, col1a;

Use of an additional column also enables you to determine which [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) each row comes from. Extra columns can provide other identifying information as well, such as a string that indicates a table name.

##### UNION Restrictions

In a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements are normal select statements, but with the following restrictions:

**HIGH\_PRIORITY** in the first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) has no effect. **HIGH\_PRIORITY** in any subsequent [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) produces a syntax error.

Only the last [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement can use an **INTO** clause. However, the entire [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) result is written to the **INTO** output destination.

As of MySQL 8.0.20, these two [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) variants containing **INTO** are deprecated and you should expect support for them to be removed in a future version of MySQL:

In the trailing query block of a query expression, use of **INTO** before **FROM** produces a warning. Example:

... UNION SELECT \* INTO OUTFILE '***file\_name***' FROM ***table\_name***;

In a parenthesized trailing block of a query expression, use of **INTO** (regardless of its position relative to **FROM**) produces a warning. Example:

... UNION (SELECT \* INTO OUTFILE '***file\_name***' FROM ***table\_name***);

Those variants are deprecated because they are confusing, as if they collect information from the named table rather than the entire query expression (the **UNION**).

[**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) queries with an aggregate function in an **ORDER BY** clause are rejected with an [**ER\_AGGREGATE\_ORDER\_FOR\_UNION**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_aggregate_order_for_union) error. Example:

SELECT 1 AS foo UNION SELECT 2 ORDER BY MAX(1);

##### UNION Handing in MySQL 8.0 Compared to MySQL 5.7

In MySQL 8.0, the parser rules for [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) were refactored to be more consistent (the same [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) syntax applies uniformly in each such context) and reduce duplication. Compared to MySQL 5.7, several user-visible effects resulted from this work, which may require rewriting of certain statements:

**NATURAL JOIN** permits an optional **INNER** keyword (**NATURAL INNER JOIN**), in compliance with standard SQL.

Right-deep joins without parentheses are permitted (for example, **... JOIN ... JOIN ... ON ... ON**), in compliance with standard SQL.

**STRAIGHT\_JOIN** now permits a **USING** clause, similar to other inner joins.

The parser accepts parentheses around query expressions. For example, **(SELECT ... UNION SELECT ...)** is permitted. See also [Section 13.2.10.4, “Parenthesized Query Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#parenthesized-query-expressions).

The parser better conforms to the documented permitted placement of the **SQL\_CACHE** and **SQL\_NO\_CACHE** query modifiers.

Left-hand nesting of unions, previously permitted only in subqueries, is now permitted in top-level statements. For example, this statement is now accepted as valid:

(SELECT 1 UNION SELECT 1) UNION SELECT 1;

Locking clauses (**FOR UPDATE**, **LOCK IN SHARE MODE**) are allowed only in non-**UNION** queries. This means that parentheses must be used for **SELECT** statements containing locking clauses. This statement is no longer accepted as valid:

SELECT 1 FOR UPDATE UNION SELECT 1 FOR UPDATE;

Instead, write the statement like this:

(SELECT 1 FOR UPDATE) UNION (SELECT 1 FOR UPDATE);

#### 13.2.10.4 Parenthesized Query Expressions

***parenthesized\_query\_expression***:

( ***query\_expression*** [***order\_by\_clause***] [***limit\_clause***] )

[***order\_by\_clause***]

[***limit\_clause***]

[***into\_clause***]

***query\_expression***:

***query\_block*** [UNION ***query\_block*** [UNION ***query\_block*** ...]]

[***order\_by\_clause***]

[***limit\_clause***]

[***into\_clause***]

***query\_block***:

SELECT ... (see [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select))

***order\_by\_clause***:

ORDER BY as for SELECT (see [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select))

***limit\_clause***:

LIMIT as for SELECT (see [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select))

***into\_clause***:

INTO as for SELECT (see [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select))

MySQL 8.0.22 and higher supports parenthesized query expressions according to the preceding syntax. At its simplest, a parenthesized query expression contains a single [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and no following optional clauses:

(SELECT 1);

(SELECT \* FROM INFORMATION\_SCHEMA.SCHEMATA WHERE SCHEMA\_NAME = 'mysql');

A parenthesized query expression can also contain a [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) comprising multiple [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, and end with any or all of the optional clauses:

mysql> **(SELECT 1 AS result UNION SELECT 2);**

+--------+

| result |

+--------+

| 1 |

| 2 |

+--------+

mysql> **(SELECT 1 AS result UNION SELECT 2) LIMIT 1;**

+--------+

| result |

+--------+

| 1 |

+--------+

mysql> **(SELECT 1 AS result UNION SELECT 2) LIMIT 1 OFFSET 1;**

+--------+

| result |

+--------+

| 2 |

+--------+

mysql> **(SELECT 1 AS result UNION SELECT 2)**

**ORDER BY result DESC LIMIT 1;**

+--------+

| result |

+--------+

| 2 |

+--------+

mysql> **(SELECT 1 AS result UNION SELECT 2)**

**ORDER BY result DESC LIMIT 1 OFFSET 1;**

+--------+

| result |

+--------+

| 1 |

+--------+

mysql> **(SELECT 1 AS result UNION SELECT 3 UNION SELECT 2)**

**ORDER BY result LIMIT 1 OFFSET 1 INTO @var;**

mysql> **SELECT @var;**

+------+

| @var |

+------+

| 2 |

+------+

Parenthesized query expressions are also used as query expressions, so a query expression, usually composed of query blocks, may also consist of parenthesized query expressions:

(SELECT \* FROM t1 ORDER BY a) UNION (SELECT \* FROM t2 ORDER BY b) ORDER BY z;

Query blocks may have trailing **ORDER BY** and **LIMIT** clauses, which are applied before the outer **UNION** and **ORDER BY** and **LIMIT**.

You cannot have a query block with a trailing **ORDER BY** or **LIMIT**, without wrapping it in parentheses, but parentheses may be used for enforcement in various ways:

To enforce **LIMIT** on each query block:

(SELECT 1 LIMIT 1) UNION (SELECT 2 LIMIT 1);

To enforce **LIMIT** on both query blocks and the entire query expression:

(SELECT 1 LIMIT 1) UNION (SELECT 2 LIMIT 1) LIMIT 1;

To enforce **LIMIT** on the entire query expression (with no parentheses):

SELECT 1 UNION SELECT 2 LIMIT 1;

Hybrid enforcement: **LIMIT** on the first query block and on the entire query expression:

(SELECT 1 LIMIT 1) UNION SELECT 2 LIMIT 1;

The syntax described in this section is subject to certain restrictions:

If **ORDER BY** occurs within a parenthesized query expression and also is applied in the outer query, the results are undefined and may change in a future version of MySQL. The same is true if **LIMIT** occurs within a parenthesized query expression and also is applied in the outer query.

A trailing **INTO** clause for a query expression is not permitted if there is another **INTO** clause inside parentheses.

Parenthesized query expressions do not permit multiple levels of **ORDER BY** or **LIMIT** operations. For example:

mysql> **(SELECT 'a' UNION SELECT 'b' LIMIT 1) LIMIT 2;**

ERROR 1235 (42000): This version of MySQL doesn't yet support 'parenthesized

query expression with more than one external level of ORDER/LIMIT operations'

### 13.2.11 Subqueries

[13.2.11.1 The Subquery as Scalar Operand](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#scalar-subqueries)

[13.2.11.2 Comparisons Using Subqueries](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#comparisons-using-subqueries)

[13.2.11.3 Subqueries with ANY, IN, or SOME](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#any-in-some-subqueries)

[13.2.11.4 Subqueries with ALL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#all-subqueries)

[13.2.11.5 Row Subqueries](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#row-subqueries)

[13.2.11.6 Subqueries with EXISTS or NOT EXISTS](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#exists-and-not-exists-subqueries)

[13.2.11.7 Correlated Subqueries](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#correlated-subqueries)

[13.2.11.8 Derived Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#derived-tables)

[13.2.11.9 Lateral Derived Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lateral-derived-tables)

[13.2.11.10 Subquery Errors](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subquery-errors)

[13.2.11.11 Optimizing Subqueries](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimizing-subqueries)

[13.2.11.12 Restrictions on Subqueries](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subquery-restrictions)

A subquery is a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement within another statement.

All subquery forms and operations that the SQL standard requires are supported, as well as a few features that are MySQL-specific.

Here is an example of a subquery:

SELECT \* FROM t1 WHERE column1 = (SELECT column1 FROM t2);

In this example, **SELECT \* FROM t1 ...** is the outer query (or outer statement), and **(SELECT column1 FROM t2)** is the subquery. We say that the subquery is nested within the outer query, and in fact it is possible to nest subqueries within other subqueries, to a considerable depth. A subquery must always appear within parentheses.

The main advantages of subqueries are:

They allow queries that are structured so that it is possible to isolate each part of a statement.

They provide alternative ways to perform operations that would otherwise require complex joins and unions.

Many people find subqueries more readable than complex joins or unions. Indeed, it was the innovation of subqueries that gave people the original idea of calling the early SQL “Structured Query Language.”

Here is an example statement that shows the major points about subquery syntax as specified by the SQL standard and supported in MySQL:

DELETE FROM t1

WHERE s11 > ANY

(SELECT COUNT(\*) /\* no hint \*/ FROM t2

WHERE NOT EXISTS

(SELECT \* FROM t3

WHERE ROW(5\*t2.s1,77)=

(SELECT 50,11\*s1 FROM t4 UNION SELECT 50,77 FROM

(SELECT \* FROM t5) AS t5)));

A subquery can return a scalar (a single value), a single row, a single column, or a table (one or more rows of one or more columns). These are called scalar, column, row, and table subqueries. Subqueries that return a particular kind of result often can be used only in certain contexts, as described in the following sections.

There are few restrictions on the type of statements in which subqueries can be used. A subquery can contain many of the keywords or clauses that an ordinary [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) can contain: **DISTINCT**, **GROUP BY**, **ORDER BY**, **LIMIT**, joins, index hints, [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) constructs, comments, functions, and so on.

Beginning with MySQL 8.0.19, [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) and [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statements can be used in subqueries. Subqueries using [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) are generally more verbose versions of subqueries that can be rewritten more compactly using set notation, or with [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) or [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) syntax; assuming that table **ts** is created using the statement [**CREATE TABLE ts VALUES ROW(2), ROW(4), ROW(6)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select), the statements shown here are all equivalent:

SELECT \* FROM tt

WHERE b > ANY (VALUES ROW(2), ROW(4), ROW(6));

SELECT \* FROM tt

WHERE b > ANY (2, 4, 6);

SELECT \* FROM tt

WHERE b > ANY (SELECT \* FROM ts);

SELECT \* FROM tt

WHERE b > ANY (TABLE ts);

Examples of [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) subqueries are shown in the sections that follow.

A subquery's outer statement can be any one of: [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable), or [**DO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#do).

For information about how the optimizer handles subqueries, see [Section 8.2.2, “Optimizing Subqueries, Derived Tables, View References, and Common Table Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#subquery-optimization). For a discussion of restrictions on subquery use, including performance issues for certain forms of subquery syntax, see [Section 13.2.11.12, “Restrictions on Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subquery-restrictions).

#### 13.2.11.1 The Subquery as Scalar Operand

In its simplest form, a subquery is a scalar subquery that returns a single value. A scalar subquery is a simple operand, and you can use it almost anywhere a single column value or literal is legal, and you can expect it to have those characteristics that all operands have: a data type, a length, an indication that it can be **NULL**, and so on. For example:

CREATE TABLE t1 (s1 INT, s2 CHAR(5) NOT NULL);

INSERT INTO t1 VALUES(100, 'abcde');

SELECT (SELECT s2 FROM t1);

The subquery in this [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) returns a single value (**'abcde'**) that has a data type of [**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), a length of 5, a character set and collation equal to the defaults in effect at [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) time, and an indication that the value in the column can be **NULL**. Nullability of the value selected by a scalar subquery is not copied because if the subquery result is empty, the result is **NULL**. For the subquery just shown, if **t1** were empty, the result would be **NULL** even though **s2** is **NOT NULL**.

There are a few contexts in which a scalar subquery cannot be used. If a statement permits only a literal value, you cannot use a subquery. For example, **LIMIT** requires literal integer arguments, and [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) requires a literal string file name. You cannot use subqueries to supply these values.

When you see examples in the following sections that contain the rather spartan construct **(SELECT column1 FROM t1)**, imagine that your own code contains much more diverse and complex constructions.

Suppose that we make two tables:

CREATE TABLE t1 (s1 INT);

INSERT INTO t1 VALUES (1);

CREATE TABLE t2 (s1 INT);

INSERT INTO t2 VALUES (2);

Then perform a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select):

SELECT (SELECT s1 FROM t2) FROM t1;

The result is **2** because there is a row in **t2** containing a column **s1** that has a value of **2**.

In MySQL 8.0.19 and later, the preceding query can also be written like this, using [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table):

SELECT (TABLE t2) FROM t1;

A scalar subquery can be part of an expression, but remember the parentheses, even if the subquery is an operand that provides an argument for a function. For example:

SELECT UPPER((SELECT s1 FROM t1)) FROM t2;

The same result can be obtained in MySQL 8.0.19 and later using **SELECT UPPER((TABLE t1)) FROM t2**.

#### 13.2.11.2 Comparisons Using Subqueries

The most common use of a subquery is in the form:

***non\_subquery\_operand*** ***comparison\_operator*** (***subquery***)

Where ***comparison\_operator*** is one of these operators:

= > < >= <= <> != <=>

For example:

... WHERE 'a' = (SELECT column1 FROM t1)

MySQL also permits this construct:

***non\_subquery\_operand*** LIKE (***subquery***)

At one time the only legal place for a subquery was on the right side of a comparison, and you might still find some old DBMSs that insist on this.

Here is an example of a common-form subquery comparison that you cannot do with a join. It finds all the rows in table **t1** for which the **column1** value is equal to a maximum value in table **t2**:

SELECT \* FROM t1

WHERE column1 = (SELECT MAX(column2) FROM t2);

Here is another example, which again is impossible with a join because it involves aggregating for one of the tables. It finds all rows in table **t1** containing a value that occurs twice in a given column:

SELECT \* FROM t1 AS t

WHERE 2 = (SELECT COUNT(\*) FROM t1 WHERE t1.id = t.id);

For a comparison of the subquery to a scalar, the subquery must return a scalar. For a comparison of the subquery to a row constructor, the subquery must be a row subquery that returns a row with the same number of values as the row constructor. See [Section 13.2.11.5, “Row Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#row-subqueries).

#### 13.2.11.3 Subqueries with ANY, IN, or SOME

Syntax:

***operand*** ***comparison\_operator*** ANY (***subquery***)

***operand*** IN (***subquery***)

***operand*** ***comparison\_operator*** SOME (***subquery***)

Where ***comparison\_operator*** is one of these operators:

= > < >= <= <> !=

The **ANY** keyword, which must follow a comparison operator, means “return **TRUE** if the comparison is **TRUE** for **ANY** of the values in the column that the subquery returns.” For example:

SELECT s1 FROM t1 WHERE s1 > ANY (SELECT s1 FROM t2);

Suppose that there is a row in table **t1** containing **(10)**. The expression is **TRUE** if table **t2** contains **(21,14,7)** because there is a value **7** in **t2** that is less than **10**. The expression is **FALSE** if table **t2** contains **(20,10)**, or if table **t2** is empty. The expression is unknown (that is, **NULL**) if table **t2** contains **(NULL,NULL,NULL)**.

When used with a subquery, the word **IN** is an alias for **= ANY**. Thus, these two statements are the same:

SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);

SELECT s1 FROM t1 WHERE s1 IN (SELECT s1 FROM t2);

**IN** and **= ANY** are not synonyms when used with an expression list. **IN** can take an expression list, but **= ANY** cannot. See [Section 12.4.2, “Comparison Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#comparison-operators).

**NOT IN** is not an alias for **<> ANY**, but for **<> ALL**. See [Section 13.2.11.4, “Subqueries with ALL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#all-subqueries).

The word **SOME** is an alias for **ANY**. Thus, these two statements are the same:

SELECT s1 FROM t1 WHERE s1 <> ANY (SELECT s1 FROM t2);

SELECT s1 FROM t1 WHERE s1 <> SOME (SELECT s1 FROM t2);

Use of the word **SOME** is rare, but this example shows why it might be useful. To most people, the English phrase “a is not equal to any b” means “there is no b which is equal to a,” but that is not what is meant by the SQL syntax. The syntax means “there is some b to which a is not equal.” Using **<> SOME** instead helps ensure that everyone understands the true meaning of the query.

Beginning with MySQL 8.0.19, you can use [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) in a scalar **IN**, **ANY**, or **SOME** subquery provided the table contains only a single column. If **t2** has only one column, the statements shown previously in this section can be written as shown here, in each case substituting **TABLE t2** for **SELECT s1 FROM t2**:

SELECT s1 FROM t1 WHERE s1 > ANY (TABLE t2);

SELECT s1 FROM t1 WHERE s1 = ANY (TABLE t2);

SELECT s1 FROM t1 WHERE s1 IN (TABLE t2);

SELECT s1 FROM t1 WHERE s1 <> ANY (TABLE t2);

SELECT s1 FROM t1 WHERE s1 <> SOME (TABLE t2);

#### 13.2.11.4 Subqueries with ALL

Syntax:

***operand*** ***comparison\_operator*** ALL (***subquery***)

The word **ALL**, which must follow a comparison operator, means “return **TRUE** if the comparison is **TRUE** for **ALL** of the values in the column that the subquery returns.” For example:

SELECT s1 FROM t1 WHERE s1 > ALL (SELECT s1 FROM t2);

Suppose that there is a row in table **t1** containing **(10)**. The expression is **TRUE** if table **t2** contains **(-5,0,+5)** because **10** is greater than all three values in **t2**. The expression is **FALSE** if table **t2** contains **(12,6,NULL,-100)** because there is a single value **12** in table **t2** that is greater than **10**. The expression is unknown (that is, **NULL**) if table **t2** contains **(0,NULL,1)**.

Finally, the expression is **TRUE** if table **t2** is empty. So, the following expression is **TRUE** when table **t2** is empty:

SELECT \* FROM t1 WHERE 1 > ALL (SELECT s1 FROM t2);

But this expression is **NULL** when table **t2** is empty:

SELECT \* FROM t1 WHERE 1 > (SELECT s1 FROM t2);

In addition, the following expression is **NULL** when table **t2** is empty:

SELECT \* FROM t1 WHERE 1 > ALL (SELECT MAX(s1) FROM t2);

In general, tables containing ***NULL*** values and empty tables are “edge cases.” When writing subqueries, always consider whether you have taken those two possibilities into account.

**NOT IN** is an alias for **<> ALL**. Thus, these two statements are the same:

SELECT s1 FROM t1 WHERE s1 <> ALL (SELECT s1 FROM t2);

SELECT s1 FROM t1 WHERE s1 NOT IN (SELECT s1 FROM t2);

MySQL 8.0.19 supports the [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement. As with **IN**, **ANY**, and **SOME**, you can use **TABLE** with **ALL** and **NOT IN** provided that the following two conditions are met:

The table in the subquery contains only one column

The subquery does not depend on a column expression

For example, assuming that table **t2** consists of a single column, the last two statements shown previously can be written using **TABLE t2** like this:

SELECT s1 FROM t1 WHERE s1 <> ALL (TABLE t2);

SELECT s1 FROM t1 WHERE s1 NOT IN (TABLE t2);

A query such as **SELECT \* FROM t1 WHERE 1 > ALL (SELECT MAX(s1) FROM t2);** cannot be written using **TABLE t2** because the subquery depends on a column expression.

#### 13.2.11.5 Row Subqueries

Scalar or column subqueries return a single value or a column of values. A row subquery is a subquery variant that returns a single row and can thus return more than one column value. Legal operators for row subquery comparisons are:

= > < >= <= <> != <=>

Here are two examples:

SELECT \* FROM t1

WHERE (col1,col2) = (SELECT col3, col4 FROM t2 WHERE id = 10);

SELECT \* FROM t1

WHERE ROW(col1,col2) = (SELECT col3, col4 FROM t2 WHERE id = 10);

For both queries, if the table **t2** contains a single row with **id = 10**, the subquery returns a single row. If this row has **col3** and **col4** values equal to the **col1** and **col2** values of any rows in **t1**, the **WHERE** expression is **TRUE** and each query returns those **t1** rows. If the **t2** row **col3** and **col4** values are not equal the **col1** and **col2** values of any **t1** row, the expression is **FALSE** and the query returns an empty result set. The expression is unknown (that is, **NULL**) if the subquery produces no rows. An error occurs if the subquery produces multiple rows because a row subquery can return at most one row.

For information about how each operator works for row comparisons, see [Section 12.4.2, “Comparison Functions and Operators”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#comparison-operators).

The expressions **(1,2)** and **ROW(1,2)** are sometimes called row constructors. The two are equivalent. The row constructor and the row returned by the subquery must contain the same number of values.

A row constructor is used for comparisons with subqueries that return two or more columns. When a subquery returns a single column, this is regarded as a scalar value and not as a row, so a row constructor cannot be used with a subquery that does not return at least two columns. Thus, the following query fails with a syntax error:

SELECT \* FROM t1 WHERE ROW(1) = (SELECT column1 FROM t2)

Row constructors are legal in other contexts. For example, the following two statements are semantically equivalent (and are handled in the same way by the optimizer):

SELECT \* FROM t1 WHERE (column1,column2) = (1,1);

SELECT \* FROM t1 WHERE column1 = 1 AND column2 = 1;

The following query answers the request, “find all rows in table **t1** that also exist in table **t2**”:

SELECT column1,column2,column3

FROM t1

WHERE (column1,column2,column3) IN

(SELECT column1,column2,column3 FROM t2);

For more information about the optimizer and row constructors, see [Section 8.2.1.22, “Row Constructor Expression Optimization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#row-constructor-optimization)

#### 13.2.11.6 Subqueries with EXISTS or NOT EXISTS

If a subquery returns any rows at all, **EXISTS *subquery*** is **TRUE**, and **NOT EXISTS *subquery*** is **FALSE**. For example:

SELECT column1 FROM t1 WHERE EXISTS (SELECT \* FROM t2);

Traditionally, an **EXISTS** subquery starts with **SELECT \***, but it could begin with **SELECT 5** or **SELECT column1** or anything at all. MySQL ignores the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) list in such a subquery, so it makes no difference.

For the preceding example, if **t2** contains any rows, even rows with nothing but **NULL** values, the **EXISTS** condition is **TRUE**. This is actually an unlikely example because a **[NOT] EXISTS** subquery almost always contains correlations. Here are some more realistic examples:

What kind of store is present in one or more cities?

SELECT DISTINCT store\_type FROM stores

WHERE EXISTS (SELECT \* FROM cities\_stores

WHERE cities\_stores.store\_type = stores.store\_type);

What kind of store is present in no cities?

SELECT DISTINCT store\_type FROM stores

WHERE NOT EXISTS (SELECT \* FROM cities\_stores

WHERE cities\_stores.store\_type = stores.store\_type);

What kind of store is present in all cities?

SELECT DISTINCT store\_type FROM stores s1

WHERE NOT EXISTS (

SELECT \* FROM cities WHERE NOT EXISTS (

SELECT \* FROM cities\_stores

WHERE cities\_stores.city = cities.city

AND cities\_stores.store\_type = stores.store\_type));

The last example is a double-nested **NOT EXISTS** query. That is, it has a **NOT EXISTS** clause within a **NOT EXISTS** clause. Formally, it answers the question “does a city exist with a store that is not in **Stores**”? But it is easier to say that a nested **NOT EXISTS** answers the question “is ***x*** **TRUE** for all ***y***?”

In MySQL 8.0.19 and later, you can also use **NOT EXISTS** or **NOT EXISTS** with [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) in the subquery, like this:

SELECT column1 FROM t1 WHERE EXISTS (TABLE t2);

The results are the same as when using **SELECT \*** with no **WHERE** clause in the subquery.

#### 13.2.11.7 Correlated Subqueries

A correlated subquery is a subquery that contains a reference to a table that also appears in the outer query. For example:

SELECT \* FROM t1

WHERE column1 = ANY (SELECT column1 FROM t2

WHERE t2.column2 = t1.column2);

Notice that the subquery contains a reference to a column of **t1**, even though the subquery's **FROM** clause does not mention a table **t1**. So, MySQL looks outside the subquery, and finds **t1** in the outer query.

Suppose that table **t1** contains a row where **column1 = 5** and **column2 = 6**; meanwhile, table **t2** contains a row where **column1 = 5** and **column2 = 7**. The simple expression **... WHERE column1 = ANY (SELECT column1 FROM t2)** would be **TRUE**, but in this example, the **WHERE** clause within the subquery is **FALSE** (because **(5,6)** is not equal to **(5,7)**), so the expression as a whole is **FALSE**.

***Scoping rule:*** MySQL evaluates from inside to outside. For example:

SELECT column1 FROM t1 AS x

WHERE x.column1 = (SELECT column1 FROM t2 AS x

WHERE x.column1 = (SELECT column1 FROM t3

WHERE x.column2 = t3.column1));

In this statement, **x.column2** must be a column in table **t2** because **SELECT column1 FROM t2 AS x ...** renames **t2**. It is not a column in table **t1** because **SELECT column1 FROM t1 ...** is an outer query that is farther out.

Beginning with MySQL 8.0.24, the optimizer can transform a correlated scalar subquery to a derived table. Consider the query shown here:

SELECT \* FROM t1

WHERE ( SELECT a FROM t2

WHERE t2.a=t1.a ) > 0;

To avoid materializing several times for a given derived table, we can instead materialize—once—a derived table which adds a grouping on the join column from the table referenced in the inner query (**t2.a**) and then an outer join on the lifted predicate (**t1.a = derived.a**) in order to select the correct group to match up with the outer row. (If the subquery already has an explicit grouping, the extra grouping is added to the end of the grouping list.) The query previously shown can thus be rewritten like this:

SELECT t1.\* FROM t1

LEFT OUTER JOIN

(SELECT a, COUNT(\*) AS ct FROM t2 GROUP BY a) AS derived

ON t1.a = derived.a

AND

REJECT\_IF(

(ct > 1),

"ERROR 1242 (21000): Subquery returns more than 1 row"

)

WHERE derived.a > 0;

In the rewritten query, **REJECT\_IF()** represents an internal function which tests a given condition (here, the comparison **ct > 1**) and raises a given error (in this case, [**ER\_SUBQUERY\_NO\_1\_ROW**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_subquery_no_1_row)) if the condition is true. This reflects the cardinality check that the optimizer performs as part of evaluating the **JOIN** or **WHERE** clause, prior to evaluating any lifted predicate, which is done only if the subquery does not return more than one row.

This type of transformation can be performed, provided the following conditions are met:

The subquery can be part of a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) list, **WHERE** condition, or **HAVING** condition, but cannot be part of a [**JOIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join) condition, and cannot contain a **LIMIT** or **OFFSET** clause. In addition, the subquery cannot contain any set operations such as [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union).

The **WHERE** clause may contain one or more predicates, combined with **AND**. If the **WHERE** clause contains an **OR** clause, it cannot be transformed. At least one of the **WHERE** clause predicates must be eligible for transformation, and none of them may reject transformation.

To be eligible for transformation, a **WHERE** clause predicate must be an equality predicate in which each operand should be a simple column reference. No other predicates—including other comparison predicates—are eligible for transformation. The predicate must employ the equality operator [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) for making the comparison; the null-safe [**≪=>**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal-to) operator is not supported in this context.

A **WHERE** clause predicate that contains only inner references is not eligible for transformation, since it can be evaluated before the grouping. A **WHERE** clause predicate that contains only outer references is eligible for transformation, even though it can be lifted up to the outer query block. This is made possible by adding a cardinality check without grouping in the derived table.

To be eligible, a **WHERE** clause predicate must have one operand that contains only inner references and one operand that contains only outer references. If the predicate is not eligible due to this rule, transformation of the query is rejected.

A correlated column can be present only in the subquery's **WHERE** clause (and not in the **SELECT** list, a **JOIN** or **ORDER BY** clause, a **GROUP BY** list, or a **HAVING** clause). Nor can there be any correlated column inside a derived table in the subquery's **FROM** list.

A correlated column can not be contained in an aggregate function's list of arguments.

A correlated column must be resolved in the query block directly containing the subquery being considered for transformation.

A correlated column cannot be present in a nested scalar subquery in the **WHERE** clause.

The subquery cannot contain any window functions, and must not contain any aggregate function which aggregates in a query block outer to the subquery. A [**COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count) aggregate function, if contained in the **SELECT** list element of the subquery, must be at the topmost level, and cannot be part of an expression.

See also [Section 13.2.11.8, “Derived Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#derived-tables).

#### 13.2.11.8 Derived Tables

This section discusses general characteristics of derived tables. For information about lateral derived tables preceded by the **LATERAL** keyword, see [Section 13.2.11.9, “Lateral Derived Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lateral-derived-tables).

A derived table is an expression that generates a table within the scope of a query **FROM** clause. For example, a subquery in a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement **FROM** clause is a derived table:

SELECT ... FROM (***subquery***) [AS] ***tbl\_name*** ...

The [**JSON\_TABLE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-table) function generates a table and provides another way to create a derived table:

SELECT \* FROM JSON\_TABLE(***arg\_list***) [AS] ***tbl\_name*** ...

The **[AS] *tbl\_name*** clause is mandatory because every table in a **FROM** clause must have a name. Any columns in the derived table must have unique names. Alternatively, ***tbl\_name*** may be followed by a parenthesized list of names for the derived table columns:

SELECT ... FROM (***subquery***) [AS] ***tbl\_name*** (***col\_list***) ...

The number of column names must be the same as the number of table columns.

For the sake of illustration, assume that you have this table:

CREATE TABLE t1 (s1 INT, s2 CHAR(5), s3 FLOAT);

Here is how to use a subquery in the **FROM** clause, using the example table:

INSERT INTO t1 VALUES (1,'1',1.0);

INSERT INTO t1 VALUES (2,'2',2.0);

SELECT sb1,sb2,sb3

FROM (SELECT s1 AS sb1, s2 AS sb2, s3\*2 AS sb3 FROM t1) AS sb

WHERE sb1 > 1;

Result:

+------+------+------+

| sb1 | sb2 | sb3 |

+------+------+------+

| 2 | 2 | 4 |

+------+------+------+

Here is another example: Suppose that you want to know the average of a set of sums for a grouped table. This does not work:

SELECT AVG(SUM(column1)) FROM t1 GROUP BY column1;

However, this query provides the desired information:

SELECT AVG(sum\_column1)

FROM (SELECT SUM(column1) AS sum\_column1

FROM t1 GROUP BY column1) AS t1;

Notice that the column name used within the subquery (**sum\_column1**) is recognized in the outer query.

The column names for a derived table come from its select list:

mysql> **SELECT \* FROM (SELECT 1, 2, 3, 4) AS dt;**

+---+---+---+---+

| 1 | 2 | 3 | 4 |

+---+---+---+---+

| 1 | 2 | 3 | 4 |

+---+---+---+---+

To provide column names explicitly, follow the derived table name with a parenthesized list of column names:

mysql> **SELECT \* FROM (SELECT 1, 2, 3, 4) AS dt (a, b, c, d);**

+---+---+---+---+

| a | b | c | d |

+---+---+---+---+

| 1 | 2 | 3 | 4 |

+---+---+---+---+

A derived table can return a scalar, column, row, or table.

Derived tables are subject to these restrictions:

A derived table cannot contain references to other tables of the same [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) (use a **LATERAL** derived table for that; see [Section 13.2.11.9, “Lateral Derived Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lateral-derived-tables)).

Prior to MySQL 8.0.14, a derived table cannot contain outer references. This is a MySQL restriction that is lifted in MySQL 8.0.14, not a restriction of the SQL standard. For example, the derived table **dt** in the following query contains a reference **t1.b** to the table **t1** in the outer query:

SELECT \* FROM t1

WHERE t1.d > (SELECT AVG(dt.a)

FROM (SELECT SUM(t2.a) AS a

FROM t2

WHERE t2.b = t1.b GROUP BY t2.c) dt

WHERE dt.a > 10);

The query is valid in MySQL 8.0.14 and higher. Before 8.0.14, it produces an error: **Unknown column 't1.b' in 'where clause'**

The optimizer determines information about derived tables in such a way that [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) does not need to materialize them. See [Section 8.2.2.4, “Optimizing Derived Tables, View References, and Common Table Expressions with Merging or Materialization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#derived-table-optimization).

It is possible under certain circumstances that using [**EXPLAIN SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) modifies table data. This can occur if the outer query accesses any tables and an inner query invokes a stored function that changes one or more rows of a table. Suppose that there are two tables **t1** and **t2** in database **d1**, and a stored function **f1** that modifies **t2**, created as shown here:

CREATE DATABASE d1;

USE d1;

CREATE TABLE t1 (c1 INT);

CREATE TABLE t2 (c1 INT);

CREATE FUNCTION f1(p1 INT) RETURNS INT

BEGIN

INSERT INTO t2 VALUES (p1);

RETURN p1;

END;

Referencing the function directly in an [**EXPLAIN SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) has no effect on **t2**, as shown here:

mysql> **SELECT \* FROM t2;**

Empty set (0.02 sec)

mysql> **EXPLAIN SELECT f1(5)\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: NULL

partitions: NULL

type: NULL

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: NULL

filtered: NULL

Extra: No tables used

1 row in set (0.01 sec)

mysql> **SELECT \* FROM t2;**

Empty set (0.01 sec)

This is because the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement did not reference any tables, as can be seen in the **table** and **Extra** columns of the output. This is also true of the following nested [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select):

mysql> **EXPLAIN SELECT NOW() AS a1, (SELECT f1(5)) AS a2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: PRIMARY

table: NULL

type: NULL

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: NULL

filtered: NULL

Extra: No tables used

1 row in set, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS;**

+-------+------+------------------------------------------+

| Level | Code | Message |

+-------+------+------------------------------------------+

| Note | 1249 | Select 2 was reduced during optimization |

+-------+------+------------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT \* FROM t2;**

Empty set (0.00 sec)

However, if the outer [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) references any tables, the optimizer executes the statement in the subquery as well, with the result that **t2** is modified:

mysql> **EXPLAIN SELECT \* FROM t1 AS a1, (SELECT f1(5)) AS a2\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: PRIMARY

table: <derived2>

partitions: NULL

type: system

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: 1

filtered: 100.00

Extra: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: PRIMARY

table: a1

partitions: NULL

type: ALL

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: 1

filtered: 100.00

Extra: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 2

select\_type: DERIVED

table: NULL

partitions: NULL

type: NULL

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: NULL

filtered: NULL

Extra: No tables used

3 rows in set (0.00 sec)

mysql> **SELECT \* FROM t2;**

+------+

| c1 |

+------+

| 5 |

+------+

1 row in set (0.00 sec)

This also means that an [**EXPLAIN SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) statement such as the one shown here may take a long time to execute because the [**BENCHMARK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_benchmark) function is executed once for each row in **t1**:

EXPLAIN SELECT \* FROM t1 AS a1, (SELECT BENCHMARK(1000000, MD5(NOW())));

The derived table optimization can also be employed with many correlated (scalar) subqueries (MySQL 8.0.24 and later). For more information and examples, see [Section 13.2.11.7, “Correlated Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#correlated-subqueries).

#### 13.2.11.9 Lateral Derived Tables

A derived table cannot normally refer to (depend on) columns of preceding tables in the same **FROM** clause. As of MySQL 8.0.14, a derived table may be defined as a lateral derived table to specify that such references are permitted.

Nonlateral derived tables are specified using the syntax discussed in [Section 13.2.11.8, “Derived Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#derived-tables). The syntax for a lateral derived table is the same as for a nonlateral derived table except that the keyword **LATERAL** is specified before the derived table specification. The **LATERAL** keyword must precede each table to be used as a lateral derived table.

Lateral derived tables are subject to these restrictions:

A lateral derived table can occur only in a **FROM** clause, either in a list of tables separated with commas or in a join specification (**JOIN**, **INNER JOIN**, **CROSS JOIN**, **LEFT [OUTER] JOIN**, or **RIGHT [OUTER] JOIN**).

If a lateral derived table is in the right operand of a join clause and contains a reference to the left operand, the join operation must be an **INNER JOIN**, **CROSS JOIN**, or **LEFT [OUTER] JOIN**.

If the table is in the left operand and contains a reference to the right operand, the join operation must be an **INNER JOIN**, **CROSS JOIN**, or **RIGHT [OUTER] JOIN**.

If a lateral derived table references an aggregate function, the function's aggregation query cannot be the one that owns the **FROM** clause in which the lateral derived table occurs.

Per the SQL standard, a table function has an implicit **LATERAL**, so it behaves as in MySQL 8.0 versions prior to 8.0.14. However, per the standard, the **LATERAL** word is not allowed before [**JSON\_TABLE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-table), even though it is implicit.

The following discussion shows how lateral derived tables make possible certain SQL operations that cannot be done with nonlateral derived tables or that require less-efficient workarounds.

Suppose that we want to solve this problem: Given a table of people in a sales force (where each row describes a member of the sales force), and a table of all sales (where each row describes a sale: salesperson, customer, amount, date), determine the size and customer of the largest sale for each salesperson. This problem can be approached two ways.

First approach to solving the problem: For each salesperson, calculate the maximum sale size, and also find the customer who provided this maximum. In MySQL, that can be done like this:

SELECT

salesperson.name,

-- find maximum sale size for this salesperson

(SELECT MAX(amount) AS amount

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id)

AS amount,

-- find customer for this maximum size

(SELECT customer\_name

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id

AND all\_sales.amount =

-- find maximum size, again

(SELECT MAX(amount) AS amount

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id))

AS customer\_name

FROM

salesperson;

That query is inefficient because it calculates the maximum size twice per salesperson (once in the first subquery and once in the second).

We can try to achieve an efficiency gain by calculating the maximum once per salesperson and “caching” it in a derived table, as shown by this modified query:

SELECT

salesperson.name,

max\_sale.amount,

max\_sale\_customer.customer\_name

FROM

salesperson,

-- calculate maximum size, cache it in transient derived table max\_sale

(SELECT MAX(amount) AS amount

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id)

AS max\_sale,

-- find customer, reusing cached maximum size

(SELECT customer\_name

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id

AND all\_sales.amount =

-- the cached maximum size

max\_sale.amount)

AS max\_sale\_customer;

However, the query is illegal in SQL-92 because derived tables cannot depend on other tables in the same **FROM** clause. Derived tables must be constant over the query's duration, not contain references to columns of other **FROM** clause tables. As written, the query produces this error:

ERROR 1054 (42S22): Unknown column 'salesperson.id' in 'where clause'

In SQL:1999, the query becomes legal if the derived tables are preceded by the **LATERAL** keyword (which means “this derived table depends on previous tables on its left side”):

SELECT

salesperson.name,

max\_sale.amount,

max\_sale\_customer.customer\_name

FROM

salesperson,

-- calculate maximum size, cache it in transient derived table max\_sale

LATERAL

(SELECT MAX(amount) AS amount

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id)

AS max\_sale,

-- find customer, reusing cached maximum size

LATERAL

(SELECT customer\_name

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id

AND all\_sales.amount =

-- the cached maximum size

max\_sale.amount)

AS max\_sale\_customer;

A lateral derived table need not be constant and is brought up to date each time a new row from a preceding table on which it depends is processed by the top query.

Second approach to solving the problem: A different solution could be used if a subquery in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) list could return multiple columns:

SELECT

salesperson.name,

-- find maximum size and customer at same time

(SELECT amount, customer\_name

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id

ORDER BY amount DESC LIMIT 1)

FROM

salesperson;

That is efficient but illegal. It does not work because such subqueries can return only a single column:

ERROR 1241 (21000): Operand should contain 1 column(s)

One attempt at rewriting the query is to select multiple columns from a derived table:

SELECT

salesperson.name,

max\_sale.amount,

max\_sale.customer\_name

FROM

salesperson,

-- find maximum size and customer at same time

(SELECT amount, customer\_name

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id

ORDER BY amount DESC LIMIT 1)

AS max\_sale;

However, that also does not work. The derived table is dependent on the **salesperson** table and thus fails without **LATERAL**:

ERROR 1054 (42S22): Unknown column 'salesperson.id' in 'where clause'

Adding the **LATERAL** keyword makes the query legal:

SELECT

salesperson.name,

max\_sale.amount,

max\_sale.customer\_name

FROM

salesperson,

-- find maximum size and customer at same time

LATERAL

(SELECT amount, customer\_name

FROM all\_sales

WHERE all\_sales.salesperson\_id = salesperson.id

ORDER BY amount DESC LIMIT 1)

AS max\_sale;

In short, **LATERAL** is the efficient solution to all drawbacks in the two approaches just discussed.

#### 13.2.11.10 Subquery Errors

There are some errors that apply only to subqueries. This section describes them.

Unsupported subquery syntax:

ERROR 1235 (ER\_NOT\_SUPPORTED\_YET)

SQLSTATE = 42000

Message = "This version of MySQL doesn't yet support

'LIMIT & IN/ALL/ANY/SOME subquery'"

This means that MySQL does not support statements like the following:

SELECT \* FROM t1 WHERE s1 IN (SELECT s2 FROM t2 ORDER BY s1 LIMIT 1)

Incorrect number of columns from subquery:

ERROR 1241 (ER\_OPERAND\_COL)

SQLSTATE = 21000

Message = "Operand should contain 1 column(s)"

This error occurs in cases like this:

SELECT (SELECT column1, column2 FROM t2) FROM t1;

You may use a subquery that returns multiple columns, if the purpose is row comparison. In other contexts, the subquery must be a scalar operand. See [Section 13.2.11.5, “Row Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#row-subqueries).

Incorrect number of rows from subquery:

ERROR 1242 (ER\_SUBSELECT\_NO\_1\_ROW)

SQLSTATE = 21000

Message = "Subquery returns more than 1 row"

This error occurs for statements where the subquery must return at most one row but returns multiple rows. Consider the following example:

SELECT \* FROM t1 WHERE column1 = (SELECT column1 FROM t2);

If **SELECT column1 FROM t2** returns just one row, the previous query works. If the subquery returns more than one row, error 1242 occurs. In that case, the query should be rewritten as:

SELECT \* FROM t1 WHERE column1 = ANY (SELECT column1 FROM t2);

Incorrectly used table in subquery:

Error 1093 (ER\_UPDATE\_TABLE\_USED)

SQLSTATE = HY000

Message = "You can't specify target table 'x'

for update in FROM clause"

This error occurs in cases such as the following, which attempts to modify a table and select from the same table in the subquery:

UPDATE t1 SET column2 = (SELECT MAX(column1) FROM t1);

You can use a common table expression or derived table to work around this. See [Section 13.2.11.12, “Restrictions on Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subquery-restrictions).

In MySQL 8.0.19 and later, all of the errors described in this section also apply when using [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) in subqueries.

For transactional storage engines, the failure of a subquery causes the entire statement to fail. For nontransactional storage engines, data modifications made before the error was encountered are preserved.

#### 13.2.11.11 Optimizing Subqueries

Development is ongoing, so no optimization tip is reliable for the long term. The following list provides some interesting tricks that you might want to play with. See also [Section 8.2.2, “Optimizing Subqueries, Derived Tables, View References, and Common Table Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#subquery-optimization).

Move clauses from outside to inside the subquery. For example, use this query:

SELECT \* FROM t1

WHERE s1 IN (SELECT s1 FROM t1 UNION ALL SELECT s1 FROM t2);

Instead of this query:

SELECT \* FROM t1

WHERE s1 IN (SELECT s1 FROM t1) OR s1 IN (SELECT s1 FROM t2);

For another example, use this query:

SELECT (SELECT column1 + 5 FROM t1) FROM t2;

Instead of this query:

SELECT (SELECT column1 FROM t1) + 5 FROM t2;

#### 13.2.11.12 Restrictions on Subqueries

In general, you cannot modify a table and select from the same table in a subquery. For example, this limitation applies to statements of the following forms:

DELETE FROM t WHERE ... (SELECT ... FROM t ...);

UPDATE t ... WHERE col = (SELECT ... FROM t ...);

{INSERT|REPLACE} INTO t (SELECT ... FROM t ...);

Exception: The preceding prohibition does not apply if for the modified table you are using a derived table and that derived table is materialized rather than merged into the outer query. (See [Section 8.2.2.4, “Optimizing Derived Tables, View References, and Common Table Expressions with Merging or Materialization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#derived-table-optimization).) Example:

UPDATE t ... WHERE col = (SELECT \* FROM (SELECT ... FROM t...) AS dt ...);

Here the result from the derived table is materialized as a temporary table, so the relevant rows in **t** have already been selected by the time the update to **t** takes place.

In general, you may be able to influence the optimizer to materialize a derived table by adding a [**NO\_MERGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-table-level) optimizer hint. See [Section 8.9.3, “Optimizer Hints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints).

Row comparison operations are only partially supported:

For ***expr* [NOT] IN *subquery***, ***expr*** can be an ***n***-tuple (specified using row constructor syntax) and the subquery can return rows of ***n***-tuples. The permitted syntax is therefore more specifically expressed as ***row\_constructor* [NOT] IN *table\_subquery***

For ***expr* *op* {ALL|ANY|SOME} *subquery***, ***expr*** must be a scalar value and the subquery must be a column subquery; it cannot return multiple-column rows.

In other words, for a subquery that returns rows of ***n***-tuples, this is supported:

(***expr\_1***, ..., ***expr\_n***) [NOT] IN ***table\_subquery***

But this is not supported:

(***expr\_1***, ..., ***expr\_n***) ***op*** {ALL|ANY|SOME} ***subquery***

The reason for supporting row comparisons for **IN** but not for the others is that **IN** is implemented by rewriting it as a sequence of [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_equal) comparisons and [**AND**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_and) operations. This approach cannot be used for **ALL**, **ANY**, or **SOME**.

Prior to MySQL 8.0.14, subqueries in the **FROM** clause cannot be correlated subqueries. They are materialized in whole (evaluated to produce a result set) during query execution, so they cannot be evaluated per row of the outer query. The optimizer delays materialization until the result is needed, which may permit materialization to be avoided. See [Section 8.2.2.4, “Optimizing Derived Tables, View References, and Common Table Expressions with Merging or Materialization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#derived-table-optimization).

MySQL does not support **LIMIT** in subqueries for certain subquery operators:

mysql> **SELECT \* FROM t1**

**WHERE s1 IN (SELECT s2 FROM t2 ORDER BY s1 LIMIT 1);**

ERROR 1235 (42000): This version of MySQL doesn't yet support

'LIMIT & IN/ALL/ANY/SOME subquery'

See [Section 13.2.11.10, “Subquery Errors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subquery-errors).

MySQL permits a subquery to refer to a stored function that has data-modifying side effects such as inserting rows into a table. For example, if **f()** inserts rows, the following query can modify data:

SELECT ... WHERE x IN (SELECT f() ...);

This behavior is an extension to the SQL standard. In MySQL, it can produce nondeterministic results because **f()** might be executed a different number of times for different executions of a given query depending on how the optimizer chooses to handle it.

For statement-based or mixed-format replication, one implication of this indeterminism is that such a query can produce different results on the source and its replicas.

### 13.2.12 TABLE Statement

**[TABLE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "table" \o "13.2.12 TABLE Statement)** is a DML statement introduced in MySQL 8.0.19 which returns rows and columns of the named table.

TABLE ***table\_name*** [ORDER BY ***column\_name***] [LIMIT ***number*** [OFFSET ***number***]]

The [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statement in some ways acts like [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). Given the existance of a table named **t**, the following two statements produce identical output:

TABLE t;

SELECT \* FROM t;

You can order and limit the number of rows produced by [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) using **ORDER BY** and **LIMIT** clauses, respectively. These function identically to the same clauses when used with [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) (including an optional **OFFSET** clause with **LIMIT**), as you can see here:

mysql> **TABLE t;**

+----+----+

| a | b |

+----+----+

| 1 | 2 |

| 6 | 7 |

| 9 | 5 |

| 10 | -4 |

| 11 | -1 |

| 13 | 3 |

| 14 | 6 |

+----+----+

7 rows in set (0.00 sec)

mysql> **TABLE t ORDER BY b;**

+----+----+

| a | b |

+----+----+

| 10 | -4 |

| 11 | -1 |

| 1 | 2 |

| 13 | 3 |

| 9 | 5 |

| 14 | 6 |

| 6 | 7 |

+----+----+

7 rows in set (0.00 sec)

mysql> **TABLE t LIMIT 3;**

+---+---+

| a | b |

+---+---+

| 1 | 2 |

| 6 | 7 |

| 9 | 5 |

+---+---+

3 rows in set (0.00 sec)

mysql> **TABLE t ORDER BY b LIMIT 3;**

+----+----+

| a | b |

+----+----+

| 10 | -4 |

| 11 | -1 |

| 1 | 2 |

+----+----+

3 rows in set (0.00 sec)

mysql> **TABLE t ORDER BY b LIMIT 3 OFFSET 2;**

+----+----+

| a | b |

+----+----+

| 1 | 2 |

| 13 | 3 |

| 9 | 5 |

+----+----+

3 rows in set (0.00 sec)

[**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) differs from [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) in two key respects:

[**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) always displays all columns of the table.

[**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) does not allow for any arbitrary filtering of rows; that is, **TABLE** does not support any **WHERE** clause.

For limiting which table columns are returned, filtering rows beyond what can be accomplished using **ORDER BY** and **LIMIT**, or both, use [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

[**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) can be used with temporary tables.

**TABLE** can also be used in place of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) in a number of other constructs, including those listed here:

With [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), as shown here:

mysql> **TABLE t1;**

+---+----+

| a | b |

+---+----+

| 2 | 10 |

| 5 | 3 |

| 7 | 8 |

+---+----+

3 rows in set (0.00 sec)

mysql> **TABLE t2;**

+---+---+

| a | b |

+---+---+

| 1 | 2 |

| 3 | 4 |

| 6 | 7 |

+---+---+

3 rows in set (0.00 sec)

mysql> **TABLE t1 UNION TABLE t2;**

+---+----+

| a | b |

+---+----+

| 2 | 10 |

| 5 | 3 |

| 7 | 8 |

| 1 | 2 |

| 3 | 4 |

| 6 | 7 |

+---+----+

6 rows in set (0.00 sec)

The [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) just shown is equivalent to the following statement:

mysql> **SELECT \* FROM t1 UNION SELECT \* FROM t2;**

+---+----+

| a | b |

+---+----+

| 2 | 10 |

| 5 | 3 |

| 7 | 8 |

| 1 | 2 |

| 3 | 4 |

| 6 | 7 |

+---+----+

6 rows in set (0.00 sec)

**TABLE** can also be used together in unions with [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statements, or both. See [Section 13.2.10.3, “UNION Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union).

With **INTO** to populate user variables, and with **INTO OUTFILE** or **INTO DUMPFILE** to write table data to a file. See [Section 13.2.10.1, “SELECT ... INTO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into), for more specific information and examples.

In many cases where you can employ subqueries. Given any table **t1** with a column named **a**, and a second table **t2** having a single column, statements such as the following are possible:

SELECT \* FROM t1 WHERE a IN (TABLE t2);

Assuming that the single column of table **ts** is named **x**, the preceding is equivalent to each of the statements shown here (and produces exactly the same result in either case):

SELECT \* FROM t1 WHERE a IN (SELECT x FROM t2);

SELECT \* FROM t1 WHERE a IN (SELECT \* FROM t2);

See [Section 13.2.11, “Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#subqueries), for more information.

With [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) and [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statements, where you would otherwise use [**SELECT \***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). See [Section 13.2.6.1, “INSERT ... SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-select), for more information and examples.

[**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) can also be used in many cases in place of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) in [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) or [**CREATE VIEW ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view). See the descriptions of these statements for more information and examples.

### 13.2.13 UPDATE Statement

**[UPDATE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "update" \o "13.2.13 UPDATE Statement)** is a DML statement that modifies rows in a table.

An [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement can start with a [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause to define common table expressions accessible within the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). See [Section 13.2.15, “WITH (Common Table Expressions)”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with).

Single-table syntax:

UPDATE [LOW\_PRIORITY] [IGNORE] ***table\_reference***

SET ***assignment\_list***

[WHERE ***where\_condition***]

[ORDER BY ...]

[LIMIT ***row\_count***]

***value***:

{***expr*** | DEFAULT}

***assignment***:

***col\_name*** = ***value***

***assignment\_list***:

***assignment*** [, ***assignment***] ...

Multiple-table syntax:

UPDATE [LOW\_PRIORITY] [IGNORE] ***table\_references***

SET ***assignment\_list***

[WHERE ***where\_condition***]

For the single-table syntax, the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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.

For the multiple-table syntax, [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) updates rows in each table named in ***table\_references*** that satisfy the conditions. Each matching row is updated once, even if it matches the conditions multiple times. For multiple-table syntax, **ORDER BY** and **LIMIT** cannot be used.

For partitioned tables, both the single-single and multiple-table forms of this statement support the use of a **PARTITION** clause as part of a table reference. This option takes a list of one or more partitions or subpartitions (or both). Only the partitions (or subpartitions) listed are checked for matches, and a row that is not in any of these partitions or subpartitions is not updated, whether it satisfies the ***where\_condition*** or not.

**Note**

Unlike the case when using **PARTITION** with an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statement, an otherwise valid **UPDATE ... PARTITION** statement is considered successful even if no rows in the listed partitions (or subpartitions) match the ***where\_condition***.

For more information and examples, see [Section 24.5, “Partition Selection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-selection).

***where\_condition*** is an expression that evaluates to true for each row to be updated. For expression syntax, see [Section 9.5, “Expressions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#expressions).

***table\_references*** and ***where\_condition*** are specified as described in [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

You need the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege only for columns referenced in an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) that are actually updated. You need only the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for any columns that are read but not modified.

The [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement supports the following modifiers:

With the **LOW\_PRIORITY** modifier, execution of the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) is delayed until no other clients are reading from the table. This affects only storage engines that use only table-level locking (such as **MyISAM**, **MEMORY**, and **MERGE**).

With the **IGNORE** modifier, the update statement does not abort even if errors occur during the update. Rows for which duplicate-key conflicts occur on a unique key value are not updated. Rows updated to values that would cause data conversion errors are updated to the closest valid values instead. For more information, see [The Effect of IGNORE on Statement Execution](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#ignore-effect-on-execution).

**[UPDATE IGNORE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "update" \o "13.2.13 UPDATE Statement)** statements, including those having an **ORDER BY** clause, are flagged as unsafe for statement-based replication. (This is because the order in which the rows are updated determines which rows are ignored.) Such statements produce a warning in the error log when using statement-based mode and are written to the binary log using the row-based format when using **MIXED** mode. (Bug #11758262, Bug #50439) See [Section 17.2.1.3, “Determination of Safe and Unsafe Statements in Binary Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rbr-safe-unsafe), for more information.

If you access a column from the table to be updated in an expression, [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) uses the current value of the column. For example, the following statement sets **col1** to one more than its current value:

UPDATE t1 SET col1 = col1 + 1;

The second assignment in the following statement sets **col2** to the current (updated) **col1** value, not the original **col1** value. The result is that **col1** and **col2** have the same value. This behavior differs from standard SQL.

UPDATE t1 SET col1 = col1 + 1, col2 = col1;

Single-table [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) assignments are generally evaluated from left to right. For multiple-table updates, there is no guarantee that assignments are carried out in any particular order.

If you set a column to the value it currently has, MySQL notices this and does not update it.

If you update a column that has been declared **NOT NULL** by setting to **NULL**, an error occurs if strict SQL mode is enabled; otherwise, the column is set to the implicit default value for the column data type and the warning count is incremented. The implicit default value is **0** for numeric types, the empty string (**''**) for string types, and the “zero” value for date and time types. See [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults).

If a generated column is updated explicitly, the only permitted value is **DEFAULT**. For information about generated columns, see [Section 13.1.20.8, “CREATE TABLE and Generated Columns”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-generated-columns).

**[UPDATE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "update" \o "13.2.13 UPDATE Statement)** returns the number of rows that were actually changed. The [**mysql\_info()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-info.html) C API function returns the number of rows that were matched and updated and the number of warnings that occurred during the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update).

You can use **LIMIT *row\_count*** to restrict the scope of the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). A **LIMIT** clause is a rows-matched restriction. The statement stops as soon as it has found ***row\_count*** rows that satisfy the **WHERE** clause, whether or not they actually were changed.

If an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement includes an **ORDER BY** clause, the rows are updated in the order specified by the clause. This can be useful in certain situations that might otherwise result in an error. Suppose that a table **t** contains a column **id** that has a unique index. The following statement could fail with a duplicate-key error, depending on the order in which rows are updated:

UPDATE t SET id = id + 1;

For example, if the table contains 1 and 2 in the **id** column and 1 is updated to 2 before 2 is updated to 3, an error occurs. To avoid this problem, add an **ORDER BY** clause to cause the rows with larger **id** values to be updated before those with smaller values:

UPDATE t SET id = id + 1 ORDER BY id DESC;

You can also perform [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) operations covering multiple tables. However, you cannot use **ORDER BY** or **LIMIT** with a multiple-table [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). The ***table\_references*** clause lists the tables involved in the join. Its syntax is described in [Section 13.2.10.2, “JOIN Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join). Here is an example:

UPDATE items,month SET items.price=month.price

WHERE items.id=month.id;

The preceding example shows an inner join that uses the comma operator, but multiple-table [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements can use any type of join permitted in [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, such as **LEFT JOIN**.

If you use a multiple-table [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement involving **InnoDB** tables for which there are foreign key constraints, the MySQL optimizer might process tables in an order that differs from that of their parent/child relationship. In this case, the statement fails and rolls back. Instead, update a single table and rely on the **ON UPDATE** capabilities that **InnoDB** provides to cause the other tables to be modified accordingly. See [Section 13.1.20.5, “FOREIGN KEY Constraints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-foreign-keys).

You cannot update a table and select directly from the same table in a subquery. You can work around this by using a multi-table update in which one of the tables is derived from the table that you actually wish to update, and referring to the derived table using an alias. Suppose you wish to update a table named **items** which is defined using the statement shown here:

CREATE TABLE items (

id BIGINT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

wholesale DECIMAL(6,2) NOT NULL DEFAULT 0.00,

retail DECIMAL(6,2) NOT NULL DEFAULT 0.00,

quantity BIGINT NOT NULL DEFAULT 0

);

To reduce the retail price of any items for which the markup is 30% or greater and of which you have fewer than one hundred in stock, you might try to use an **UPDATE** statement such as the one following, which uses a subquery in the **WHERE** clause. As shown here, this statement does not work:

mysql> **UPDATE items**

> **SET retail = retail \* 0.9**

> **WHERE id IN**

> **(SELECT id FROM items**

> **WHERE retail / wholesale >= 1.3 AND quantity > 100);**

ERROR 1093 (HY000): You can't specify target table 'items' for update in FROM clause

Instead, you can employ a multi-table update in which the subquery is moved into the list of tables to be updated, using an alias to reference it in the outermost **WHERE** clause, like this:

UPDATE items,

(SELECT id FROM items

WHERE id IN

(SELECT id FROM items

WHERE retail / wholesale >= 1.3 AND quantity < 100))

AS discounted

SET items.retail = items.retail \* 0.9

WHERE items.id = discounted.id;

Because the optimizer tries by default to merge the derived table **discounted** into the outermost query block, this works only if you force materialization of the derived table. You can do this by setting the [**derived\_merge**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optflag_derived-merge) flag of the [**optimizer\_switch**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_optimizer_switch) system variable to **off** before running the update, or by using the [**NO\_MERGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-table-level) optimizer hint, as shown here:

UPDATE /\*+ NO\_MERGE(discounted) \*/ items,

(SELECT id FROM items

WHERE retail / wholesale >= 1.3 AND quantity < 100)

AS discounted

SET items.retail = items.retail \* 0.9

WHERE items.id = discounted.id;

The advantage of using the optimizer hint in such a case is that it applies only within the query block where it is used, so that it is not necessary to change the value of **optimizer\_switch** again after executing the **UPDATE**.

Another possibility is to rewrite the subquery so that it does not use **IN** or **EXISTS**, like this:

UPDATE items,

(SELECT id, retail / wholesale AS markup, quantity FROM items)

AS discounted

SET items.retail = items.retail \* 0.9

WHERE discounted.markup >= 1.3

AND discounted.quantity < 100

AND items.id = discounted.id;

In this case, the subquery is materialized by default rather than merged, so it is not necessary to disable merging of the derived table.

### 13.2.14 VALUES Statement

**[VALUES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "values" \o "13.2.14 VALUES Statement)** is a DML statement introduced in MySQL 8.0.19 which returns a set of one or more rows as a table. In other words, it is a table value constructor which also functions as a standalone SQL statement.

VALUES ***row\_constructor\_list*** [ORDER BY ***column\_designator***] [LIMIT BY ***number***]

***row\_constructor\_list***:

ROW(***value\_list***)[, ROW(***value\_list***)][, ...]

***value\_list***:

***value***[, ***value***][, ...]

***column\_designator***:

column\_***index***

The [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statement consists of the **VALUES** keyword followed by a list of one or more row constructors, separated by commas. A row constructor consists of the **ROW()** row constructor clause with a value list of one or more scalar values enclosed in the parentheses. A value can be a literal of any MySQL data type or an expression that resolves to a scalar value.

**ROW()** cannot be empty (but each of the supplied scalar values can be **NULL**). Each **ROW()** in the same [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statement must have the same number of values in its value list.

The **DEFAULT** keyword is not supported by **VALUES** and causes a syntax error, except when it is used to supply values in an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement.

The output of [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) is a table:

mysql> **VALUES ROW(1,-2,3), ROW(5,7,9), ROW(4,6,8);**

+----------+----------+----------+

| column\_0 | column\_1 | column\_2 |

+----------+----------+----------+

| 1 | -2 | 3 |

| 5 | 7 | 9 |

| 4 | 6 | 8 |

+----------+----------+----------+

3 rows in set (0.00 sec)

The columns of the table output from [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) have the implicitly named columns **column\_0**, **column\_1**, **column\_2**, and so on, always beginning with **0**. This fact can be used to order the rows by column using an optional **ORDER BY** clause in the same way that this clause works with a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, as shown here:

mysql> **VALUES ROW(1,-2,3), ROW(5,7,9), ROW(4,6,8) ORDER BY column\_1;**

+----------+----------+----------+

| column\_0 | column\_1 | column\_2 |

+----------+----------+----------+

| 1 | -2 | 3 |

| 4 | 6 | 8 |

| 5 | 7 | 9 |

+----------+----------+----------+

3 rows in set (0.00 sec)

The [**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) statement also supports a **LIMIT** clause for limiting the number of rows in the output.

The **VALUES** statement is permissive regarding data types of column values; you can mix types within the same column, as shown here:

mysql> **VALUES ROW("q", 42, '2019-12-18'),**

-> **ROW(23, "abc", 98.6),**

-> **ROW(27.0002, "Mary Smith", '{"a": 10, "b": 25}');**

+----------+------------+--------------------+

| column\_0 | column\_1 | column\_2 |

+----------+------------+--------------------+

| q | 42 | 2019-12-18 |

| 23 | abc | 98.6 |

| 27.0002 | Mary Smith | {"a": 10, "b": 25} |

+----------+------------+--------------------+

3 rows in set (0.00 sec)

**Important**

**VALUES** with one or more instances of **ROW()** acts as a table value constructor; although it can be used to supply values in an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statement, do not confuse it with the **VALUES** keyword that is also used for this purpose. You should also not confuse it with the [**VALUES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) function that refers to column values in [**INSERT ... ON DUPLICATE KEY UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert-on-duplicate).

You should also bear in mind that **ROW()** is a row value constructor (see [Section 13.2.11.5, “Row Subqueries”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#row-subqueries), whereas **VALUES ROW()** is a table value constructor; the two cannot be used interchangeably.

[**VALUES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#values) can be used in many cases where you could employ [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), including those listed here:

With [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), as shown here:

mysql> **SELECT 1,2 UNION SELECT 10,15;**

+----+----+

| 1 | 2 |

+----+----+

| 1 | 2 |

| 10 | 15 |

+----+----+

2 rows in set (0.00 sec)

mysql> **VALUES ROW(1,2) UNION VALUES ROW(10,15);**

+----------+----------+

| column\_0 | column\_1 |

+----------+----------+

| 1 | 2 |

| 10 | 15 |

+----------+----------+

2 rows in set (0.00 sec)

It is also possible in this fashion to union together constructed tables having more than one row, like this:

mysql> **VALUES ROW(1,2), ROW(3,4), ROW(5,6)**

> **UNION VALUES ROW(10,15),ROW(20,25);**

+----------+----------+

| column\_0 | column\_1 |

+----------+----------+

| 1 | 2 |

| 3 | 4 |

| 5 | 6 |

| 10 | 15 |

| 20 | 25 |

+----------+----------+

5 rows in set (0.00 sec)

You can also (and it is usually preferable to) omit [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) altogether in such cases and use a single **VALUES** statement, like this:

mysql> **VALUES ROW(1,2), ROW(3,4), ROW(5,6), ROW(10,15), ROW(20,25);**

+----------+----------+

| column\_0 | column\_1 |

+----------+----------+

| 1 | 2 |

| 3 | 4 |

| 5 | 6 |

| 10 | 15 |

| 20 | 25 |

+----------+----------+

**VALUES** can also be used in unions with [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statements, or both.

The constructed tables in the [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) must contain the same number of columns, just as if you were using [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). See [Section 13.2.10.3, “UNION Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), for further examples.

In joins. See [Section 13.2.10.2, “JOIN Clause”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#join), for more information and examples.

In place of [**VALUES()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_values) in an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) or [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace) statement, in which case its semantics differ slightly from what is described here. See [Section 13.2.6, “INSERT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), for details.

In place of the source table in [**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table-select) and [**CREATE VIEW ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view). See the descriptions of these statements for more information and examples.

### 13.2.15 WITH (Common Table Expressions)

A common table expression (CTE) is a named temporary result set that exists within the scope of a single statement and that can be referred to later within that statement, possibly multiple times. The following discussion describes how to write statements that use CTEs.

[Common Table Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions)

[Recursive Common Table Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursive)

[Limiting Common Table Expression Recursion](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursion-limits)

[Recursive Common Table Expression Examples](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursive-examples)

[Common Table Expressions Compared to Similar Constructs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-similar-constructs)

For information about CTE optimization, see [Section 8.2.2.4, “Optimizing Derived Tables, View References, and Common Table Expressions with Merging or Materialization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#derived-table-optimization).

#### Additional Resources

These articles contain additional information about using CTEs in MySQL, including many examples:

[MySQL 8.0 Labs: [Recursive] Common Table Expressions in MySQL (CTEs)](http://mysqlserverteam.com/mysql-8-0-labs-recursive-common-table-expressions-in-mysql-ctes/)

[MySQL 8.0 Labs: [Recursive] Common Table Expressions in MySQL (CTEs), Part Two – how to generate series](http://mysqlserverteam.com/mysql-8-0-labs-recursive-common-table-expressions-in-mysql-ctes-part-two-how-to-generate-series/)

[MySQL 8.0 Labs: [Recursive] Common Table Expressions in MySQL (CTEs), Part Three – hierarchies](http://mysqlserverteam.com/mysql-8-0-labs-recursive-common-table-expressions-in-mysql-ctes-part-three-hierarchies/)

[MySQL 8.0.1: [Recursive] Common Table Expressions in MySQL (CTEs), Part Four – depth-first or breadth-first traversal, transitive closure, cycle avoidance](http://mysqlserverteam.com/mysql-8-0-1-recursive-common-table-expressions-in-mysql-ctes-part-four-depth-first-or-breadth-first-traversal-transitive-closure-cycle-avoidance/)

#### Common Table Expressions

To specify common table expressions, use a [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause that has one or more comma-separated subclauses. Each subclause provides a subquery that produces a result set, and associates a name with the subquery. The following example defines CTEs named **cte1** and **cte2** in the [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause, and refers to them in the top-level [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) that follows the [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause:

WITH

cte1 AS (SELECT a, b FROM table1),

cte2 AS (SELECT c, d FROM table2)

SELECT b, d FROM cte1 JOIN cte2

WHERE cte1.a = cte2.c;

In the statement containing the [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause, each CTE name can be referenced to access the corresponding CTE result set.

A CTE name can be referenced in other CTEs, enabling CTEs to be defined based on other CTEs.

A CTE can refer to itself to define a recursive CTE. Common applications of recursive CTEs include series generation and traversal of hierarchical or tree-structured data.

Common table expressions are an optional part of the syntax for DML statements. They are defined using a [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause:

***with\_clause***:

WITH [RECURSIVE]

***cte\_name*** [(***col\_name*** [, ***col\_name***] ...)] AS (***subquery***)

[, ***cte\_name*** [(***col\_name*** [, ***col\_name***] ...)] AS (***subquery***)] ...

***cte\_name*** names a single common table expression and can be used as a table reference in the statement containing the [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause.

The ***subquery*** part of **AS (*subquery*)** is called the “subquery of the CTE” and is what produces the CTE result set. The parentheses following **AS** are required.

A common table expression is recursive if its subquery refers to its own name. The **RECURSIVE** keyword must be included if any CTE in the [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause is recursive. For more information, see [Recursive Common Table Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursive).

Determination of column names for a given CTE occurs as follows:

If a parenthesized list of names follows the CTE name, those names are the column names:

WITH cte (col1, col2) AS

(

SELECT 1, 2

UNION ALL

SELECT 3, 4

)

SELECT col1, col2 FROM cte;

The number of names in the list must be the same as the number of columns in the result set.

Otherwise, the column names come from the select list of the first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) within the **AS (*subquery*)** part:

WITH cte AS

(

SELECT 1 AS col1, 2 AS col2

UNION ALL

SELECT 3, 4

)

SELECT col1, col2 FROM cte;

A [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause is permitted in these contexts:

At the beginning of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements.

WITH ... SELECT ...

WITH ... UPDATE ...

WITH ... DELETE ...

At the beginning of subqueries (including derived table subqueries):

SELECT ... WHERE id IN (WITH ... SELECT ...) ...

SELECT \* FROM (WITH ... SELECT ...) AS dt ...

Immediately preceding [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) for statements that include a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement:

INSERT ... WITH ... SELECT ...

REPLACE ... WITH ... SELECT ...

CREATE TABLE ... WITH ... SELECT ...

CREATE VIEW ... WITH ... SELECT ...

DECLARE CURSOR ... WITH ... SELECT ...

EXPLAIN ... WITH ... SELECT ...

Only one [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause is permitted at the same level. [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) followed by [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) at the same level is not permitted, so this is illegal:

WITH cte1 AS (...) WITH cte2 AS (...) SELECT ...

To make the statement legal, use a single [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause that separates the subclauses by a comma:

WITH cte1 AS (...), cte2 AS (...) SELECT ...

However, a statement can contain multiple [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clauses if they occur at different levels:

WITH cte1 AS (SELECT 1)

SELECT \* FROM (WITH cte2 AS (SELECT 2) SELECT \* FROM cte2 JOIN cte1) AS dt;

A [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause can define one or more common table expressions, but each CTE name must be unique to the clause. This is illegal:

WITH cte1 AS (...), cte1 AS (...) SELECT ...

To make the statement legal, define the CTEs with unique names:

WITH cte1 AS (...), cte2 AS (...) SELECT ...

A CTE can refer to itself or to other CTEs:

A self-referencing CTE is recursive.

A CTE can refer to CTEs defined earlier in the same [**WITH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#with) clause, but not those defined later.

This constraint rules out mutually-recursive CTEs, where **cte1** references **cte2** and **cte2** references **cte1**. One of those references must be to a CTE defined later, which is not permitted.

A CTE in a given query block can refer to CTEs defined in query blocks at a more outer level, but not CTEs defined in query blocks at a more inner level.

For resolving references to objects with the same names, derived tables hide CTEs; and CTEs hide base tables, **TEMPORARY** tables, and views. Name resolution occurs by searching for objects in the same query block, then proceeding to outer blocks in turn while no object with the name is found.

Like derived tables, a CTE cannot contain outer references prior to MySQL 8.0.14. This is a MySQL restriction that is lifted in MySQL 8.0.14, not a restriction of the SQL standard. For additional syntax considerations specific to recursive CTEs, see [Recursive Common Table Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursive).

#### Recursive Common Table Expressions

A recursive common table expression is one having a subquery that refers to its own name. For example:

WITH RECURSIVE cte (n) AS

(

SELECT 1

UNION ALL

SELECT n + 1 FROM cte WHERE n < 5

)

SELECT \* FROM cte;

When executed, the statement produces this result, a single column containing a simple linear sequence:

+------+

| n |

+------+

| 1 |

| 2 |

| 3 |

| 4 |

| 5 |

+------+

A recursive CTE has this structure:

The **WITH** clause must begin with **WITH RECURSIVE** if any CTE in the **WITH** clause refers to itself. (If no CTE refers to itself, **RECURSIVE** is permitted but not required.)

If you forget **RECURSIVE** for a recursive CTE, this error is a likely result:

ERROR 1146 (42S02): Table '***cte\_name***' doesn't exist

The recursive CTE subquery has two parts, separated by [**UNION [ALL]**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) or [**UNION DISTINCT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union):

SELECT ... -- return initial row set

UNION ALL

SELECT ... -- return additional row sets

The first [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) produces the initial row or rows for the CTE and does not refer to the CTE name. The second [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) produces additional rows and recurses by referring to the CTE name in its **FROM** clause. Recursion ends when this part produces no new rows. Thus, a recursive CTE consists of a nonrecursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part followed by a recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part.

Each [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part can itself be a union of multiple [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements.

The types of the CTE result columns are inferred from the column types of the nonrecursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part only, and the columns are all nullable. For type determination, the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part is ignored.

If the nonrecursive and recursive parts are separated by [**UNION DISTINCT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union), duplicate rows are eliminated. This is useful for queries that perform transitive closures, to avoid infinite loops.

Each iteration of the recursive part operates only on the rows produced by the previous iteration. If the recursive part has multiple query blocks, iterations of each query block are scheduled in unspecified order, and each query block operates on rows that have been produced either by its previous iteration or by other query blocks since that previous iteration's end.

The recursive CTE subquery shown earlier has this nonrecursive part that retrieves a single row to produce the initial row set:

SELECT 1

The CTE subquery also has this recursive part:

SELECT n + 1 FROM cte WHERE n < 5

At each iteration, that [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) produces a row with a new value one greater than the value of **n** from the previous row set. The first iteration operates on the initial row set (**1**) and produces **1+1=2**; the second iteration operates on the first iteration's row set (**2**) and produces **2+1=3**; and so forth. This continues until recursion ends, which occurs when **n** is no longer less than 5.

If the recursive part of a CTE produces wider values for a column than the nonrecursive part, it may be necessary to widen the column in the nonrecursive part to avoid data truncation. Consider this statement:

WITH RECURSIVE cte AS

(

SELECT 1 AS n, 'abc' AS str

UNION ALL

SELECT n + 1, CONCAT(str, str) FROM cte WHERE n < 3

)

SELECT \* FROM cte;

In nonstrict SQL mode, the statement produces this output:

+------+------+

| n | str |

+------+------+

| 1 | abc |

| 2 | abc |

| 3 | abc |

+------+------+

The **str** column values are all **'abc'** because the nonrecursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) determines the column widths. Consequently, the wider **str** values produced by the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) are truncated.

In strict SQL mode, the statement produces an error:

ERROR 1406 (22001): Data too long for column 'str' at row 1

To address this issue, so that the statement does not produce truncation or errors, use [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) in the nonrecursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) to make the **str** column wider:

WITH RECURSIVE cte AS

(

SELECT 1 AS n, CAST('abc' AS CHAR(20)) AS str

UNION ALL

SELECT n + 1, CONCAT(str, str) FROM cte WHERE n < 3

)

SELECT \* FROM cte;

Now the statement produces this result, without truncation:

+------+--------------+

| n | str |

+------+--------------+

| 1 | abc |

| 2 | abcabc |

| 3 | abcabcabcabc |

+------+--------------+

Columns are accessed by name, not position, which means that columns in the recursive part can access columns in the nonrecursive part that have a different position, as this CTE illustrates:

WITH RECURSIVE cte AS

(

SELECT 1 AS n, 1 AS p, -1 AS q

UNION ALL

SELECT n + 1, q \* 2, p \* 2 FROM cte WHERE n < 5

)

SELECT \* FROM cte;

Because **p** in one row is derived from **q** in the previous row, and vice versa, the positive and negative values swap positions in each successive row of the output:

+------+------+------+

| n | p | q |

+------+------+------+

| 1 | 1 | -1 |

| 2 | -2 | 2 |

| 3 | 4 | -4 |

| 4 | -8 | 8 |

| 5 | 16 | -16 |

+------+------+------+

Some syntax constraints apply within recursive CTE subqueries:

The recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part must not contain these constructs:

Aggregate functions such as **SUM()**

Window functions

**GROUP BY**

**ORDER BY**

**DISTINCT**

Prior to MySQL 8.0.19, the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part of a recursive CTE also could not use a **LIMIT** clause. This restriction is lifted in MySQL 8.0.19, and **LIMIT** is now supported in such cases, along with an optional **OFFSET** clause. The effect on the result set is the same as when using **LIMIT** in the outermost **SELECT**, but is also more efficient, since using it with the recursive **SELECT** stops the generation of rows as soon as the requested number of them has been produced.

These constraints do not apply to the nonrecursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part of a recursive CTE. The prohibition on **DISTINCT** applies only to [**UNION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#union) members; **UNION DISTINCT** is permitted.

The recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part must reference the CTE only once and only in its **FROM** clause, not in any subquery. It can reference tables other than the CTE and join them with the CTE. If used in a join like this, the CTE must not be on the right side of a **LEFT JOIN**.

These constraints come from the SQL standard, other than the MySQL-specific exclusions of **ORDER BY**, **LIMIT** (MySQL 8.0.18 and earlier), and **DISTINCT**.

For recursive CTEs, [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) output rows for recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) parts display **Recursive** in the **Extra** column.

Cost estimates displayed by [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) represent cost per iteration, which might differ considerably from total cost. The optimizer cannot predict the number of iterations because it cannot predict at what point the **WHERE** clause becomes false.

CTE actual cost may also be affected by result set size. A CTE that produces many rows may require an internal temporary table large enough to be converted from in-memory to on-disk format and may suffer a performance penalty. If so, increasing the permitted in-memory temporary table size may improve performance; see [Section 8.4.4, “Internal Temporary Table Use in MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#internal-temporary-tables).

#### Limiting Common Table Expression Recursion

It is important for recursive CTEs that the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) part include a condition to terminate recursion. As a development technique to guard against a runaway recursive CTE, you can force termination by placing a limit on execution time:

The [**cte\_max\_recursion\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_cte_max_recursion_depth) system variable enforces a limit on the number of recursion levels for CTEs. The server terminates execution of any CTE that recurses more levels than the value of this variable.

The [**max\_execution\_time**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_execution_time) system variable enforces an execution timeout for [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements executed within the current session.

The [**MAX\_EXECUTION\_TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-execution-time) optimizer hint enforces a per-query execution timeout for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement in which it appears.

Suppose that a recursive CTE is mistakenly written with no recursion execution termination condition:

WITH RECURSIVE cte (n) AS

(

SELECT 1

UNION ALL

SELECT n + 1 FROM cte

)

SELECT \* FROM cte;

By default, [**cte\_max\_recursion\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_cte_max_recursion_depth) has a value of 1000, causing the CTE to terminate when it recurses past 1000 levels. Applications can change the session value to adjust for their requirements:

SET SESSION cte\_max\_recursion\_depth = 10; -- permit only shallow recursion

SET SESSION cte\_max\_recursion\_depth = 1000000; -- permit deeper recursion

You can also set the global [**cte\_max\_recursion\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_cte_max_recursion_depth) value to affect all sessions that begin subsequently.

For queries that execute and thus recurse slowly or in contexts for which there is reason to set the [**cte\_max\_recursion\_depth**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_cte_max_recursion_depth) value very high, another way to guard against deep recursion is to set a per-session timeout. To do so, execute a statement like this prior to executing the CTE statement:

SET max\_execution\_time = 1000; -- impose one second timeout

Alternatively, include an optimizer hint within the CTE statement itself:

WITH RECURSIVE cte (n) AS

(

SELECT 1

UNION ALL

SELECT n + 1 FROM cte

)

SELECT /\*+ SET\_VAR(cte\_max\_recursion\_depth = 1M) \*/ \* FROM cte;

WITH RECURSIVE cte (n) AS

(

SELECT 1

UNION ALL

SELECT n + 1 FROM cte

)

SELECT /\*+ MAX\_EXECUTION\_TIME(1000) \*/ \* FROM cte;

Beginning with MySQL 8.0.19, you can also use **LIMIT** within the recursive query to impose a maximum nuber of rows to be returned to the outermost [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), for example:

WITH RECURSIVE cte (n) AS

(

SELECT 1

UNION ALL

SELECT n + 1 FROM cte LIMIT 10000

)

SELECT \* FROM cte;

You can do this in addition to or instead of setting a time limit. Thus, the following CTE terminates after returning ten thousand rows or running for one thousand seconds, whichever occurs first:

WITH RECURSIVE cte (n) AS

(

SELECT 1

UNION ALL

SELECT n + 1 FROM cte LIMIT 10000

)

SELECT /\*+ MAX\_EXECUTION\_TIME(1000) \*/ \* FROM cte;

If a recursive query without an execution time limit enters an infinite loop, you can terminate it from another session using [**KILL QUERY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill). Within the session itself, the client program used to run the query might provide a way to kill the query. For example, in [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql), typing **Control+C** interrupts the current statement.

#### Recursive Common Table Expression Examples

As mentioned previously, recursive common table expressions (CTEs) are frequently used for series generation and traversing hierarchical or tree-structured data. This section shows some simple examples of these techniques.

[Fibonacci Series Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursive-fibonacci-series)

[Date Series Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursive-date-series)

[Hierarchical Data Traversal](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#common-table-expressions-recursive-hierarchy-traversal)

##### Fibonacci Series Generation

A Fibonacci series begins with the two numbers 0 and 1 (or 1 and 1) and each number after that is the sum of the previous two numbers. A recursive common table expression can generate a Fibonacci series if each row produced by the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) has access to the two previous numbers from the series. The following CTE generates a 10-number series using 0 and 1 as the first two numbers:

WITH RECURSIVE fibonacci (n, fib\_n, next\_fib\_n) AS

(

SELECT 1, 0, 1

UNION ALL

SELECT n + 1, next\_fib\_n, fib\_n + next\_fib\_n

FROM fibonacci WHERE n < 10

)

SELECT \* FROM fibonacci;

The CTE produces this result:

+------+-------+------------+

| n | fib\_n | next\_fib\_n |

+------+-------+------------+

| 1 | 0 | 1 |

| 2 | 1 | 1 |

| 3 | 1 | 2 |

| 4 | 2 | 3 |

| 5 | 3 | 5 |

| 6 | 5 | 8 |

| 7 | 8 | 13 |

| 8 | 13 | 21 |

| 9 | 21 | 34 |

| 10 | 34 | 55 |

+------+-------+------------+

How the CTE works:

**n** is a display column to indicate that the row contains the **n**-th Fibonacci number. For example, the 8th Fibonacci number is 13.

The **fib\_n** column displays Fibonacci number **n**.

The **next\_fib\_n** column displays the next Fibonacci number after number **n**. This column provides the next series value to the next row, so that row can produce the sum of the two previous series values in its **fib\_n** column.

Recursion ends when **n** reaches 10. This is an arbitrary choice, to limit the output to a small set of rows.

The preceding output shows the entire CTE result. To select just part of it, add an appropriate **WHERE** clause to the top-level [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). For example, to select the 8th Fibonacci number, do this:

mysql> **WITH RECURSIVE fibonacci ...**

...

**SELECT fib\_n FROM fibonacci WHERE n = 8;**

+-------+

| fib\_n |

+-------+

| 13 |

+-------+

##### Date Series Generation

A common table expression can generate a series of successive dates, which is useful for generating summaries that include a row for all dates in the series, including dates not represented in the summarized data.

Suppose that a table of sales numbers contains these rows:

mysql> **SELECT \* FROM sales ORDER BY date, price;**

+------------+--------+

| date | price |

+------------+--------+

| 2017-01-03 | 100.00 |

| 2017-01-03 | 200.00 |

| 2017-01-06 | 50.00 |

| 2017-01-08 | 10.00 |

| 2017-01-08 | 20.00 |

| 2017-01-08 | 150.00 |

| 2017-01-10 | 5.00 |

+------------+--------+

This query summarizes the sales per day:

mysql> **SELECT date, SUM(price) AS sum\_price**

**FROM sales**

**GROUP BY date**

**ORDER BY date;**

+------------+-----------+

| date | sum\_price |

+------------+-----------+

| 2017-01-03 | 300.00 |

| 2017-01-06 | 50.00 |

| 2017-01-08 | 180.00 |

| 2017-01-10 | 5.00 |

+------------+-----------+

However, that result contains “holes” for dates not represented in the range of dates spanned by the table. A result that represents all dates in the range can be produced using a recursive CTE to generate that set of dates, joined with a **LEFT JOIN** to the sales data.

Here is the CTE to generate the date range series:

WITH RECURSIVE dates (date) AS

(

SELECT MIN(date) FROM sales

UNION ALL

SELECT date + INTERVAL 1 DAY FROM dates

WHERE date + INTERVAL 1 DAY <= (SELECT MAX(date) FROM sales)

)

SELECT \* FROM dates;

The CTE produces this result:

+------------+

| date |

+------------+

| 2017-01-03 |

| 2017-01-04 |

| 2017-01-05 |

| 2017-01-06 |

| 2017-01-07 |

| 2017-01-08 |

| 2017-01-09 |

| 2017-01-10 |

+------------+

How the CTE works:

The nonrecursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) produces the lowest date in the date range spanned by the **sales** table.

Each row produced by the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) adds one day to the date produced by the previous row.

Recursion ends after the dates reach the highest date in the date range spanned by the **sales** table.

Joining the CTE with a **LEFT JOIN** against the **sales** table produces the sales summary with a row for each date in the range:

WITH RECURSIVE dates (date) AS

(

SELECT MIN(date) FROM sales

UNION ALL

SELECT date + INTERVAL 1 DAY FROM dates

WHERE date + INTERVAL 1 DAY <= (SELECT MAX(date) FROM sales)

)

SELECT dates.date, COALESCE(SUM(price), 0) AS sum\_price

FROM dates LEFT JOIN sales ON dates.date = sales.date

GROUP BY dates.date

ORDER BY dates.date;

The output looks like this:

+------------+-----------+

| date | sum\_price |

+------------+-----------+

| 2017-01-03 | 300.00 |

| 2017-01-04 | 0.00 |

| 2017-01-05 | 0.00 |

| 2017-01-06 | 50.00 |

| 2017-01-07 | 0.00 |

| 2017-01-08 | 180.00 |

| 2017-01-09 | 0.00 |

| 2017-01-10 | 5.00 |

+------------+-----------+

Some points to note:

Are the queries inefficient, particularly the one with the [**MAX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_max) subquery executed for each row in the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select)? [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) shows that the subquery containing [**MAX()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_max) is evaluated only once and the result is cached.

The use of [**COALESCE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coalesce) avoids displaying **NULL** in the **sum\_price** column on days for which no sales data occur in the **sales** table.

##### Hierarchical Data Traversal

Recursive common table expressions are useful for traversing data that forms a hierarchy. Consider these statements that create a small data set that shows, for each employee in a company, the employee name and ID number, and the ID of the employee's manager. The top-level employee (the CEO), has a manager ID of **NULL** (no manager).

CREATE TABLE employees (

id INT PRIMARY KEY NOT NULL,

name VARCHAR(100) NOT NULL,

manager\_id INT NULL,

INDEX (manager\_id),

FOREIGN KEY (manager\_id) REFERENCES employees (id)

);

INSERT INTO employees VALUES

(333, "Yasmina", NULL), # Yasmina is the CEO (manager\_id is NULL)

(198, "John", 333), # John has ID 198 and reports to 333 (Yasmina)

(692, "Tarek", 333),

(29, "Pedro", 198),

(4610, "Sarah", 29),

(72, "Pierre", 29),

(123, "Adil", 692);

The resulting data set looks like this:

mysql> **SELECT \* FROM employees ORDER BY id;**

+------+---------+------------+

| id | name | manager\_id |

+------+---------+------------+

| 29 | Pedro | 198 |

| 72 | Pierre | 29 |

| 123 | Adil | 692 |

| 198 | John | 333 |

| 333 | Yasmina | NULL |

| 692 | Tarek | 333 |

| 4610 | Sarah | 29 |

+------+---------+------------+

To produce the organizational chart with the management chain for each employee (that is, the path from CEO to employee), use a recursive CTE:

WITH RECURSIVE employee\_paths (id, name, path) AS

(

SELECT id, name, CAST(id AS CHAR(200))

FROM employees

WHERE manager\_id IS NULL

UNION ALL

SELECT e.id, e.name, CONCAT(ep.path, ',', e.id)

FROM employee\_paths AS ep JOIN employees AS e

ON ep.id = e.manager\_id

)

SELECT \* FROM employee\_paths ORDER BY path;

The CTE produces this output:

+------+---------+-----------------+

| id | name | path |

+------+---------+-----------------+

| 333 | Yasmina | 333 |

| 198 | John | 333,198 |

| 29 | Pedro | 333,198,29 |

| 4610 | Sarah | 333,198,29,4610 |

| 72 | Pierre | 333,198,29,72 |

| 692 | Tarek | 333,692 |

| 123 | Adil | 333,692,123 |

+------+---------+-----------------+

How the CTE works:

The nonrecursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) produces the row for the CEO (the row with a **NULL** manager ID).

The **path** column is widened to **CHAR(200)** to ensure that there is room for the longer **path** values produced by the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

Each row produced by the recursive [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) finds all employees who report directly to an employee produced by a previous row. For each such employee, the row includes the employee ID and name, and the employee management chain. The chain is the manager's chain, with the employee ID added to the end.

Recursion ends when employees have no others who report to them.

To find the path for a specific employee or employees, add a **WHERE** clause to the top-level [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). For example, to display the results for Tarek and Sarah, modify that [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) like this:

mysql> **WITH RECURSIVE ...**

...

**SELECT \* FROM employees\_extended**

**WHERE id IN (692, 4610)**

**ORDER BY path;**

+------+-------+-----------------+

| id | name | path |

+------+-------+-----------------+

| 4610 | Sarah | 333,198,29,4610 |

| 692 | Tarek | 333,692 |

+------+-------+-----------------+

#### Common Table Expressions Compared to Similar Constructs

Common table expressions (CTEs) are similar to derived tables in some ways:

Both constructs are named.

Both constructs exist for the scope of a single statement.

Because of these similarities, CTEs and derived tables often can be used interchangeably. As a trivial example, these statements are equivalent:

WITH cte AS (SELECT 1) SELECT \* FROM cte;

SELECT \* FROM (SELECT 1) AS dt;

However, CTEs have some advantages over derived tables:

A derived table can be referenced only a single time within a query. A CTE can be referenced multiple times. To use multiple instances of a derived table result, you must derive the result multiple times.

A CTE can be self-referencing (recursive).

One CTE can refer to another.

A CTE may be easier to read when its definition appears at the beginning of the statement rather than embedded within it.

CTEs are similar to tables created with [**CREATE [TEMPORARY] TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) but need not be defined or dropped explicitly. For a CTE, you need no privileges to create tables.

## 13.3 Transactional and Locking Statements

[13.3.1 START TRANSACTION, COMMIT, and ROLLBACK Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit)

[13.3.2 Statements That Cannot Be Rolled Back](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cannot-roll-back)

[13.3.3 Statements That Cause an Implicit Commit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit)

[13.3.4 SAVEPOINT, ROLLBACK TO SAVEPOINT, and RELEASE SAVEPOINT Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#savepoint)

[13.3.5 LOCK INSTANCE FOR BACKUP and UNLOCK INSTANCE Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-instance-for-backup)

[13.3.6 LOCK TABLES and UNLOCK TABLES Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables)

[13.3.7 SET TRANSACTION Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction)

[13.3.8 XA Transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa)

MySQL supports local transactions (within a given client session) through statements such as [**SET autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), and [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit). See [Section 13.3.1, “START TRANSACTION, COMMIT, and ROLLBACK Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit). XA transaction support enables MySQL to participate in distributed transactions as well. See [Section 13.3.8, “XA Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa).

### 13.3.1 START TRANSACTION, COMMIT, and ROLLBACK Statements

START TRANSACTION

[***transaction\_characteristic*** [, ***transaction\_characteristic***] ...]

***transaction\_characteristic***: {

WITH CONSISTENT SNAPSHOT

| READ WRITE

| READ ONLY

}

BEGIN [WORK]

COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]

ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]

SET autocommit = {0 | 1}

These statements provide control over use of [transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction):

**START TRANSACTION** or **BEGIN** start a new transaction.

**COMMIT** commits the current transaction, making its changes permanent.

**ROLLBACK** rolls back the current transaction, canceling its changes.

**SET autocommit** disables or enables the default autocommit mode for the current session.

By default, MySQL runs with [autocommit](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_autocommit) mode enabled. This means that, when not otherwise inside a transaction, each statement is atomic, as if it were surrounded by **START TRANSACTION** and **COMMIT**. You cannot use **ROLLBACK** to undo the effect; however, if an error occurs during statement execution, the statement is rolled back.

To disable autocommit mode implicitly for a single series of statements, use the **START TRANSACTION** statement:

START TRANSACTION;

SELECT @A:=SUM(salary) FROM table1 WHERE type=1;

UPDATE table2 SET summary=@A WHERE type=1;

COMMIT;

With **START TRANSACTION**, autocommit remains disabled until you end the transaction with **COMMIT** or **ROLLBACK**. The autocommit mode then reverts to its previous state.

**START TRANSACTION** permits several modifiers that control transaction characteristics. To specify multiple modifiers, separate them by commas.

The **WITH CONSISTENT SNAPSHOT** modifier starts a [consistent read](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_consistent_read) for storage engines that are capable of it. This applies only to **InnoDB**. The effect is the same as issuing a **START TRANSACTION** followed by a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) from any **InnoDB** table. See [Section 15.7.2.3, “Consistent Nonlocking Reads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-consistent-read). The **WITH CONSISTENT SNAPSHOT** modifier does not change the current transaction [isolation level](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_isolation_level), so it provides a consistent snapshot only if the current isolation level is one that permits a consistent read. The only isolation level that permits a consistent read is [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read). For all other isolation levels, the **WITH CONSISTENT SNAPSHOT** clause is ignored. A warning is generated when the **WITH CONSISTENT SNAPSHOT** clause is ignored.

The **READ WRITE** and **READ ONLY** modifiers set the transaction access mode. They permit or prohibit changes to tables used in the transaction. The **READ ONLY** restriction prevents the transaction from modifying or locking both transactional and nontransactional tables that are visible to other transactions; the transaction can still modify or lock temporary tables.

MySQL enables extra optimizations for queries on **InnoDB** tables when the transaction is known to be read-only. Specifying **READ ONLY** ensures these optimizations are applied in cases where the read-only status cannot be determined automatically. See [Section 8.5.3, “Optimizing InnoDB Read-Only Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#innodb-performance-ro-txn) for more information.

If no access mode is specified, the default mode applies. Unless the default has been changed, it is read/write. It is not permitted to specify both **READ WRITE** and **READ ONLY** in the same statement.

In read-only mode, it remains possible to change tables created with the **TEMPORARY** keyword using DML statements. Changes made with DDL statements are not permitted, just as with permanent tables.

For additional information about transaction access mode, including ways to change the default mode, see [Section 13.3.7, “SET TRANSACTION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction).

If the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, explicitly starting a transaction with **START TRANSACTION READ WRITE** requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

**Important**

Many APIs used for writing MySQL client applications (such as JDBC) provide their own methods for starting transactions that can (and sometimes should) be used instead of sending a **START TRANSACTION** statement from the client. See [Chapter 29, *Connectors and APIs*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\connectors-apis.html), or the documentation for your API, for more information.

To disable autocommit mode explicitly, use the following statement:

SET autocommit=0;

After disabling autocommit mode by setting the [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) variable to zero, changes to transaction-safe tables (such as those for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) or [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html)) are not made permanent immediately. You must use [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) to store your changes to disk or **ROLLBACK** to ignore the changes.

[**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) is a session variable and must be set for each session. To disable autocommit mode for each new connection, see the description of the [**autocommit**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit) system variable at [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

**BEGIN** and **BEGIN WORK** are supported as aliases of **START TRANSACTION** for initiating a transaction. **START TRANSACTION** is standard SQL syntax, is the recommended way to start an ad-hoc transaction, and permits modifiers that **BEGIN** does not.

The **BEGIN** statement differs from the use of the **BEGIN** keyword that starts a [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) compound statement. The latter does not begin a transaction. See [Section 13.6.1, “BEGIN ... END Compound Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end).

**Note**

Within all stored programs (stored procedures and functions, triggers, and events), the parser treats **BEGIN [WORK]** as the beginning of a [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block. Begin a transaction in this context with [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) instead.

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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_completion_type) system variable determines the default completion behavior. See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

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 new transaction also uses the same access mode (**READ WRITE** or **READ ONLY**) 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_completion_type) system variable is set to cause chaining or release completion by default.

Beginning a transaction causes any pending transaction to be committed. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit), for more information.

Beginning a transaction also causes table locks acquired with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to be released, as though you had executed [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables). Beginning a transaction does not release a global read lock acquired with [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock).

For best results, transactions should be performed using only tables managed by a single transaction-safe storage engine. Otherwise, the following problems can occur:

If you use tables from more than one transaction-safe storage engine (such as **InnoDB**), and the transaction isolation level is not [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable), it is possible that when one transaction commits, another ongoing transaction that uses the same tables sees only some of the changes made by the first transaction. That is, the atomicity of transactions is not guaranteed with mixed engines and inconsistencies can result. (If mixed-engine transactions are infrequent, you can use [**SET TRANSACTION ISOLATION LEVEL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) to set the isolation level to [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable) on a per-transaction basis as necessary.)

If you use tables that are not transaction-safe within a transaction, changes to those tables are stored at once, regardless of the status of autocommit mode.

If you issue a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement after updating a nontransactional table within a transaction, an [**ER\_WARNING\_NOT\_COMPLETE\_ROLLBACK**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_warning_not_complete_rollback) warning occurs. Changes to transaction-safe tables are rolled back, but not changes to nontransaction-safe tables.

Each transaction is stored in the binary log in one chunk, upon [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit). Transactions that are rolled back are not logged. (***Exception***: Modifications to nontransactional tables cannot be rolled back. If a transaction that is rolled back includes modifications to nontransactional tables, the entire transaction is logged with a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement at the end to ensure that modifications to the nontransactional tables are replicated.) See [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log).

You can change the isolation level or access mode for transactions with the [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) statement. See [Section 13.3.7, “SET TRANSACTION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction).

Rolling back can be a slow operation that may occur implicitly without the user having explicitly asked for it (for example, when an error occurs). Because of this, [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) displays **Rolling back** in the **State** column for the session, not only for explicit rollbacks performed with the [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement but also for implicit rollbacks.

**Note**

In MySQL 8.0, **BEGIN**, **COMMIT**, and **ROLLBACK** are not affected by [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) or [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) rules.

When **InnoDB** performs a complete rollback of a transaction, all locks set by the transaction are released. If a single SQL statement within a transaction rolls back as a result of an error, such as a duplicate key error, locks set by the statement are preserved while the transaction remains active. This happens because **InnoDB** stores row locks in a format such that it cannot know afterward which lock was set by which statement.

If a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement within a transaction calls a stored function, and a statement within the stored function fails, that statement rolls back. If [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) is executed for the transaction subsequently, the entire transaction rolls back.

### 13.3.2 Statements That Cannot Be Rolled Back

Some statements cannot be rolled back. In general, these include data definition language (DDL) statements, such as those that create or drop databases, those that create, drop, or alter tables or stored routines.

You should design your transactions not to include such statements. If you issue a statement early in a transaction that cannot be rolled back, and then another statement later fails, the full effect of the transaction cannot be rolled back in such cases by issuing a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement.

### 13.3.3 Statements That Cause an Implicit Commit

The statements listed in this section (and any synonyms for them) implicitly end any transaction active in the current session, as if you had done a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) before executing the statement.

Most of these statements also cause an implicit commit after executing. The intent is to handle each such statement in its own special transaction. Transaction-control and locking statements are exceptions: If an implicit commit occurs before execution, another does not occur after.

***Data definition language (DDL) statements that define or modify database objects.*** [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event), [**ALTER FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-function), [**ALTER PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-procedure), [**ALTER SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-server), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), [**ALTER VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-view), [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database), [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event), [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function), [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index), [**CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure), [**CREATE ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-role), [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server), [**CREATE SPATIAL REFERENCE SYSTEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-spatial-reference-system), [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger), [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view), [**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database), [**DROP EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-event), [**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function), [**DROP INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-index), [**DROP PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-procedure), [**DROP ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-role), [**DROP SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-server), [**DROP SPATIAL REFERENCE SYSTEM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-spatial-reference-system), [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table), [**DROP TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-trigger), [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view), [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin), [**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table), [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table), [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin).

[**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statements do not commit a transaction if the **TEMPORARY** keyword is used. (This does not apply to other operations on temporary tables such as [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) and [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index), which do cause a commit.) However, although no implicit commit occurs, neither can the statement be rolled back, which means that the use of such statements causes transactional atomicity to be violated. For example, if you use [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and then roll back the transaction, the table remains in existence.

The [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement in **InnoDB** is processed as a single transaction. This means that a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) from the user does not undo [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statements the user made during that transaction.

[**CREATE TABLE ... SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) causes an implicit commit before and after the statement is executed when you are creating nontemporary tables. (No commit occurs for **CREATE TEMPORARY TABLE ... SELECT**.)

***Statements that implicitly use or modify tables in the mysql database.*** [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user), [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user), [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user), [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant), [**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user), [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke), [**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password).

***Transaction-control and locking statements.*** [**BEGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), **SET autocommit = 1** (if the value is not already 1), [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables).

[**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) commits a transaction only if any tables currently have been locked with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to acquire nontransactional table locks. A commit does not occur for [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) following [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock) because the latter statement does not acquire table-level locks.

Transactions cannot be nested. This is a consequence of the implicit commit performed for any current transaction when you issue a [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement or one of its synonyms.

Statements that cause an implicit commit cannot be used in an XA transaction while the transaction is in an **ACTIVE** state.

The [**BEGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement differs from the use of the **BEGIN** keyword that starts a [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) compound statement. The latter does not cause an implicit commit. See [Section 13.6.1, “BEGIN ... END Compound Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end).

***Data loading statements.*** [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data). [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) causes an implicit commit only for tables using the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine.

***Administrative statements.*** [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table), [**CACHE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cache-index), [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table), [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush), [**LOAD INDEX INTO CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-index), [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table), [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), [**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset) (but not [**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist)).

***Replication control statements***. [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica), [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica), [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica), [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to), [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to).

### 13.3.4 SAVEPOINT, ROLLBACK TO SAVEPOINT, and RELEASE SAVEPOINT Statements

SAVEPOINT ***identifier***

ROLLBACK [WORK] TO [SAVEPOINT] ***identifier***

RELEASE SAVEPOINT ***identifier***

**InnoDB** supports the SQL statements [**SAVEPOINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#savepoint), [**ROLLBACK TO SAVEPOINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#savepoint), [**RELEASE SAVEPOINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#savepoint) and the optional **WORK** keyword for [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit).

The [**SAVEPOINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#savepoint) statement sets a named transaction savepoint with a name of ***identifier***. If the current transaction has a savepoint with the same name, the old savepoint is deleted and a new one is set.

The [**ROLLBACK TO SAVEPOINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#savepoint) statement rolls back a transaction to the named savepoint without terminating the transaction. Modifications that the current transaction made to rows after the savepoint was set are undone in the rollback, but **InnoDB** does not release the row locks that were stored in memory after the savepoint. (For a new inserted row, the lock information is carried by the transaction ID stored in the row; the lock is not separately stored in memory. In this case, the row lock is released in the undo.) Savepoints that were set at a later time than the named savepoint are deleted.

If the [**ROLLBACK TO SAVEPOINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#savepoint) statement returns the following error, it means that no savepoint with the specified name exists:

ERROR 1305 (42000): SAVEPOINT ***identifier*** does not exist

The [**RELEASE SAVEPOINT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#savepoint) statement removes the named savepoint from the set of savepoints of the current transaction. No commit or rollback occurs. It is an error if the savepoint does not exist.

All savepoints of the current transaction are deleted if you execute a [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), or a [**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) that does not name a savepoint.

A new savepoint level is created when a stored function is invoked or a trigger is activated. The savepoints on previous levels become unavailable and thus do not conflict with savepoints on the new level. When the function or trigger terminates, any savepoints it created are released and the previous savepoint level is restored.

### 13.3.5 LOCK INSTANCE FOR BACKUP and UNLOCK INSTANCE Statements

LOCK INSTANCE FOR BACKUP

UNLOCK INSTANCE

**LOCK INSTANCE FOR BACKUP** acquires an instance-level backup lock that permits DML during an online backup while preventing operations that could result in an inconsistent snapshot.

Executing the **LOCK INSTANCE FOR BACKUP** statement requires the [**BACKUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_backup-admin) privilege. The [**BACKUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_backup-admin) privilege is automatically granted to users with the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege when performing an in-place upgrade to MySQL 8.0 from an earlier version.

Multiple sessions can hold a backup lock simultaneously.

**UNLOCK INSTANCE** releases a backup lock held by the current session. A backup lock held by a session is also released if the session is terminated.

**LOCK INSTANCE FOR BACKUP** prevents files from being created, renamed, or removed. [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table), [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table), and account management statements are blocked. See [Section 13.7.1, “Account Management Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#account-management-statements). Operations that modify **InnoDB** files that are not recorded in the **InnoDB** redo log are also blocked.

**LOCK INSTANCE FOR BACKUP** permits DDL operations that only affect user-created temporary tables. In effect, files that belong to user-created temporary tables can be created, renamed, or removed while a backup lock is held. Creation of binary log files is also permitted.

A backup lock acquired by **LOCK INSTANCE FOR BACKUP** is independent of transactional locks and locks taken by [**FLUSH TABLES *tbl\_name* [, *tbl\_name*] ... WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock-with-list), and the following sequences of statements are permitted:

LOCK INSTANCE FOR BACKUP;

FLUSH TABLES ***tbl\_name*** [, ***tbl\_name***] ... WITH READ LOCK;

UNLOCK TABLES;

UNLOCK INSTANCE;

FLUSH TABLES ***tbl\_name*** [, ***tbl\_name***] ... WITH READ LOCK;

LOCK INSTANCE FOR BACKUP;

UNLOCK INSTANCE;

UNLOCK TABLES;

The [**lock\_wait\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lock_wait_timeout) setting defines the amount of time that a **LOCK INSTANCE FOR BACKUP** statement waits to acquire a lock before giving up.

### 13.3.6 LOCK TABLES and UNLOCK TABLES Statements

LOCK TABLES

***tbl\_name*** [[AS] ***alias***] ***lock\_type***

[, ***tbl\_name*** [[AS] ***alias***] ***lock\_type***] ...

***lock\_type***: {

READ [LOCAL]

| [LOW\_PRIORITY] WRITE

}

UNLOCK TABLES

MySQL 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. This is explained in more detail in [Table-Locking Restrictions and Conditions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables-restrictions).

[**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) explicitly acquires table locks for the current client session. Table locks can be acquired for base tables or views. You must have the [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_lock-tables) privilege, and the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for each object to be locked.

For view locking, [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) adds all base tables used in the view to the set of tables to be locked and locks them automatically. For tables underlying any view being locked, [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) checks that the view definer (for **SQL SECURITY DEFINER** views) or invoker (for all views) has the proper privileges on the tables.

If you lock a table explicitly with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), any tables used in triggers are also locked implicitly, as described in [LOCK TABLES and Triggers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables-and-triggers).

If you lock a table explicitly with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), any tables related by a foreign key constraint are opened and locked implicitly. For foreign key checks, a shared read-only lock ([**LOCK TABLES READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables)) is taken on related tables. For cascading updates, a shared-nothing write lock ([**LOCK TABLES WRITE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables)) is taken on related tables that are involved in the operation.

[**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) explicitly releases any table locks held by the current session. [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) implicitly releases any table locks held by the current session before acquiring new locks.

Another use for [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) is to release the global read lock acquired with the [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock) statement, which enables you to lock all tables in all databases. See [Section 13.7.8.3, “FLUSH Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush). (This is a very convenient way to get backups if you have a file system such as Veritas that can take snapshots in time.)

A table lock protects only against inappropriate reads or writes by other sessions. A session holding a **WRITE** lock can perform table-level operations such as [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) or [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table). For sessions holding a **READ** lock, [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) and [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) operations are not permitted.

The following discussion applies only to non-**TEMPORARY** tables. [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) is permitted (but ignored) for a **TEMPORARY** table. The table can be accessed freely by the session within which it was created, regardless of what other locking may be in effect. No lock is necessary because no other session can see the table.

[Table Lock Acquisition](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table-lock-acquisition)

[Table Lock Release](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table-lock-release)

[Interaction of Table Locking and Transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables-and-transactions)

[LOCK TABLES and Triggers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables-and-triggers)

[Table-Locking Restrictions and Conditions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables-restrictions)

#### Table Lock Acquisition

To acquire table locks within the current session, use the [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement, which acquires metadata locks (see [Section 8.11.4, “Metadata Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#metadata-locking)).

The following lock types are available:

**READ [LOCAL]** lock:

The session that holds the lock can read the table (but not write it).

Multiple sessions can acquire a **READ** lock for the table at the same time.

Other sessions can read the table without explicitly acquiring a **READ** lock.

The **LOCAL** modifier enables nonconflicting [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements (concurrent inserts) by other sessions to execute while the lock is held. (See [Section 8.11.3, “Concurrent Inserts”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#concurrent-inserts).) However, **READ LOCAL** cannot be used if you are going to manipulate the database using processes external to the server while you hold the lock. For **InnoDB** tables, **READ LOCAL** is the same as **READ**.

**[LOW\_PRIORITY] WRITE** lock:

The session that holds the lock can read and write the table.

Only the session that holds the lock can access the table. No other session can access it until the lock is released.

Lock requests for the table by other sessions block while the **WRITE** lock is held.

The **LOW\_PRIORITY** modifier has no effect. In previous versions of MySQL, it affected locking behavior, but this is no longer true. It is now deprecated and its use produces a warning. Use **WRITE** without **LOW\_PRIORITY** instead.

**WRITE** locks normally have higher priority than **READ** locks to ensure that updates are processed as soon as possible. This means that if one session obtains a **READ** lock and then another session requests a **WRITE** lock, subsequent **READ** lock requests wait until the session that requested the **WRITE** lock has obtained the lock and released it. (An exception to this policy can occur for small values of the [**max\_write\_lock\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_write_lock_count) system variable; see [Section 8.11.4, “Metadata Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#metadata-locking).)

If the [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement must wait due to locks held by other sessions on any of the tables, it blocks until all locks can be acquired.

A session that requires locks must acquire all the locks that it needs in a single [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement. While the locks thus obtained are held, the session can access only the locked tables. For example, in the following sequence of statements, an error occurs for the attempt to access **t2** because it was not locked in the [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement:

mysql> **LOCK TABLES t1 READ;**

mysql> **SELECT COUNT(\*) FROM t1;**

+----------+

| COUNT(\*) |

+----------+

| 3 |

+----------+

mysql> **SELECT COUNT(\*) FROM t2;**

ERROR 1100 (HY000): Table 't2' was not locked with LOCK TABLES

Tables in the **INFORMATION\_SCHEMA** database are an exception. They can be accessed without being locked explicitly even while a session holds table locks obtained with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables).

You cannot refer to a locked table multiple times in a single query using the same name. Use aliases instead, and obtain a separate lock for the table and each alias:

mysql> **LOCK TABLE t WRITE, t AS t1 READ;**

mysql> **INSERT INTO t SELECT \* FROM t;**

ERROR 1100: Table 't' was not locked with LOCK TABLES

mysql> **INSERT INTO t SELECT \* FROM t AS t1;**

The error occurs for the first [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) because there are two references to the same name for a locked table. The second [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) succeeds because the references to the table use different names.

If your statements refer to a table by means of an alias, you must lock the table using that same alias. It does not work to lock the table without specifying the alias:

mysql> **LOCK TABLE t READ;**

mysql> **SELECT \* FROM t AS myalias;**

ERROR 1100: Table 'myalias' was not locked with LOCK TABLES

Conversely, if you lock a table using an alias, you must refer to it in your statements using that alias:

mysql> **LOCK TABLE t AS myalias READ;**

mysql> **SELECT \* FROM t;**

ERROR 1100: Table 't' was not locked with LOCK TABLES

mysql> **SELECT \* FROM t AS myalias;**

#### Table Lock Release

When the table locks held by a session are released, they are all released at the same time. A session can release its locks explicitly, or locks may be released implicitly under certain conditions.

A session can release its locks explicitly with [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables).

If a session issues a [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement to acquire a lock while already holding locks, its existing locks are released implicitly before the new locks are granted.

If a session begins a transaction (for example, with [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit)), an implicit [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) is performed, which causes existing locks to be released. (For additional information about the interaction between table locking and transactions, see [Interaction of Table Locking and Transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables-and-transactions).)

If the connection for a client session terminates, whether normally or abnormally, the server implicitly releases all table locks held by the session (transactional and nontransactional). If the client reconnects, the locks are no longer in effect. In addition, if the client had an active transaction, the server rolls back the transaction upon disconnect, and if reconnect occurs, the new session begins with autocommit enabled. For this reason, clients may wish to disable auto-reconnect. With auto-reconnect in effect, the client is not notified if reconnect occurs but any table locks or current transaction are lost. With auto-reconnect disabled, if the connection drops, an error occurs for the next statement issued. The client can detect the error and take appropriate action such as reacquiring the locks or redoing the transaction. See [Automatic Reconnection Control](https://dev.mysql.com/doc/c-api/8.0/en/c-api-auto-reconnect.html).

**Note**

If you use [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on a locked table, it may become unlocked. For example, if you attempt a second [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operation, the result may be an error **Table '*tbl\_name*' was not locked with LOCK TABLES**. To handle this, lock the table again prior to the second alteration. See also [Section B.3.6.1, “Problems with ALTER TABLE”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#alter-table-problems).

#### Interaction of Table Locking and Transactions

**[LOCK TABLES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "lock-tables" \o "13.3.6 LOCK TABLES and UNLOCK TABLES Statements)** and [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) interact with the use of transactions as follows:

[**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) is not transaction-safe and implicitly commits any active transaction before attempting to lock the tables.

[**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) implicitly commits any active transaction, but only if [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) has been used to acquire table locks. For example, in the following set of statements, [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) releases the global read lock but does not commit the transaction because no table locks are in effect:

FLUSH TABLES WITH READ LOCK;

START TRANSACTION;

SELECT ... ;

UNLOCK TABLES;

Beginning a transaction (for example, with [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit)) implicitly commits any current transaction and releases existing table locks.

[**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock) acquires a global read lock and not table locks, so it is not subject to the same behavior as [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) and [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) with respect to table locking and implicit commits. For example, [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) does not release the global read lock. See [Section 13.7.8.3, “FLUSH Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush).

Other statements that implicitly cause transactions to be committed do not release existing table locks. For a list of such statements, see [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

The correct way to use [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) and [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) with transactional tables, such as **InnoDB** tables, is to begin a transaction with **SET autocommit = 0** (not [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit)) followed by [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), and to not call [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) until you commit the transaction explicitly. For example, if you need to write to table **t1** and read from table **t2**, you can do this:

SET autocommit=0;

LOCK TABLES t1 WRITE, t2 READ, ...;

***... do something with tables t1 and t2 here ...***

COMMIT;

UNLOCK TABLES;

When you call [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), **InnoDB** internally takes its own table lock, and MySQL takes its own table lock. **InnoDB** releases its internal table lock at the next commit, but for MySQL to release its table lock, you have to call [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables). You should not have [**autocommit = 1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit), because then **InnoDB** releases its internal table lock immediately after the call of [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), and deadlocks can very easily happen. **InnoDB** does not acquire the internal table lock at all if [**autocommit = 1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_autocommit), to help old applications avoid unnecessary deadlocks.

[**ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) does not release table locks.

#### LOCK TABLES and Triggers

If you lock a table explicitly with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), any tables used in triggers are also locked implicitly:

The locks are taken as the same time as those acquired explicitly with the [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement.

The lock on a table used in a trigger depends on whether the table is used only for reading. If so, a read lock suffices. Otherwise, a write lock is used.

If a table is locked explicitly for reading with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), but needs to be locked for writing because it might be modified within a trigger, a write lock is taken rather than a read lock. (That is, an implicit write lock needed due to the table's appearance within a trigger causes an explicit read lock request for the table to be converted to a write lock request.)

Suppose that you lock two tables, **t1** and **t2**, using this statement:

LOCK TABLES t1 WRITE, t2 READ;

If **t1** or **t2** have any triggers, tables used within the triggers are also locked. Suppose that **t1** has a trigger defined like this:

CREATE TRIGGER t1\_a\_ins AFTER INSERT ON t1 FOR EACH ROW

BEGIN

UPDATE t4 SET count = count+1

WHERE id = NEW.id AND EXISTS (SELECT a FROM t3);

INSERT INTO t2 VALUES(1, 2);

END;

The result of the [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement is that **t1** and **t2** are locked because they appear in the statement, and **t3** and **t4** are locked because they are used within the trigger:

**t1** is locked for writing per the **WRITE** lock request.

**t2** is locked for writing, even though the request is for a **READ** lock. This occurs because **t2** is inserted into within the trigger, so the **READ** request is converted to a **WRITE** request.

**t3** is locked for reading because it is only read from within the trigger.

**t4** is locked for writing because it might be updated within the trigger.

#### Table-Locking Restrictions and Conditions

You can safely use [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) to terminate a session that is waiting for a table lock. See [Section 13.7.8.4, “KILL Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill).

[**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) and [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) cannot be used within stored programs.

Tables in the **performance\_schema** database cannot be locked with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), except the **setup\_*xxx*** tables.

The following statements are prohibited while a [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement is in effect: [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**CREATE TABLE ... LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view), [**DROP VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-view), and DDL statements on stored functions and procedures and events.

For some operations, system tables in the **mysql** database must be accessed. For example, the [**HELP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#help) statement requires the contents of the server-side help tables, and [**CONVERT\_TZ()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_convert-tz) might need to read the time zone tables. The server implicitly locks the system tables for reading as necessary so that you need not lock them explicitly. These tables are treated as just described:

mysql.help\_category

mysql.help\_keyword

mysql.help\_relation

mysql.help\_topic

mysql.time\_zone

mysql.time\_zone\_leap\_second

mysql.time\_zone\_name

mysql.time\_zone\_transition

mysql.time\_zone\_transition\_type

If you want to explicitly place a **WRITE** lock on any of those tables with a [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) statement, the table must be the only one locked; no other table can be locked with the same statement.

Normally, you do not need to lock tables, because all single [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements are atomic; no other session can interfere with any other currently executing SQL statement. However, there are a few cases when locking tables may provide an advantage:

If you are going to run many operations on a set of **MyISAM** tables, it is much faster to lock the tables you are going to use. Locking **MyISAM** tables speeds up inserting, updating, or deleting on them because MySQL does not flush the key cache for the locked tables until [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) is called. Normally, the key cache is flushed after each SQL statement.

The downside to locking the tables is that no session can update a **READ**-locked table (including the one holding the lock) and no session can access a **WRITE**-locked table other than the one holding the lock.

If you are using tables for a nontransactional storage engine, you must use [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) if you want to ensure that no other session modifies the tables between a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). The example shown here requires [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to execute safely:

LOCK TABLES trans READ, customer WRITE;

SELECT SUM(value) FROM trans WHERE customer\_id=***some\_id***;

UPDATE customer

SET total\_value=***sum\_from\_previous\_statement***

WHERE customer\_id=***some\_id***;

UNLOCK TABLES;

Without [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), it is possible that another session might insert a new row in the **trans** table between execution of the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements.

You can avoid using [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) in many cases by using relative updates (**UPDATE customer SET *value*=*value*+*new\_value***) or the [**LAST\_INSERT\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_last-insert-id) function.

You can also avoid locking tables in some cases by using the user-level advisory lock functions [**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) and [**RELEASE\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_release-lock). These locks are saved in a hash table in the server and implemented with **pthread\_mutex\_lock()** and **pthread\_mutex\_unlock()** for high speed. See [Section 12.15, “Locking Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#locking-functions).

See [Section 8.11.1, “Internal Locking Methods”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#internal-locking), for more information on locking policy.

### 13.3.7 SET TRANSACTION Statement

SET [GLOBAL | SESSION] TRANSACTION

***transaction\_characteristic*** [, ***transaction\_characteristic***] ...

***transaction\_characteristic***: {

ISOLATION LEVEL ***level***

| ***access\_mode***

}

***level***: {

REPEATABLE READ

| READ COMMITTED

| READ UNCOMMITTED

| SERIALIZABLE

}

***access\_mode***: {

READ WRITE

| READ ONLY

}

This statement specifies [transaction](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_transaction) characteristics. It takes a list of one or more characteristic values separated by commas. Each characteristic value sets the transaction [isolation level](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_isolation_level) or access mode. The isolation level is used for operations on [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables. The access mode specifies whether transactions operate in read/write or read-only mode.

In addition, [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) can include an optional **GLOBAL** or **SESSION** keyword to indicate the scope of the statement.

[Transaction Isolation Levels](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction-isolation-level)

[Transaction Access Mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction-access-mode)

[Transaction Characteristic Scope](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction-scope)

#### Transaction Isolation Levels

To set the transaction isolation level, use an **ISOLATION LEVEL *level*** clause. It is not permitted to specify multiple **ISOLATION LEVEL** clauses in the same [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) statement.

The default isolation level is [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read). Other permitted values are [**READ COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed), [**READ UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted), and [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable). For information about these isolation levels, see [Section 15.7.2.1, “Transaction Isolation Levels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-transaction-isolation-levels).

#### Transaction Access Mode

To set the transaction access mode, use a **READ WRITE** or **READ ONLY** clause. It is not permitted to specify multiple access-mode clauses in the same [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) statement.

By default, a transaction takes place in read/write mode, with both reads and writes permitted to tables used in the transaction. This mode may be specified explicitly using [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) with an access mode of **READ WRITE**.

If the transaction access mode is set to **READ ONLY**, changes to tables are prohibited. This may enable storage engines to make performance improvements that are possible when writes are not permitted.

In read-only mode, it remains possible to change tables created with the **TEMPORARY** keyword using DML statements. Changes made with DDL statements are not permitted, just as with permanent tables.

The **READ WRITE** and **READ ONLY** access modes also may be specified for an individual transaction using the [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) statement.

#### Transaction Characteristic Scope

You can set transaction characteristics globally, for the current session, or for the next transaction only:

With the **GLOBAL** keyword:

The statement applies globally for all subsequent sessions.

Existing sessions are unaffected.

With the **SESSION** keyword:

The statement applies to all subsequent transactions performed within the current session.

The statement is permitted within transactions, but does not affect the current ongoing transaction.

If executed between transactions, the statement overrides any preceding statement that sets the next-transaction value of the named characteristics.

Without any **SESSION** or **GLOBAL** keyword:

The statement applies only to the next single transaction performed within the session.

Subsequent transactions revert to using the session value of the named characteristics.

The statement is not permitted within transactions:

mysql> **START TRANSACTION;**

Query OK, 0 rows affected (0.02 sec)

mysql> **SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;**

ERROR 1568 (25001): Transaction characteristics can't be changed

while a transaction is in progress

A change to global transaction characteristics requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege). Any session is free to change its session characteristics (even in the middle of a transaction), or the characteristics for its next transaction (prior to the start of that transaction).

To set the global isolation level at server startup, use the [--transaction-isolation=***level***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_transaction-isolation) option on the command line or in an option file. Values of ***level*** for this option use dashes rather than spaces, so the permissible values are [**READ-UNCOMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-uncommitted), [**READ-COMMITTED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_read-committed), [**REPEATABLE-READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read), or [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable).

Similarly, to set the global transaction access mode at server startup, use the [--transaction-read-only](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_transaction-read-only) option. The default is **OFF** (read/write mode) but the value can be set to **ON** for a mode of read only.

For example, to set the isolation level to [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) and the access mode to **READ WRITE**, use these lines in the **[mysqld]** section of an option file:

[mysqld]

transaction-isolation = REPEATABLE-READ

transaction-read-only = OFF

At runtime, characteristics at the global, session, and next-transaction scope levels can be set indirectly using the [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) statement, as described previously. They can also be set directly using the [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement to assign values to the [**transaction\_isolation**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_transaction_isolation) and [**transaction\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_transaction_read_only) system variables:

[**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) permits optional **GLOBAL** and **SESSION** keywords for setting transaction characteristics at different scope levels.

The [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement for assigning values to the [**transaction\_isolation**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_transaction_isolation) and [**transaction\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_transaction_read_only) system variables has syntaxes for setting these variables at different scope levels.

The following tables show the characteristic scope level set by each [**SET TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) and variable-assignment syntax.

**Table 13.9 SET TRANSACTION Syntax for Transaction Characteristics**

| **Syntax** | **Affected Characteristic Scope** |
| --- | --- |
| **SET GLOBAL TRANSACTION *transaction\_characteristic*** | Global |
| **SET SESSION TRANSACTION *transaction\_characteristic*** | Session |
| **SET TRANSACTION *transaction\_characteristic*** | Next transaction only |

**Table 13.10 SET Syntax for Transaction Characteristics**

| **Syntax** | **Affected Characteristic Scope** |
| --- | --- |
| **SET GLOBAL *var\_name* = *value*** | Global |
| **SET @@GLOBAL.*var\_name* = *value*** | Global |
| **SET PERSIST *var\_name* = *value*** | Global |
| **SET @@PERSIST.*var\_name* = *value*** | Global |
| **SET PERSIST\_ONLY *var\_name* = *value*** | No runtime effect |
| **SET @@PERSIST\_ONLY.*var\_name* = *value*** | No runtime effect |
| **SET SESSION *var\_name* = *value*** | Session |
| **SET @@SESSION.*var\_name* = *value*** | Session |
| **SET *var\_name* = *value*** | Session |
| **SET @@*var\_name* = *value*** | Next transaction only |

It is possible to check the global and session values of transaction characteristics at runtime:

SELECT @@GLOBAL.transaction\_isolation, @@GLOBAL.transaction\_read\_only;

SELECT @@SESSION.transaction\_isolation, @@SESSION.transaction\_read\_only;

### 13.3.8 XA Transactions

[13.3.8.1 XA Transaction SQL Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements)

[13.3.8.2 XA Transaction States](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-states)

[13.3.8.3 Restrictions on XA Transactions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-restrictions)

Support for [XA](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_xa) transactions is available for the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) storage engine. The MySQL 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>. Limitations of the current XA implementation are described in [Section 13.3.8.3, “Restrictions on XA Transactions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-restrictions).

On the client side, there are no special requirements. The XA interface to a MySQL server consists of SQL statements that begin with the **XA** keyword. MySQL client programs must be able to send SQL statements and to understand the semantics of the XA statement interface. They do not need be linked against a recent client library. Older client libraries also work.

Among the MySQL Connectors, MySQL Connector/J 5.0.0 and higher supports XA directly, by means of a class interface that handles the XA SQL statement interface for you.

XA supports distributed transactions, that is, the ability to permit multiple separate transactional resources to participate in a global transaction. Transactional resources often are RDBMSs but may be other kinds of resources.

A global transaction involves several actions that are transactional in themselves, but that all must either complete successfully as a group, or all be rolled back as a group. In essence, this extends ACID properties “up a level” so that multiple ACID transactions can be executed in concert as components of a global operation that also has ACID properties. (As with nondistributed transactions, [**SERIALIZABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_serializable) may be preferred if your applications are sensitive to read phenomena. [**REPEATABLE READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#isolevel_repeatable-read) may not be sufficient for distributed transactions.)

Some examples of distributed transactions:

An application may act as an integration tool that combines a messaging service with an RDBMS. The application makes sure that transactions dealing with message sending, retrieval, and processing that also involve a transactional database all happen in a global transaction. You can think of this as “transactional email.”

An application performs actions that involve different database servers, such as a MySQL server and an Oracle server (or multiple MySQL servers), where actions that involve multiple servers must happen as part of a global transaction, rather than as separate transactions local to each server.

A bank keeps account information in an RDBMS and distributes and receives money through automated teller machines (ATMs). It is necessary to ensure that ATM actions are correctly reflected in the accounts, but this cannot be done with the RDBMS alone. A global transaction manager integrates the ATM and database resources to ensure overall consistency of financial transactions.

Applications that use global transactions involve one or more Resource Managers and a Transaction Manager:

A Resource Manager (RM) provides access to transactional resources. A database server is one kind of resource manager. It must be possible to either commit or roll back transactions managed by the RM.

A Transaction Manager (TM) coordinates the transactions that are part of a global transaction. It communicates with the RMs that handle each of these transactions. The individual transactions within a global transaction are “branches” of the global transaction. Global transactions and their branches are identified by a naming scheme described later.

The MySQL implementation of XA enables a MySQL server to act as a Resource Manager that handles XA transactions within a global transaction. A client program that connects to the MySQL server acts as the Transaction Manager.

To carry out a global transaction, it is necessary to know which components are involved, and bring each component to a point when it can be committed or rolled back. Depending on what each component reports about its ability to succeed, they must all commit or roll back as an atomic group. That is, either all components must commit, or all components must roll back. To manage a global transaction, it is necessary to take into account that any component or the connecting network might fail.

The process for executing a global transaction uses two-phase commit (2PC). This takes place after the actions performed by the branches of the global transaction have been executed.

In the first phase, all branches are prepared. That is, they are told by the TM to get ready to commit. Typically, this means each RM that manages a branch records the actions for the branch in stable storage. The branches indicate whether they are able to do this, and these results are used for the second phase.

In the second phase, the TM tells the RMs whether to commit or roll back. If all branches indicated when they were prepared that they were able to commit, all branches are told to commit. If any branch indicated when it was prepared that it was not able to commit, all branches are told to roll back.

In some cases, a global transaction might use one-phase commit (1PC). For example, when a Transaction Manager finds that a global transaction consists of only one transactional resource (that is, a single branch), that resource can be told to prepare and commit at the same time.

#### 13.3.8.1 XA Transaction SQL Statements

To perform XA transactions in MySQL, use the following statements:

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 [CONVERT XID]

For [**XA START**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements), the **JOIN** and **RESUME** clauses are recognized but have no effect.

For [**XA END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) the **SUSPEND [FOR MIGRATE]** clause is recognized but has no effect.

Each XA statement begins with the **XA** keyword, and most of them require an ***xid*** value. An ***xid*** is an XA transaction identifier. It indicates which transaction the statement applies to. ***xid*** values are supplied by the client, or generated by the MySQL server. An ***xid*** value has from one to three parts:

***xid***: ***gtrid*** [, ***bqual*** [, ***formatID*** ]]

***gtrid*** is a global transaction identifier, ***bqual*** is a branch qualifier, and ***formatID*** is a number that identifies the format used by the ***gtrid*** and ***bqual*** values. As indicated by the syntax, ***bqual*** and ***formatID*** are optional. The default ***bqual*** value is **''** if not given. The default ***formatID*** value is 1 if not given.

***gtrid*** and ***bqual*** must be string literals, each up to 64 bytes (not characters) long. ***gtrid*** and ***bqual*** can be specified in several ways. You can use a quoted string (**'ab'**), hex string (**X'6162'**, **0x6162**), or bit value (**b'*nnnn*'**).

***formatID*** is an unsigned integer.

The ***gtrid*** and ***bqual*** values are interpreted in bytes by the MySQL server's underlying XA support routines. However, while an SQL statement containing an XA statement is being parsed, the server works with some specific character set. To be safe, write ***gtrid*** and ***bqual*** as hex strings.

***xid*** values typically are generated by the Transaction Manager. Values generated by one TM must be different from values generated by other TMs. A given TM must be able to recognize its own ***xid*** values in a list of values returned by the [**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) statement.

[**XA START *xid***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) starts an XA transaction with the given ***xid*** value. Each XA transaction must have a unique ***xid*** value, so the value must not currently be used by another XA transaction. Uniqueness is assessed using the ***gtrid*** and ***bqual*** values. All following XA statements for the XA transaction must be specified using the same ***xid*** value as that given in the [**XA START**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) statement. If you use any of those statements but specify an ***xid*** value that does not correspond to some existing XA transaction, an error occurs.

One or more XA transactions can be part of the same global transaction. All XA transactions within a given global transaction must use the same ***gtrid*** value in the ***xid*** value. For this reason, ***gtrid*** values must be globally unique so that there is no ambiguity about which global transaction a given XA transaction is part of. The ***bqual*** part of the ***xid*** value must be different for each XA transaction within a global transaction. (The requirement that ***bqual*** values be different is a limitation of the current MySQL XA implementation. It is not part of the XA specification.)

The [**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) statement returns information for those XA transactions on the MySQL server that are in the **PREPARED** state. (See [Section 13.3.8.2, “XA Transaction States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-states).) The output includes a row for each such XA transaction on the server, regardless of which client started it.

[**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) requires the [**XA\_RECOVER\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_xa-recover-admin) privilege. This privilege requirement prevents users from discovering the XID values for outstanding prepared XA transactions other than their own. It does not affect normal commit or rollback of an XA transaction because the user who started it knows its XID.

[**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) output rows look like this (for an example ***xid*** value consisting of the parts **'abc'**, **'def'**, and **7**):

mysql> **XA RECOVER;**

+----------+--------------+--------------+--------+

| formatID | gtrid\_length | bqual\_length | data |

+----------+--------------+--------------+--------+

| 7 | 3 | 3 | abcdef |

+----------+--------------+--------------+--------+

The output columns have the following meanings:

**formatID** is the ***formatID*** part of the transaction ***xid***

**gtrid\_length** is the length in bytes of the ***gtrid*** part of the ***xid***

**bqual\_length** is the length in bytes of the ***bqual*** part of the ***xid***

**data** is the concatenation of the ***gtrid*** and ***bqual*** parts of the ***xid***

XID values may contain nonprintable characters. [**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) permits an optional **CONVERT XID** clause so that clients can request XID values in hexadecimal.

#### 13.3.8.2 XA Transaction States

An XA transaction progresses through the following states:

Use [**XA START**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) to start an XA transaction and put it in the **ACTIVE** state.

For an **ACTIVE** XA transaction, issue the SQL statements that make up the transaction, and then issue an [**XA END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) statement. [**XA END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) puts the transaction in the **IDLE** state.

For an **IDLE** XA transaction, you can issue either an [**XA PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) statement or an **XA COMMIT ... ONE PHASE** statement:

[**XA PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) puts the transaction in the **PREPARED** state. An [**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) statement at this point includes the transaction's ***xid*** value in its output, because [**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) lists all XA transactions that are in the **PREPARED** state.

**XA COMMIT ... ONE PHASE** prepares and commits the transaction. The ***xid*** value is not listed by [**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) because the transaction terminates.

For a **PREPARED** XA transaction, you can issue an [**XA COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) statement to commit and terminate the transaction, or [**XA ROLLBACK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) to roll back and terminate the transaction.

Here is a simple XA transaction that inserts a row into a table as part of a global transaction:

mysql> **XA START 'xatest';**

Query OK, 0 rows affected (0.00 sec)

mysql> **INSERT INTO mytable (i) VALUES(10);**

Query OK, 1 row affected (0.04 sec)

mysql> **XA END 'xatest';**

Query OK, 0 rows affected (0.00 sec)

mysql> **XA PREPARE 'xatest';**

Query OK, 0 rows affected (0.00 sec)

mysql> **XA COMMIT 'xatest';**

Query OK, 0 rows affected (0.00 sec)

Within the context of a given client connection, XA transactions and local (non-XA) transactions are mutually exclusive. For example, if [**XA START**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) has been issued to begin an XA transaction, a local transaction cannot be started until the XA transaction has been committed or rolled back. Conversely, if a local transaction has been started with [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit), no XA statements can be used until the transaction has been committed or rolled back.

If an XA transaction is in the **ACTIVE** state, you cannot issue any statements that cause an implicit commit. That would violate the XA contract because you could not roll back the XA transaction. Trying to execute such a statement raises the following error:

ERROR 1399 (XAE07): XAER\_RMFAIL: The command cannot be executed

when global transaction is in the ACTIVE state

Statements to which the preceding remark applies are listed at [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

#### 13.3.8.3 Restrictions on XA Transactions

XA transaction support is limited to the **InnoDB** storage engine.

For “external XA,” a MySQL server acts as a Resource Manager and client programs act as Transaction Managers. For “Internal XA”, storage engines within a MySQL server act as RMs, and the server itself acts as a TM. Internal XA support is limited by the capabilities of individual storage engines. Internal XA is required for handling XA transactions that involve more than one storage engine. The implementation of internal XA requires that a storage engine support two-phase commit at the table handler level, and currently this is true only for **InnoDB**.

For [**XA START**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements), the **JOIN** and **RESUME** clauses are recognized but have no effect.

For [**XA END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) the **SUSPEND [FOR MIGRATE]** clause is recognized but has no effect.

The requirement that the ***bqual*** part of the ***xid*** value be different for each XA transaction within a global transaction is a limitation of the current MySQL XA implementation. It is not part of the XA specification.

An XA transaction is written to the binary log in two parts. When **XA PREPARE**is issued, the first part of the transaction up to **XA PREPARE** is written using an initial GTID. A **XA\_prepare\_log\_event** is used to identify such transactions in the binary log. When **XA COMMIT** or **XA ROLLBACK** is issued, a second part of the transaction containing only the **XA COMMIT** or **XA ROLLBACK** statement is written using a second GTID. Note that the initial part of the transaction, identified by **XA\_prepare\_log\_event**, is not necessarily followed by its **XA COMMIT** or **XA ROLLBACK**, which can cause interleaved binary logging of any two XA transactions. The two parts of the XA transaction can even appear in different binary log files. This means that an XA transaction in **PREPARED** state is now persistent until an explicit **XA COMMIT** or **XA ROLLBACK** statement is issued, ensuring that XA transactions are compatible with replication.

On a replica, immediately after the XA transaction is prepared, it is detached from the replication applier thread, and can be committed or rolled back by any thread on the replica. This means that the same XA transaction can appear in the [**events\_transactions\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-transactions-current-table) table with different states on different threads. The [**events\_transactions\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-events-transactions-current-table) table displays the current status of the most recent monitored transaction event on the thread, and does not update this status when the thread is idle. So the XA transaction can still be displayed in the **PREPARED** state for the original applier thread, after it has been processed by another thread. To positively identify XA transactions that are still in the **PREPARED** state and need to be recovered, use the [**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) statement rather than the Performance Schema transaction tables.

The following restrictions exist for using XA transactions:

XA transactions are not fully resilient to an unexpected halt with respect to the binary log. If there is an unexpected halt while the server is in the middle of executing an **XA PREPARE**, **XA COMMIT**, **XA ROLLBACK**, or **XA COMMIT ... ONE PHASE** statement, the server might not be able to recover to a correct state, leaving the server and the binary log in an inconsistent state. In this situation, the binary log might either contain extra XA transactions that are not applied, or miss XA transactions that are applied. Also, if GTIDs are enabled, after recovery **@@GLOBAL.GTID\_EXECUTED** might not correctly describe the transactions that have been applied. Note that if an unexpected halt occurs before **XA PREPARE**, between **XA PREPARE** and **XA COMMIT** (or **XA ROLLBACK**), or after **XA COMMIT** (or **XA ROLLBACK**), the server and binary log are correctly recovered and taken to a consistent state.

The use of replication filters or binary log filters in combination with XA transactions is not supported. Filtering of tables could cause an XA transaction to be empty on a replica, and empty XA transactions are not supported. Also, with the replica's connection metadata repository and applier metadata repository stored in [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, which became the default in MySQL 8.0, the internal state of the data engine transaction is changed following a filtered XA transaction, and can become inconsistent with the replication transaction context state.

The error [**ER\_XA\_REPLICATION\_FILTERS**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_xa_replication_filters) is logged whenever an XA transaction is impacted by a replication filter, whether or not the transaction was empty as a result. If the transaction is not empty, the replica is able to continue running, but you should take steps to discontinue the use of replication filters with XA transactions in order to avoid potential issues. If the transaction is empty, the replica stops. In that event, the replica might be in an undetermined state in which the consistency of the replication process might be compromised. In particular, the **gtid\_executed** set on a replica of the replica might be inconsistent with that on the source. To resolve this situation, isolate the source and stop all replication, then check GTID consistency across the replication topology. Undo the XA transaction that generated the error message, then restart replication.

XA transactions are considered unsafe for statement-based replication. If two XA transactions committed in parallel on the source are being prepared on the replica in the inverse order, locking dependencies can occur that cannot be safely resolved, and it is possible for replication to fail with deadlock on the replica. This situation can occur for a single-threaded or multithreaded replica. When [**binlog\_format=STATEMENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, a warning is issued for DML statements inside XA transactions. When [**binlog\_format=MIXED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) or [**binlog\_format=ROW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_format) is set, DML statements inside XA transactions are logged using row-based replication, and the potential issue is not present.

**Note**

Prior to MySQL 5.7.7, XA transactions were not compatible with replication at all. This was because an XA transaction that was in **PREPARED** state would be rolled back on clean server shutdown or client disconnect. Similarly, an XA transaction that was in **PREPARED** state would still exist in **PREPARED** state in case the server was shutdown abnormally and then started again, but the contents of the transaction could not be written to the binary log. In both of these situations the XA transaction could not be replicated correctly.

## 13.4 Replication Statements

[13.4.1 SQL Statements for Controlling Source Servers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-statements-master)

[13.4.2 SQL Statements for Controlling Replica Servers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-statements-replica)

[13.4.3 SQL Statements for Controlling Group Replication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-statements-group)

Replication can be controlled through the SQL interface using the statements described in this section. Statements are split into a group which controls source servers, a group which controls replica servers, and a group which can be applied to any replication servers.

### 13.4.1 SQL Statements for Controlling Source Servers

[13.4.1.1 PURGE BINARY LOGS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs)

[13.4.1.2 RESET MASTER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master)

[13.4.1.3 SET sql\_log\_bin Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-sql-log-bin)

This section discusses statements for managing replication source servers. [Section 13.4.2, “SQL Statements for Controlling Replica Servers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-statements-replica), discusses statements for managing replica servers.

In addition to the statements described here, the following [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statements are used with source servers in replication. For information about these statements, see [Section 13.7.7, “SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show).

[**SHOW BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binary-logs)

[**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events)

[**SHOW MASTER STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status)

[**SHOW REPLICAS | SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas)

#### 13.4.1.1 PURGE BINARY LOGS Statement

PURGE { BINARY | MASTER } LOGS {

TO '***log\_name***'

| BEFORE ***datetime\_expr***

}

The binary log is a set of files that contain information about data modifications made by the MySQL server. The log consists of a set of binary log files, plus an index file (see [Section 5.4.4, “The Binary Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log)).

The [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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.

[**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs) requires the [**BINLOG\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_binlog-admin) privilege. This statement has no effect if the server was not started with the [--log-bin](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_log-bin) option to enable binary logging.

Examples:

PURGE BINARY LOGS TO 'mysql-bin.010';

PURGE BINARY LOGS BEFORE '2019-04-02 22:46:26';

The **BEFORE** variant's ***datetime\_expr*** argument should evaluate to a [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value (a value in **'*YYYY-MM-DD hh:mm:ss*'** format).

This statement is safe to run while replicas are replicating. You need not stop them. If you have an active replica that currently is reading one of the log files you are trying to delete, this statement does not delete the log file that is in use or any log files later than that one, but it deletes any earlier log files. A warning message is issued in this situation. However, if a replica is not connected and you happen to purge one of the log files it has yet to read, the replica cannot replicate after it reconnects.

To safely purge binary log files, follow this procedure:

On each replica, use [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) to check which log file it is reading.

Obtain a listing of the binary log files on the source with [**SHOW BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binary-logs).

Determine the earliest log file among all the replicas. This is the target file. If all the replicas are up to date, this is the last log file on the list.

Make a backup of all the log files you are about to delete. (This step is optional, but always advisable.)

Purge all log files up to but not including the target file.

**PURGE BINARY LOGS TO** and **PURGE BINARY LOGS BEFORE** both fail with an error when binary log files listed in the .index file had been removed from the system by some other means (such as using **rm** on Linux). (Bug #18199, Bug #18453) To handle such errors, edit the .index file (which is a simple text file) manually to ensure that it lists only the binary log files that are actually present, then run again the [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs) statement that failed.

Binary log files are automatically removed after the server's binary log expiration period. Removal of the files can take place at startup and when the binary log is flushed. The default binary log expiration period is 30 days. You can specify an alternative expiration period using the [**binlog\_expire\_logs\_seconds**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_expire_logs_seconds) system variable. If you are using replication, you should specify an expiration period that is no lower than the maximum amount of time your replicas might lag behind the source.

#### 13.4.1.2 RESET MASTER Statement

RESET MASTER [TO ***binary\_log\_file\_index\_number***]

**Warning**

Use this statement with caution to ensure you do not lose any wanted binary log file data and GTID execution history.

[**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

For a server where binary logging is enabled ([**log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_log_bin) is **ON**), **RESET MASTER** deletes all existing binary log files and resets the binary log index file, resetting the server to its state before binary logging was started. A new empty binary log file is created so that binary logging can be restarted.

For a server where GTIDs are in use ([**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is **ON**), issuing **RESET MASTER** resets the GTID execution history. The value of the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variable is set to an empty string (**''**), the global value (but not the session value) of the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable is set to an empty string, and the **mysql.gtid\_executed** table is cleared (see [mysql.gtid\_executed Table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-gtid-executed-table)). If the GTID-enabled server has binary logging enabled, [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) also resets the binary log as described above. Note that [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) is the method to reset the GTID execution history even if the GTID-enabled server is a replica where binary logging is disabled; [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) has no effect on the GTID execution history. For more information on resetting the GTID execution history, see [Resetting the GTID Execution History](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-execution-history).

Issuing **RESET MASTER** without the optional **TO** clause 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 starting at **1**. Use the optional **TO** clause to start the binary log file index from a number other than **1** after the reset.

Using **RESET MASTER** with the **TO** clause to specify a binary log file index number to start from simplifies failover by providing a single statement alternative to the [**FLUSH BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-binary-logs) and [**PURGE BINARY LOGS TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs) statements. Check that you are using a reasonable value for the index number. If you enter an incorrect value, you can correct this by issuing another [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) statement with or without the **TO** clause. If you do not correct a value that is out of range, the server cannot be restarted.

The following example demonstrates **TO** clause usage:

RESET MASTER TO 1234;

SHOW BINARY LOGS;

+-------------------+-----------+-----------+

| Log\_name | File\_size | Encrypted |

+-------------------+-----------+-----------+

| source-bin.001234 | 154 | No |

+-------------------+-----------+-----------+

**Important**

The effects of [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) without the **TO** clause differ from those of [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs) in 2 key ways:

[**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) removes all binary log files that are listed in the index file, leaving only a single, empty binary log file with a numeric suffix of **.000001**, whereas the numbering is not reset by [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs).

[**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) is not intended to be used while any replicas are running. The behavior of [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) when used while replicas are running is undefined (and thus unsupported), whereas [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs) may be safely used while replicas are running.

See also [Section 13.4.1.1, “PURGE BINARY LOGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs).

[**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) without the **TO** clause can prove useful when you first set up a source and replica, so that you can verify the setup as follows:

Start the source and replica, and start replication (see [Section 17.1.2, “Setting Up Binary Log File Position Based Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-howto)).

Execute a few test queries on the source.

Check that the queries were replicated to the replica.

When replication is running correctly, issue [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) followed by [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) on the replica, then verify that no unwanted data from the test queries exists on the replica.

Issue [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) on the source to clean up the test queries.

After verifying the setup, resetting the source and replica and ensuring that no unwanted data or binary log files generated by testing remain on the source or replica, you can start the replica and begin replicating.

#### 13.4.1.3 SET sql\_log\_bin Statement

SET sql\_log\_bin = {OFF|ON}

The [**sql\_log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_log_bin) variable controls whether logging to the binary log is enabled for the current session (assuming that the binary log itself is enabled). The default value is **ON**. To disable or enable binary logging for the current session, set the session [**sql\_log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_log_bin) variable to **OFF** or **ON**.

Set this variable to **OFF** for a session to temporarily disable binary logging while making changes to the source that you do not want replicated to the replica.

Setting the session value of this system variable is a restricted operation. The session user must have privileges sufficient to set restricted session variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

It is not possible to set the session value of [**sql\_log\_bin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_log_bin) within a transaction or subquery.

Setting this variable to ***OFF*** prevents new GTIDs from being assigned to transactions in the binary log. If you are using GTIDs for replication, this means that even when binary logging is later enabled again, the GTIDs written into the log from this point do not account for any transactions that occurred in the meantime, so in effect those transactions are lost.

[**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) adds a **SET @@SESSION.sql\_log\_bin=0** statement to a dump file from a server where GTIDs are in use, which disables binary logging while the dump file is being reloaded. The statement prevents new GTIDs from being generated and assigned to the transactions in the dump file as they are executed, so that the original GTIDs for the transactions are used.

### 13.4.2 SQL Statements for Controlling Replica Servers

[13.4.2.1 CHANGE MASTER TO Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to)

[13.4.2.2 CHANGE REPLICATION FILTER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter)

[13.4.2.3 CHANGE REPLICATION SOURCE TO Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to)

[13.4.2.4 MASTER\_POS\_WAIT() Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#master-pos-wait)

[13.4.2.5 RESET REPLICA | SLAVE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica)

[13.4.2.6 RESET SLAVE | REPLICA Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-slave)

[13.4.2.7 START REPLICA | SLAVE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica)

[13.4.2.8 START SLAVE | REPLICA Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-slave)

[13.4.2.9 STOP REPLICA | SLAVE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica)

[13.4.2.10 STOP SLAVE | REPLICA Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-slave)

[13.4.2.11 Functions which Configure the Source List](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-functions-source-list)

This section discusses statements for managing replica servers. [Section 13.4.1, “SQL Statements for Controlling Source Servers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replication-statements-master), discusses statements for managing source servers.

In addition to the statements described here, [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) and [**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events) are also used with replicas. For information about these statements, see [Section 13.7.7.35, “SHOW REPLICA | SLAVE STATUS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status), and [Section 13.7.7.32, “SHOW RELAYLOG EVENTS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events).

#### 13.4.2.1 CHANGE MASTER TO Statement

CHANGE MASTER TO ***option*** [, ***option***] ... [ ***channel\_option*** ]

***option***: {

MASTER\_BIND = '***interface\_name***'

| MASTER\_HOST = '***host\_name***'

| MASTER\_USER = '***user\_name***'

| MASTER\_PASSWORD = '***password***'

| MASTER\_PORT = ***port\_num***

| PRIVILEGE\_CHECKS\_USER = {'***account***' | NULL}

| REQUIRE\_ROW\_FORMAT = {0|1}

| REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK = {STREAM | ON | OFF}

| ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS = {OFF | LOCAL | ***uuid***}

| MASTER\_LOG\_FILE = '***source\_log\_name***'

| MASTER\_LOG\_POS = ***source\_log\_pos***

| MASTER\_AUTO\_POSITION = {0|1}

| RELAY\_LOG\_FILE = '***relay\_log\_name***'

| RELAY\_LOG\_POS = ***relay\_log\_pos***

| MASTER\_HEARTBEAT\_PERIOD = ***interval***

| MASTER\_CONNECT\_RETRY = ***interval***

| MASTER\_RETRY\_COUNT = ***count***

| SOURCE\_CONNECTION\_AUTO\_FAILOVER = {0|1}

| MASTER\_DELAY = ***interval***

| MASTER\_COMPRESSION\_ALGORITHMS = '***value***'

| MASTER\_ZSTD\_COMPRESSION\_LEVEL = ***level***

| 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\_TLS\_VERSION = '***protocol\_list***'

| MASTER\_TLS\_CIPHERSUITES = '***ciphersuite\_list***'

| MASTER\_PUBLIC\_KEY\_PATH = '***key\_file\_name***'

| GET\_MASTER\_PUBLIC\_KEY = {0|1}

| NETWORK\_NAMESPACE = '***namespace***'

| IGNORE\_SERVER\_IDS = (***server\_id\_list***)

}

***channel\_option***:

FOR CHANNEL ***channel***

***server\_id\_list***:

[***server\_id*** [, ***server\_id***] ... ]

[**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) changes the parameters that the replica server uses for connecting to the source and for reading data from the source. It also updates the contents of the replication metadata repositories (see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs)). From MySQL 8.0.23, use [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) in place of [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to), which is deprecated from that release. In releases before MySQL 8.0.23, use [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to).

You can issue **CHANGE MASTER TO** statements on a running replica without first stopping it, depending on the states of the replication SQL thread and replication I/O thread. The rules governing such use are provided later in this section. [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) requires the [**REPLICATION\_SLAVE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

When using a multithreaded replica (in other words [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) is greater than 0), stopping the replica can cause “gaps” in the sequence of transactions that have been executed from the relay log, regardless of whether the replica was stopped intentionally or otherwise. When such gaps exist, issuing [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) fails. The solution in this situation is to issue [**START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) which ensures that the gaps are closed.

The optional **FOR CHANNEL *channel*** clause enables you to name which replication channel the statement applies to. Providing a **FOR CHANNEL *channel*** clause applies the **CHANGE MASTER TO** statement to a specific replication channel, and is used to add a new channel or modify an existing channel. For example, to add a new channel called **channel2**:

CHANGE MASTER TO MASTER\_HOST=host1, MASTER\_PORT=3002 FOR CHANNEL 'channel2'

If no clause is named and no extra channels exist, the statement applies to the default channel.

When using multiple replication channels, if a **CHANGE MASTER TO** statement does not name a channel using a **FOR CHANNEL *channel*** clause, an error occurs. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels) for more information.

Values used for **MASTER\_HOST** and other **CHANGE MASTER TO** options are checked for linefeed (**\n** or **0x0A**) characters. The presence of such characters in these values causes the statement to fail with [**ER\_MASTER\_INFO**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_master_info).

Invoking [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) causes the previous values for **MASTER\_HOST**, **MASTER\_PORT**, **MASTER\_LOG\_FILE**, and **MASTER\_LOG\_POS** to be written to the error log, along with other information about the replica's state prior to execution.

**CHANGE MASTER TO** causes an implicit commit of an ongoing transaction. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

For some of the options of the **CHANGE MASTER TO** statement, you must issue a [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement prior to issuing a [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (and a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement afterwards). Sometimes, you only need to stop the replication SQL thread or the replication I/O thread, not both:

When the SQL thread is stopped, you can execute **CHANGE MASTER TO** using any combination that is otherwise allowed of **RELAY\_LOG\_FILE**, **RELAY\_LOG\_POS**, and **MASTER\_DELAY** options, even if the replication I/O thread is running. No other options may be used with this statement when the I/O thread is running.

When the I/O thread is stopped, you can execute **CHANGE MASTER TO** using any of the options for this statement (in any allowed combination) except **RELAY\_LOG\_FILE**, **RELAY\_LOG\_POS**, **MASTER\_DELAY**, or **MASTER\_AUTO\_POSITION = 1** even when the SQL thread is running.

Both the SQL thread and the I/O thread must be stopped before issuing a **CHANGE MASTER TO** statement that employs **MASTER\_AUTO\_POSITION = 1** or **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**.

You can check the current state of the replication SQL thread and replication I/O thread using [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). Note that the Group Replication applier channel (**group\_replication\_applier**) has no I/O thread, only an SQL thread.

For more information, see [Section 17.4.8, “Switching Sources During Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-switch).

If you are using statement-based replication and temporary tables, it is possible for a **CHANGE MASTER TO** statement following a [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement to leave behind temporary tables on the replica. A warning ([**ER\_WARN\_OPEN\_TEMP\_TABLES\_MUST\_BE\_ZERO**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_warn_open_temp_tables_must_be_zero)) is now issued whenever this occurs. You can avoid this in such cases by making sure that the value of the [**Slave\_open\_temp\_tables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Slave_open_temp_tables) system status variable is equal to 0 prior to executing such a **CHANGE MASTER TO** statement.

[**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) is useful for setting up a replica when you have the snapshot of the source and have recorded the source's binary log coordinates corresponding to the time of the snapshot. After loading the snapshot into the replica to synchronize it with the source, you can run **CHANGE MASTER TO MASTER\_LOG\_FILE='*log\_name*', MASTER\_LOG\_POS=*log\_pos*** on the replica to specify the coordinates at which the replica should begin reading the source's binary log.

The following example changes the source server the replica uses and establishes the source's binary log coordinates from which the replica begins reading:

CHANGE MASTER TO

MASTER\_HOST='source2.example.com',

MASTER\_USER='replication',

MASTER\_PASSWORD='***password***',

MASTER\_PORT=3306,

MASTER\_LOG\_FILE='source2-bin.001',

MASTER\_LOG\_POS=4,

MASTER\_CONNECT\_RETRY=10;

The next example shows an operation that is less frequently employed. It is used when the replica has relay log files that you want it to execute again for some reason. To do this, the source need not be reachable. You need only use [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) and start the SQL thread ([**START REPLICA | SLAVE SQL\_THREAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica)):

CHANGE MASTER TO

RELAY\_LOG\_FILE='replica-relay-bin.006',

RELAY\_LOG\_POS=4025;

Options that you do not specify on a [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement retain their value, except as indicated in the following discussion. Thus, in most cases, there is no need to specify options that do not change.

**ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**

Makes the replication channel assign a GTID to replicated transactions that do not have one, enabling replication from a source that does not use GTID-based replication, to a replica that does. For a multi-source replica, you can have a mix of channels that use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**, and channels that do not. The default is **OFF**, meaning that the feature is not used.

**LOCAL** assigns a GTID including the replica's own UUID (the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) setting). ***uuid*** assigns a GTID including the specified UUID, such as the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) setting for the replication source server. Using a nonlocal UUID lets you differentiate between transactions that originated on the replica and transactions that originated on the source, and for a multi-source replica, between transactions that originated on different sources. The UUID you choose only has significance for the replica's own use. If any of the transactions sent by the source do have a GTID already, that GTID is retained.

Channels specific to Group Replication cannot use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**, but an asynchronous replication channel for another source on a server instance that is a Group Replication group member can do so. In that case, do not specify the Group Replication group name as the UUID for creating the GTIDs.

To set **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** to **LOCAL** or ***uuid***, the replica must have [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) set, and this cannot be changed afterwards. This option is for use with a source that has binary log file position based replication, so **MASTER\_AUTO\_POSITION=1** cannot be set for the channel. Both the replication SQL thread and the replication I/O thread must be stopped before setting this option.

**Important**

A replica set up with **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** on any channel cannot be promoted to replace the replication source server in the event that a failover is required, and a backup taken from the replica cannot be used to restore the replication source server. The same restriction applies to replacing or restoring other replicas that use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** on any channel.

For further restrictions and information, see [Section 17.1.3.6, “Replication From a Source Without GTIDs to a Replica With GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-assign-anon).

**GET\_MASTER\_PUBLIC\_KEY**

Enables RSA key pair-based password exchange by requesting the public key from the source. This option applies to replicas that authenticate with the **caching\_sha2\_password** authentication plugin. For connections by accounts that authenticate using this plugin, the source does not send the public key unless requested, so it must be requested or specified in the client. If **MASTER\_PUBLIC\_KEY\_PATH** is given and specifies a valid public key file, it takes precedence over **GET\_MASTER\_PUBLIC\_KEY**. If you are using a replication user account that authenticates with the **caching\_sha2\_password** plugin (which is the default from MySQL 8.0), and you are not using a secure connection, you must specify either this option or the **MASTER\_PUBLIC\_KEY\_PATH** option to provide the RSA public key to the replica.

**IGNORE\_SERVER\_IDS**

Makes the replica ignore events originating from the specified servers. The option takes a comma-separated list of 0 or more server IDs. Log rotation and deletion events from the servers are not ignored, and are recorded in the relay log.

In circular replication, the originating server normally acts as the terminator of its own events, so that they are not applied more than once. Thus, this option is useful in circular replication when one of the servers in the circle is removed. Suppose that you have a circular replication setup with 4 servers, having server IDs 1, 2, 3, and 4, and server 3 fails. When bridging the gap by starting replication from server 2 to server 4, you can include **IGNORE\_SERVER\_IDS = (3)** in the [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement that you issue on server 4 to tell it to use server 2 as its source instead of server 3. Doing so causes it to ignore and not to propagate any statements that originated with the server that is no longer in use.

If **IGNORE\_SERVER\_IDS** contains the server's own ID and the server was started with the [--replicate-same-server-id](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-same-server-id) option enabled, an error results.

**Note**

When global transaction identifiers (GTIDs) are used for replication, transactions that have already been applied are automatically ignored, so the **IGNORE\_SERVER\_IDS** function is not required and is deprecated. If [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is set for the server, a deprecation warning is issued if you include the **IGNORE\_SERVER\_IDS** option in a [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement.

The source metadata repository and the output of [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) provide the list of servers that are currently ignored. For more information, see [Section 17.2.4.2, “Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-status), and [Section 13.7.7.35, “SHOW REPLICA | SLAVE STATUS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status).

If a [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement is issued without any **IGNORE\_SERVER\_IDS** option, any existing list is preserved. To clear the list of ignored servers, it is necessary to use the option with an empty list:

CHANGE MASTER TO IGNORE\_SERVER\_IDS = ();

[**RESET REPLICA | SLAVE ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) clears **IGNORE\_SERVER\_IDS**.

**Note**

A deprecation warning is issued if **SET GTID\_MODE=ON** is issued when any channel has existing server IDs set with **IGNORE\_SERVER\_IDS**. Before starting GTID-based replication, check for and clear all ignored server ID lists on the servers involved. The [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement displays the list of ignored IDs, if there is one. If you do receive the deprecation warning, you can still clear a list after [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is set by issuing a [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement containing the **IGNORE\_SERVER\_IDS** option with an empty list.

**MASTER\_AUTO\_POSITION**

Makes the replica attempt to connect to the source using the auto-positioning feature of GTID-based replication, rather than a binary log file based position. This option can be used with **CHANGE MASTER TO** only if both the replication SQL thread and replication I/O thread are stopped.

Both the replica and the source must have GTIDs enabled ([**GTID\_MODE=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode), **ON\_PERMISSIVE,** or **OFF\_PERMISSIVE** on the replica, and [**GTID\_MODE=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) on the source). **MASTER\_LOG\_FILE**, **MASTER\_LOG\_POS**, **RELAY\_LOG\_FILE**, and **RELAY\_LOG\_POS** cannot be specified together with **MASTER\_AUTO\_POSITION = 1**. If multi-source replication is enabled on the replica, you need to set the **MASTER\_AUTO\_POSITION = 1** option for each applicable replication channel.

With **MASTER\_AUTO\_POSITION = 1** set, in the initial connection handshake, the replica sends a GTID set containing the transactions that it has already received, committed, or both. The source responds by sending all transactions recorded in its binary log whose GTID is not included in the GTID set sent by the replica. This exchange ensures that the source only sends the transactions with a GTID that the replica has not already recorded or committed. If the replica receives transactions from more than one source, as in the case of a diamond topology, the auto-skip function ensures that the transactions are not applied twice. For details of how the GTID set sent by the replica is computed, see [Section 17.1.3.3, “GTID Auto-Positioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning).

If any of the transactions that should be sent by the source have been purged from the source's binary log, or added to the set of GTIDs in the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variable by another method, the source sends the error **ER\_MASTER\_HAS\_PURGED\_REQUIRED\_GTIDS** to the replica, and replication does not start. The GTIDs of the missing purged transactions are identified and listed in the source's error log in the warning message **ER\_FOUND\_MISSING\_GTIDS**. Also, if during the exchange of transactions it is found that the replica has recorded or committed transactions with the source's UUID in the GTID, but the source itself has not committed them, the source sends the error **ER\_SLAVE\_HAS\_MORE\_GTIDS\_THAN\_MASTER** to the replica and replication does not start. For information on how to handle these situations, see [Section 17.1.3.3, “GTID Auto-Positioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning).

You can see whether replication is running with GTID auto-positioning enabled by checking the Performance Schema [**replication\_connection\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-status-table) table or the output of [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). Disabling the **MASTER\_AUTO\_POSITION** option again makes the replica revert to file-based replication.

**MASTER\_BIND**

Determines which of the replica's network interfaces is chosen for connecting to the source, for use on replicas that have multiple network interfaces. The address configured with this option, if any, can be seen in the **Master\_Bind** column of the output from [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). In the source metadata repository table **mysql.slave\_master\_info**, the value can be seen as the **Master\_bind** column. The ability to bind a replica to a specific network interface is also supported by NDB Cluster.

**MASTER\_COMPRESSION\_ALGORITHMS**

Specifies the permitted compression algorithms for connections to the replication source server. The available algorithms are the same as for the [**protocol\_compression\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_protocol_compression_algorithms) system variable. The default value is **uncompressed**. **MASTER\_COMPRESSION\_ALGORITHMS** is available as of MySQL 8.0.18.

The value of **MASTER\_COMPRESSION\_ALGORITHMS** applies only if the [**slave\_compressed\_protocol**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_compressed_protocol) system variable is disabled. If [**slave\_compressed\_protocol**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_compressed_protocol) is enabled, it takes precedence over **MASTER\_COMPRESSION\_ALGORITHMS** and connections to the source use **zlib** compression if both source and replica support that algorithm. For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

Binary log transaction compression (available as of MySQL 8.0.20), which is activated by the [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression) system variable, can also be used to save bandwidth. If you do this in combination with connection compression, connection compression has less opportunity to act on the data, but can still compress headers and those events and transaction payloads that are uncompressed. For more information on binary log transaction compression, see [Section 5.4.4.5, “Binary Log Transaction Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-transaction-compression).

**MASTER\_CONNECT\_RETRY**

Specifies the interval between the reconnection attempts that the replica makes after the connection to the source times out. The attempts are limited by the **MASTER\_RETRY\_COUNT** option. If both the default settings are used, the replica waits 60 seconds between reconnection attempts (**MASTER\_CONNECT\_RETRY=60**), and keeps attempting to reconnect at this rate for 60 days (**MASTER\_RETRY\_COUNT=86400**). These values are recorded in the source metadata repository and shown in the [**replication\_connection\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-configuration-table) Performance Schema table.

**MASTER\_DELAY**

Specifies how many seconds behind the source the replica must lag. An event received from the source is not executed until at least ***interval*** seconds later than its execution on the source. The default is 0. An error occurs if ***interval*** is not a nonnegative integer in the range from 0 to 231−1. For more information, see [Section 17.4.11, “Delayed Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-delayed). A **CHANGE MASTER TO** statement employing the **MASTER\_DELAY** option can be executed on a running replica when the replication SQL thread is stopped.

**MASTER\_HEARTBEAT\_PERIOD**

Controls the heartbeat interval, which stops the connection timeout occurring in the absence of data if the connection is still good. A heartbeat signal is sent to the replica after that number of seconds, and the waiting period is reset whenever the source's binary log is updated with an event. Heartbeats are therefore sent by the source only if there are no unsent events in the binary log file for a period longer than this.

The heartbeat interval ***interval*** is a decimal value having the range 0 to 4294967 seconds and a resolution in milliseconds; the smallest nonzero value is 0.001. Setting ***interval*** to 0 disables heartbeats altogether. The heartbeat interval defaults to half the value of the [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) system variable. It is recorded in the source metadata repository and shown in the [**replication\_connection\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-configuration-table) Performance Schema table. Issuing [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) resets the heartbeat interval to the default value.

The [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) system variable specifies the number of seconds that the replica waits for either more data or a heartbeat signal from the source, before the replica considers the connection broken, aborts the read, and tries to reconnect. The default value is 60 seconds (one minute). Note that a change to the value or default setting of [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) does not automatically change the heartbeat interval, whether that has been set explicitly or is using a previously calculated default. A warning is issued if you set **@@GLOBAL.slave\_net\_timeout** to a value less than that of the current heartbeat interval. If [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) is changed, you must also issue [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) to adjust the heartbeat interval to an appropriate value so that the heartbeat signal occurs before the connection timeout. If you do not do this, the heartbeat signal has no effect, and if no data is received from the source, the replica can make repeated reconnection attempts, creating zombie dump threads.

**MASTER\_HOST**

The host name or IP address of the replication source server. The replica uses this to connect to the source.

If you specify **MASTER\_HOST** or **MASTER\_PORT**, the replica assumes that the source server is different from before (even if the option value is the same as its current value.) In this case, the old values for the source's binary log file name and position are considered no longer applicable, so if you do not specify **MASTER\_LOG\_FILE** and **MASTER\_LOG\_POS** in the statement, **MASTER\_LOG\_FILE=''** and **MASTER\_LOG\_POS=4** are silently appended to it.

Setting **MASTER\_HOST=''** (that is, setting its value explicitly to an empty string) is not the same as not setting **MASTER\_HOST** at all. Trying to set **MASTER\_HOST** to an empty string fails with an error.

**MASTER\_LOG\_FILE, MASTER\_LOG\_POS**

The binary log file name, and the location in that file, at which the replication I/O thread begins reading from the source's binary log the next time the thread starts. Specify these options if you are using binary log file position based replication. **MASTER\_LOG\_FILE** must include the numeric suffix of a specific binary log file that is available on the source server, for example, **MASTER\_LOG\_FILE='binlog.000145'**. **MASTER\_LOG\_POS** is the numeric position for the replica to start reading in that file. **MASTER\_LOG\_POS=4** represents the start of the events in a binary log file.

If you specify either of **MASTER\_LOG\_FILE** or **MASTER\_LOG\_POS**, you cannot specify **MASTER\_AUTO\_POSITION = 1**, which is for GTID-based replication.

If neither of **MASTER\_LOG\_FILE** or **MASTER\_LOG\_POS** is specified, the replica uses the last coordinates of the replication SQL thread before [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) was issued. This ensures that there is no discontinuity in replication, even if the replication SQL thread was late compared to the replication I/O thread.

**MASTER\_PASSWORD**

The password for the replication user account to use for connecting to the replication source server. If you specify **MASTER\_PASSWORD**, **MASTER\_USER** is also required.

The password used for a replication user account in a **CHANGE MASTER TO** statement is limited to 32 characters in length. Trying to use a password of more than 32 characters causes **CHANGE MASTER TO** to fail.

**MASTER\_PORT**

The TCP/IP port number that the replica uses to connect to the replication source server.

**Note**

Replication cannot use Unix socket files. You must be able to connect to the replication source server using TCP/IP.

If you specify **MASTER\_HOST** or **MASTER\_PORT**, the replica assumes that the source server is different from before (even if the option value is the same as its current value.) In this case, the old values for the source's binary log file name and position are considered no longer applicable, so if you do not specify **MASTER\_LOG\_FILE** and **MASTER\_LOG\_POS** in the statement, **MASTER\_LOG\_FILE=''** and **MASTER\_LOG\_POS=4** are silently appended to it.

**MASTER\_PUBLIC\_KEY\_PATH**

Enables RSA key pair-based password exchange by providing the path name to a file containing a replica-side copy of the public key required by the source. The file must be in PEM format. This option applies to replicas that authenticate with the **sha256\_password** or **caching\_sha2\_password** authentication plugin. (For **sha256\_password**, **MASTER\_PUBLIC\_KEY\_PATH** can be used only if MySQL was built using OpenSSL.) If you are using a replication user account that authenticates with the **caching\_sha2\_password** plugin (which is the default from MySQL 8.0), and you are not using a secure connection, you must specify either this option or the **GET\_MASTER\_PUBLIC\_KEY=1** option to provide the RSA public key to the replica.

**MASTER\_RETRY\_COUNT**

Sets the maximum number of reconnection attempts that the replica makes after the connection to the source times out, as determined by the [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) system variable. If the replica does need to reconnect, the first retry occurs immediately after the timeout. The interval between the attempts is specified by the **MASTER\_CONNECT\_RETRY** option. If both the default settings are used, the replica waits 60 seconds between reconnection attempts (**MASTER\_CONNECT\_RETRY=60**), and keeps attempting to reconnect at this rate for 60 days (**MASTER\_RETRY\_COUNT=86400**). These values are recorded in the source metadata repository and shown in the [**replication\_connection\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-configuration-table) Performance Schema table. **MASTER\_RETRY\_COUNT** supersedes the [--master-retry-count](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_master-retry-count) server startup option.

**MASTER\_SSL\_*xxx*, MASTER\_TLS\_*xxx***

Specify how the replica uses encryption and ciphers to secure the replication connection. These options can be changed even on replicas that are compiled without SSL support. They are saved to the source metadata repository, but are ignored if the replica does not have SSL support enabled. The **MASTER\_SSL\_*xxx*** and **MASTER\_TLS\_*xxx*** options perform the same functions as the **--ssl-*xxx*** and **--tls-*xxx*** client options described in [Command Options for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#encrypted-connection-options). The correspondence between the two sets of options, and the use of the **MASTER\_SSL\_*xxx*** and **MASTER\_TLS\_*xxx*** options to set up a secure connection, is explained in [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections).

**MASTER\_USER**

The user name for the replication user account to use for connecting to the replication source server.

**Important**

To connect to the source using a replication user account that authenticates with the **caching\_sha2\_password** plugin, you must either set up a secure connection as described in [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections), or enable the unencrypted connection to support password exchange using an RSA key pair. The **caching\_sha2\_password** authentication plugin is the default for new users created from MySQL 8.0 (for details, see [Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#caching-sha2-pluggable-authentication)). If the user account that you create or use for replication uses this authentication plugin, and you are not using a secure connection, you must enable RSA key pair-based password exchange for a successful connection. You can do this using either the **MASTER\_PUBLIC\_KEY\_PATH** option or the **GET\_MASTER\_PUBLIC\_KEY=1** option for this statement.

It is possible to set an empty user name by specifying **MASTER\_USER=''**, but the replication channel cannot be started with an empty user name. In releases before MySQL 8.0.21, only set an empty **MASTER\_USER** user name if you need to clear previously used credentials from the replication metadata repositories for security purposes. Do not use the channel afterwards, due to a bug in these releases that can substitute a default user name if an empty user name is read from the repositories (for example, during an automatic restart of a Group Replication channel). From MySQL 8.0.21, it is valid to set an empty **MASTER\_USER** user name and use the channel afterwards if you always provide user credentials using the [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement or [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement that starts the replication channel. This approach means that the replication channel always needs operator intervention to restart, but the user credentials are not recorded in the replication metadata repositories.

The text of a running [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, including values for **MASTER\_USER** and **MASTER\_PASSWORD**, can be seen in the output of a concurrent [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) statement. (The complete text of a [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement is also visible to [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist).)

**MASTER\_ZSTD\_COMPRESSION\_LEVEL**

The compression level to use for connections to the replication source server that use the **zstd** compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default **zstd** compression level is 3. The compression level setting has no effect on connections that do not use **zstd** compression. **MASTER\_ZSTD\_COMPRESSION\_LEVEL** is available as of MySQL 8.0.18. For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

**NETWORK\_NAMESPACE**

The network namespace to use for TCP/IP connections to the replication source server. If this option is omitted, connections from the replica use the default (global) namespace. On platforms that do not implement network namespace support, failure occurs when the replica attempts to connect to the source. For information about network namespaces, see [Section 5.1.14, “Network Namespace Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#network-namespace-support). **NETWORK\_NAMESPACE** is available as of MySQL 8.0.22.

**PRIVILEGE\_CHECKS\_USER**

Names a user account that supplies a security context for the specified channel. **NULL**, which is the default, means no security context is used. **PRIVILEGE\_CHECKS\_USER** is available as of MySQL 8.0.18.

The user name and host name for the user account must follow the syntax described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names), and the user must not be an anonymous user (with a blank user name) or the **CURRENT\_USER**. The account must have the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege, plus the required privileges to execute the transactions replicated on the channel. For details of the privileges required by the account, see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks). When you restart the replication channel, the privilege checks are applied from that point on. If you do not specify a channel and no other channels exist, the statement is applied to the default channel.

The use of row-based binary logging is strongly recommended when **PRIVILEGE\_CHECKS\_USER** is set, and you can set **REQUIRE\_ROW\_FORMAT** to enforce this. For example, to start privilege checks on the channel **channel\_1** on a running replica, issue the following statements:

mysql> STOP REPLICA | SLAVE FOR CHANNEL 'channel\_1';

mysql> CHANGE MASTER TO

PRIVILEGE\_CHECKS\_USER = '***priv\_repl***'@'***%.example.com***',

REQUIRE\_ROW\_FORMAT = 1,

FOR CHANNEL 'channel\_1';

mysql> START REPLICA | SLAVE FOR CHANNEL 'channel\_1';

**RELAY\_LOG\_FILE, RELAY\_LOG\_POS**

The relay log file name, and the location in that file, at which the replication SQL thread begins reading from the replica's relay log the next time the thread starts. **RELAY\_LOG\_FILE** can use either an absolute or relative path, and uses the same base name as **MASTER\_LOG\_FILE**.

A **CHANGE MASTER TO** statement using **RELAY\_LOG\_FILE**, **RELAY\_LOG\_POS**, or both options can be executed on a running replica when the replication SQL thread is stopped. Relay logs are preserved if at least one of the replication SQL thread and the replication I/O thread is running. If both threads are stopped, all relay log files are deleted unless at least one of **RELAY\_LOG\_FILE** or **RELAY\_LOG\_POS** is specified. For the Group Replication applier channel (**group\_replication\_applier**), which only has an SQL thread and no I/O thread, this is the case if the SQL thread is stopped, but with that channel you cannot use the **RELAY\_LOG\_FILE** and **RELAY\_LOG\_POS** options.

**REQUIRE\_ROW\_FORMAT**

Permits only row-based replication events to be processed by the replication channel. This option prevents the replication applier from taking actions such as creating temporary tables and executing **LOAD DATA INFILE** requests, which increases the security of the channel. Group Replication channels are automatically created with **REQUIRE\_ROW\_FORMAT** set, and you cannot change the option for those channels. For more information, see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks). **REQUIRE\_ROW\_FORMAT** is available as of MySQL 8.0.19.

**REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK**

Enables a replica to select its own policy for primary key checks. When the option is set to **ON** for a replication channel, the replica always uses the value **ON** for the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable in replication operations, requiring a primary key. When the option is set to **OFF**, the replica always uses the value **OFF** for the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable in replication operations, so that a primary key is never required, even if the source required one. When the **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** option is set to **STREAM**, which is the default, the replica uses whatever value is replicated from the source for each transaction. **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** is available as of MySQL 8.0.20.

For multisource replication, setting **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** to **ON** or **OFF** enables a replica to normalize behavior across the replication channels for different sources, and keep a consistent setting for the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable. Using **ON** safeguards against the accidental loss of primary keys when multiple sources update the same set of tables. Using **OFF** allows sources that can manipulate primary keys to work alongside sources that cannot.

When **PRIVILEGE\_CHECKS\_USER** is set, setting **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** to **ON** or **OFF** means that the user account does not need session administration level privileges to set restricted session variables, which are required to change the value of [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) to match the source's setting for each transaction. For more information, see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks).

**SOURCE\_CONNECTION\_AUTO\_FAILOVER**

Activates the asynchronous connection failover mechanism for a replication channel if one or more alternative replication source servers are available (so when there are multiple MySQL servers or groups of servers that share the replicated data). **SOURCE\_CONNECTION\_AUTO\_FAILOVER** is available as of MySQL 8.0.22. The asynchronous connection failover mechanism takes over after the reconnection attempts controlled by **MASTER\_CONNECT\_RETRY** and **MASTER\_RETRY\_COUNT** are exhausted. It reconnects the replica to an alternative source chosen from a specified source list, which you manage using the [**asynchronous\_connection\_failover\_add\_source**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-source) and [**asynchronous\_connection\_failover\_delete\_source**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-source) UDFs. To add and remove managed groups of servers, use the [**asynchronous\_connection\_failover\_add\_managed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-managed) and [**asynchronous\_connection\_failover\_delete\_managed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-managed) UDFs instead. For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

**Important**

You can only set **SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1** when GTID auto-positioning is in use (**MASTER\_AUTO\_POSITION = 1**).

When you set **SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1**, set **MASTER\_RETRY\_COUNT** and **MASTER\_CONNECT\_RETRY** to minimal numbers that just allow a few retry attempts with the same source in a short time, in case the connection failure is caused by a transient network outage. Otherwise the asynchronous connection failover mechanism cannot be activated promptly. Suitable values are **MASTER\_RETRY\_COUNT=3** and **MASTER\_CONNECT\_RETRY=10**, which make the replica retry the connection 3 times with 10-second intervals between.

When you set **SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1**, the replication metadata repositories must contain the credentials for a replication user account that can be used to connect to all the servers on the source list for the replication channel. These credentials can be set using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement with the **MASTER\_USER** and **MASTER\_PASSWORD** options. For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

The following table shows the maximum permissible length for the string-valued options.

| **Option** | **Maximum Length** |
| --- | --- |
| **MASTER\_HOST** | 255 (60 prior to MySQL 8.0.17) |
| **MASTER\_USER** | 96 |
| **MASTER\_PASSWORD** | 32 |
| **MASTER\_LOG\_FILE** | 511 |
| **RELAY\_LOG\_FILE** | 511 |
| **MASTER\_SSL\_CA** | 511 |
| **MASTER\_SSL\_CAPATH** | 511 |
| **MASTER\_SSL\_CERT** | 511 |
| **MASTER\_SSL\_CRL** | 511 |
| **MASTER\_SSL\_CRLPATH** | 511 |
| **MASTER\_SSL\_KEY** | 511 |
| **MASTER\_SSL\_CIPHER** | 511 |
| **MASTER\_TLS\_VERSION** | 511 |
| **MASTER\_TLS\_CIPHERSUITES** | 4000 |
| **MASTER\_PUBLIC\_KEY\_PATH** | 511 |
| **MASTER\_COMPRESSION\_ALGORITHMS** | 99 |
| **NETWORK\_NAMESPACE** | 64 |

#### 13.4.2.2 CHANGE REPLICATION FILTER Statement

CHANGE REPLICATION FILTER ***filter***[, ***filter***]

[, ...] [FOR CHANNEL ***channel***]

***filter***: {

REPLICATE\_DO\_DB = (***db\_list***)

| REPLICATE\_IGNORE\_DB = (***db\_list***)

| REPLICATE\_DO\_TABLE = (***tbl\_list***)

| REPLICATE\_IGNORE\_TABLE = (***tbl\_list***)

| REPLICATE\_WILD\_DO\_TABLE = (***wild\_tbl\_list***)

| REPLICATE\_WILD\_IGNORE\_TABLE = (***wild\_tbl\_list***)

| REPLICATE\_REWRITE\_DB = (***db\_pair\_list***)

}

***db\_list***:

***db\_name***[, ***db\_name***][, ...]

***tbl\_list***:

***db\_name.table\_name***[, ***db\_name.table\_name***][, ...]

***wild\_tbl\_list***:

'***db\_pattern.table\_pattern***'[, '***db\_pattern.table\_pattern***'][, ...]

***db\_pair\_list***:

(***db\_pair***)[, (***db\_pair***)][, ...]

***db\_pair***:

***from\_db***, ***to\_db***

**CHANGE REPLICATION FILTER** sets one or more replication filtering rules on the replica in the same way as starting the replica [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) with replication filtering options such as [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) or [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table). Unlike the case with the server options, this statement does not require restarting the server to take effect, only that the replication SQL thread be stopped using [**STOP REPLICA | SLAVE SQL\_THREAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) first (and restarted with [**START REPLICA | SLAVE SQL\_THREAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) afterwards). [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) requires the [**REPLICATION\_SLAVE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege). Use the **FOR CHANNEL *channel*** clause to make a replication filter specific to a replication channel, for example on a multi-source replica. Filters applied without a specific **FOR CHANNEL** clause are considered global filters, meaning that they are applied to all replication channels.

**Note**

Global replication filters cannot be set on a MySQL server instance that is configured for Group Replication, because filtering transactions on some servers would make the group unable to reach agreement on a consistent state. Channel specific replication filters can be set on replication channels that are not directly involved with Group Replication, such as where a group member also acts as a replica to a source that is outside the group. They cannot be set on the **group\_replication\_applier** or **group\_replication\_recovery** channels.

The following list shows the **CHANGE REPLICATION FILTER** options and how they relate to --replicate-\* server options:

**REPLICATE\_DO\_DB**: Include updates based on database name. Equivalent to [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db).

**REPLICATE\_IGNORE\_DB**: Exclude updates based on database name. Equivalent to [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db).

**REPLICATE\_DO\_TABLE**: Include updates based on table name. Equivalent to [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table).

**REPLICATE\_IGNORE\_TABLE**: Exclude updates based on table name. Equivalent to [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table).

**REPLICATE\_WILD\_DO\_TABLE**: Include updates based on wildcard pattern matching table name. Equivalent to [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table).

**REPLICATE\_WILD\_IGNORE\_TABLE**: Exclude updates based on wildcard pattern matching table name. Equivalent to [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table).

**REPLICATE\_REWRITE\_DB**: Perform updates on replica after substituting new name on replica for specified database on source. Equivalent to [--replicate-rewrite-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-rewrite-db).

The precise effects of **REPLICATE\_DO\_DB** and **REPLICATE\_IGNORE\_DB** filters are dependent on whether statement-based or row-based replication is in effect. See [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules), for more information.

Multiple replication filtering rules can be created in a single **CHANGE REPLICATION FILTER** statement by separating the rules with commas, as shown here:

CHANGE REPLICATION FILTER

REPLICATE\_DO\_DB = (d1), REPLICATE\_IGNORE\_DB = (d2);

Issuing the statement just shown is equivalent to starting the replica [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) with the options [--replicate-do-db=d1](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) [--replicate-ignore-db=d2](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db).

On a multi-source replica, which uses multiple replication channels to process transaction from different sources, use the **FOR CHANNEL *channel*** clause to set a replication filter on a replication channel:

CHANGE REPLICATION FILTER REPLICATE\_DO\_DB = (d1) FOR CHANNEL channel\_1;

This enables you to create a channel specific replication filter to filter out selected data from a source. When a **FOR CHANNEL** clause is provided, the replication filter statement acts on that replication channel, removing any existing replication filter which has the same filter type as the specified replication filters, and replacing them with the specified filter. Filter types not explicitly listed in the statement are not modified. If issued against a replication channel which is not configured, the statement fails with an ER\_SLAVE\_CONFIGURATION error. If issued against Group Replication channels, the statement fails with an ER\_SLAVE\_CHANNEL\_OPERATION\_NOT\_ALLOWED error.

On a replica with multiple replication channels configured, issuing [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) with no **FOR CHANNEL** clause configures the replication filter for every configured replication channel, and for the global replication filters. For every filter type, if the filter type is listed in the statement, then any existing filter rules of that type are replaced by the filter rules specified in the most recently issued statement, otherwise the old value of the filter type is retained. For more information see [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters).

If the same filtering rule is specified multiple times, only the last such rule is actually used. For example, the two statements shown here have exactly the same effect, because the first **REPLICATE\_DO\_DB** rule in the first statement is ignored:

CHANGE REPLICATION FILTER

REPLICATE\_DO\_DB = (db1, db2), REPLICATE\_DO\_DB = (db3, db4);

CHANGE REPLICATION FILTER

REPLICATE\_DO\_DB = (db3, db4);

**Caution**

This behavior differs from that of the --replicate-\* filter options where specifying the same option multiple times causes the creation of multiple filter rules.

Names of tables and database not containing any special characters need not be quoted. Values used with **REPLICATION\_WILD\_TABLE** and **REPLICATION\_WILD\_IGNORE\_TABLE** are string expressions, possibly containing (special) wildcard characters, and so must be quoted. This is shown in the following example statements:

CHANGE REPLICATION FILTER

REPLICATE\_WILD\_DO\_TABLE = ('db1.old%');

CHANGE REPLICATION FILTER

REPLICATE\_WILD\_IGNORE\_TABLE = ('db1.new%', 'db2.new%');

Values used with **REPLICATE\_REWRITE\_DB** represent pairs of database names; each such value must be enclosed in parentheses. The following statement rewrites statements occurring on database **db1** on the source to database **db2** on the replica:

CHANGE REPLICATION FILTER REPLICATE\_REWRITE\_DB = ((db1, db2));

The statement just shown contains two sets of parentheses, one enclosing the pair of database names, and the other enclosing the entire list. This is perhaps more easily seen in the following example, which creates two **rewrite-db** rules, one rewriting database **dbA** to **dbB**, and one rewriting database **dbC** to **dbD**:

CHANGE REPLICATION FILTER

REPLICATE\_REWRITE\_DB = ((dbA, dbB), (dbC, dbD));

The [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement replaces replication filtering rules only for the filter types and replication channels affected by the statement, and leaves other rules and channels unchanged. If you want to unset all filters of a given type, set the filter's value to an explicitly empty list, as shown in this example, which removes all existing **REPLICATE\_DO\_DB** and **REPLICATE\_IGNORE\_DB** rules:

CHANGE REPLICATION FILTER

REPLICATE\_DO\_DB = (), REPLICATE\_IGNORE\_DB = ();

Setting a filter to empty in this way removes all existing rules, does not create any new ones, and does not restore any rules set at mysqld startup using --replicate-\* options on the command line or in the configuration file.

The [**RESET REPLICA | SLAVE ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) statement removes channel specific replication filters that were set on channels deleted by the statement. When the deleted channel or channels are recreated, any global replication filters specified for the replica are copied to them, and no channel specific replication filters are applied.

For more information, see [Section 17.2.5, “How Servers Evaluate Replication Filtering Rules”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules).

#### 13.4.2.3 CHANGE REPLICATION SOURCE TO Statement

CHANGE REPLICATION SOURCE TO ***option*** [, ***option***] ... [ ***channel\_option*** ]

***option***: {

SOURCE\_BIND = '***interface\_name***'

| SOURCE\_HOST = '***host\_name***'

| SOURCE\_USER = '***user\_name***'

| SOURCE\_PASSWORD = '***password***'

| SOURCE\_PORT = ***port\_num***

| PRIVILEGE\_CHECKS\_USER = {'***account***' | NULL}

| REQUIRE\_ROW\_FORMAT = {0|1}

| REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK = {STREAM | ON | OFF}

| ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS = {OFF | LOCAL | ***uuid***}

| SOURCE\_LOG\_FILE = '***source\_log\_name***'

| SOURCE\_LOG\_POS = ***source\_log\_pos***

| SOURCE\_AUTO\_POSITION = {0|1}

| RELAY\_LOG\_FILE = '***relay\_log\_name***'

| RELAY\_LOG\_POS = ***relay\_log\_pos***

| SOURCE\_HEARTBEAT\_PERIOD = ***interval***

| SOURCE\_CONNECT\_RETRY = ***interval***

| SOURCE\_RETRY\_COUNT = ***count***

| SOURCE\_CONNECTION\_AUTO\_FAILOVER = {0|1}

| SOURCE\_DELAY = ***interval***

| SOURCE\_COMPRESSION\_ALGORITHMS = '***value***'

| SOURCE\_ZSTD\_COMPRESSION\_LEVEL = ***level***

| SOURCE\_SSL = {0|1}

| SOURCE\_SSL\_CA = '***ca\_file\_name***'

| SOURCE\_SSL\_CAPATH = '***ca\_directory\_name***'

| SOURCE\_SSL\_CERT = '***cert\_file\_name***'

| SOURCE\_SSL\_CRL = '***crl\_file\_name***'

| SOURCE\_SSL\_CRLPATH = '***crl\_directory\_name***'

| SOURCE\_SSL\_KEY = '***key\_file\_name***'

| SOURCE\_SSL\_CIPHER = '***cipher\_list***'

| SOURCE\_SSL\_VERIFY\_SERVER\_CERT = {0|1}

| SOURCE\_TLS\_VERSION = '***protocol\_list***'

| SOURCE\_TLS\_CIPHERSUITES = '***ciphersuite\_list***'

| SOURCE\_PUBLIC\_KEY\_PATH = '***key\_file\_name***'

| GET\_SOURCE\_PUBLIC\_KEY = {0|1}

| NETWORK\_NAMESPACE = '***namespace***'

| IGNORE\_SERVER\_IDS = (***server\_id\_list***)

}

***channel\_option***:

FOR CHANNEL ***channel***

***server\_id\_list***:

[***server\_id*** [, ***server\_id***] ... ]

[**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) changes the parameters that the replica server uses for connecting to the source and for reading data from the source. It also updates the contents of the replication metadata repositories (see [Section 17.2.4, “Relay Log and Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs)). From MySQL 8.0.23, use [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) in place of [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to), which is deprecated from that release. In releases before MySQL 8.0.23, use [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to).

You can issue [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statements on a running replica without first stopping it, depending on the states of the replication SQL thread and replication I/O thread. The rules governing such use are provided later in this section. [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) requires the [**REPLICATION\_SLAVE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

When using a multithreaded replica (in other words [**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) is greater than 0), stopping the replica can cause “gaps” in the sequence of transactions that have been executed from the relay log, regardless of whether the replica was stopped intentionally or otherwise. When such gaps exist, issuing [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) fails. The solution in this situation is to issue [**START REPLICA UNTIL SQL\_AFTER\_MTS\_GAPS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) which ensures that the gaps are closed.

The optional **FOR CHANNEL *channel*** clause enables you to name which replication channel the statement applies to. Providing a **FOR CHANNEL *channel*** clause applies the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement to a specific replication channel, and is used to add a new channel or modify an existing channel. For example, to add a new channel called **channel2**:

CHANGE REPLICATION SOURCE TO SOURCE\_HOST=host1, SOURCE\_PORT=3002 FOR CHANNEL 'channel2'

If no clause is named and no extra channels exist, the statement applies to the default channel.

When using multiple replication channels, if a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement does not name a channel using a **FOR CHANNEL *channel*** clause, an error occurs. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels) for more information.

Values used for **SOURCE\_HOST** and other [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) options are checked for linefeed (**\n** or **0x0A**) characters. The presence of such characters in these values causes the statement to fail with [**ER\_MASTER\_INFO**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_master_info).

Invoking [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) causes the previous values for **SOURCE\_HOST**, **SOURCE\_PORT**, **SOURCE\_LOG\_FILE**, and **SOURCE\_LOG\_POS** to be written to the error log, along with other information about the replica's state prior to execution.

[**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) causes an implicit commit of an ongoing transaction. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

For some of the options of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement, you must issue a [**STOP REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement prior to issuing a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (and a [**START REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement afterwards). Sometimes, you only need to stop the replication SQL thread or the replication I/O thread, not both:

When the SQL thread is stopped, you can execute [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) using any combination that is otherwise allowed of **RELAY\_LOG\_FILE**, **RELAY\_LOG\_POS**, and **SOURCE\_DELAY** options, even if the replication I/O thread is running. No other options may be used with this statement when the I/O thread is running.

When the I/O thread is stopped, you can execute [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) using any of the options for this statement (in any allowed combination) except **RELAY\_LOG\_FILE**, **RELAY\_LOG\_POS**, **SOURCE\_DELAY**, or **SOURCE\_AUTO\_POSITION = 1** even when the SQL thread is running.

Both the SQL thread and the I/O thread must be stopped before issuing a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement that employs **SOURCE\_AUTO\_POSITION = 1** or **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**.

You can check the current state of the replication SQL thread and replication I/O thread using [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). Note that the Group Replication applier channel (**group\_replication\_applier**) has no I/O thread, only an SQL thread.

For more information, see [Section 17.4.8, “Switching Sources During Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-switch).

If you are using statement-based replication and temporary tables, it is possible for a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement following a [**STOP REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement to leave behind temporary tables on the replica. A warning ([**ER\_WARN\_OPEN\_TEMP\_TABLES\_MUST\_BE\_ZERO**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_warn_open_temp_tables_must_be_zero)) is now issued whenever this occurs. You can avoid this in such cases by making sure that the value of the [**Slave\_open\_temp\_tables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Slave_open_temp_tables) system status variable is equal to 0 prior to executing such a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement.

[**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) is useful for setting up a replica when you have the snapshot of the source and have recorded the source's binary log coordinates corresponding to the time of the snapshot. After loading the snapshot into the replica to synchronize it with the source, you can run **CHANGE REPLICATION SOURCE TO SOURCE\_LOG\_FILE='*log\_name*', SOURCE\_LOG\_POS=*log\_pos*** on the replica to specify the coordinates at which the replica should begin reading the source's binary log.

The following example changes the source server the replica uses and establishes the source's binary log coordinates from which the replica begins reading:

CHANGE REPLICATION SOURCE TO

SOURCE\_HOST='source2.example.com',

SOURCE\_USER='replication',

SOURCE\_PASSWORD='***password***',

SOURCE\_PORT=3306,

SOURCE\_LOG\_FILE='source2-bin.001',

SOURCE\_LOG\_POS=4,

SOURCE\_CONNECT\_RETRY=10;

The next example shows an operation that is less frequently employed. It is used when the replica has relay log files that you want it to execute again for some reason. To do this, the source need not be reachable. You need only use [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) and start the SQL thread ([**START REPLICA SQL\_THREAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica)):

CHANGE REPLICATION SOURCE TO

RELAY\_LOG\_FILE='replica-relay-bin.006',

RELAY\_LOG\_POS=4025;

Options that you do not specify on a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement retain their value, except as indicated in the following discussion. Thus, in most cases, there is no need to specify options that do not change.

**ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**

Makes the replication channel assign a GTID to replicated transactions that do not have one, enabling replication from a source that does not use GTID-based replication, to a replica that does. For a multi-source replica, you can have a mix of channels that use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**, and channels that do not. The default is **OFF**, meaning that the feature is not used.

**LOCAL** assigns a GTID including the replica's own UUID (the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) setting). ***uuid*** assigns a GTID including the specified UUID, such as the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) setting for the replication source server. Using a nonlocal UUID lets you differentiate between transactions that originated on the replica and transactions that originated on the source, and for a multi-source replica, between transactions that originated on different sources. The UUID you choose only has significance for the replica's own use. If any of the transactions sent by the source do have a GTID already, that GTID is retained.

Channels specific to Group Replication cannot use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**, but an asynchronous replication channel for another source on a server instance that is a Group Replication group member can do so. In that case, do not specify the Group Replication group name as the UUID for creating the GTIDs.

To set **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** to **LOCAL** or ***uuid***, the replica must have [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) set, and this cannot be changed afterwards. This option is for use with a source that has binary log file position based replication, so **MASTER\_AUTO\_POSITION=1** cannot be set for the channel. Both the replication SQL thread and the replication I/O thread must be stopped before setting this option.

**Important**

A replica set up with **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** on any channel cannot be promoted to replace the replication source server in the event that a failover is required, and a backup taken from the replica cannot be used to restore the replication source server. The same restriction applies to replacing or restoring other replicas that use **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS** on any channel.

For further restrictions and information, see [Section 17.1.3.6, “Replication From a Source Without GTIDs to a Replica With GTIDs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-assign-anon).

**GET\_SOURCE\_PUBLIC\_KEY**

Enables RSA key pair-based password exchange by requesting the public key from the source. This option applies to replicas that authenticate with the **caching\_sha2\_password** authentication plugin. For connections by accounts that authenticate using this plugin, the source does not send the public key unless requested, so it must be requested or specified in the client. If **SOURCE\_PUBLIC\_KEY\_PATH** is given and specifies a valid public key file, it takes precedence over **GET\_SOURCE\_PUBLIC\_KEY**. If you are using a replication user account that authenticates with the **caching\_sha2\_password** plugin (which is the default from MySQL 8.0), and you are not using a secure connection, you must specify either this option or the **SOURCE\_PUBLIC\_KEY\_PATH** option to provide the RSA public key to the replica.

**IGNORE\_SERVER\_IDS**

Makes the replica ignore events originating from the specified servers. The option takes a comma-separated list of 0 or more server IDs. Log rotation and deletion events from the servers are not ignored, and are recorded in the relay log.

In circular replication, the originating server normally acts as the terminator of its own events, so that they are not applied more than once. Thus, this option is useful in circular replication when one of the servers in the circle is removed. Suppose that you have a circular replication setup with 4 servers, having server IDs 1, 2, 3, and 4, and server 3 fails. When bridging the gap by starting replication from server 2 to server 4, you can include **IGNORE\_SERVER\_IDS = (3)** in the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement that you issue on server 4 to tell it to use server 2 as its source instead of server 3. Doing so causes it to ignore and not to propagate any statements that originated with the server that is no longer in use.

If **IGNORE\_SERVER\_IDS** contains the server's own ID and the server was started with the [--replicate-same-server-id](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-same-server-id) option enabled, an error results.

**Note**

When global transaction identifiers (GTIDs) are used for replication, transactions that have already been applied are automatically ignored, so the **IGNORE\_SERVER\_IDS** function is not required and is deprecated. If [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is set for the server, a deprecation warning is issued if you include the **IGNORE\_SERVER\_IDS** option in a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement.

The source metadata repository and the output of [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) provide the list of servers that are currently ignored. For more information, see [Section 17.2.4.2, “Replication Metadata Repositories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replica-logs-status), and [Section 13.7.7.35, “SHOW REPLICA | SLAVE STATUS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status).

If a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement is issued without any **IGNORE\_SERVER\_IDS** option, any existing list is preserved. To clear the list of ignored servers, it is necessary to use the option with an empty list:

CHANGE REPLICATION SOURCE TO IGNORE\_SERVER\_IDS = ();

[**RESET REPLICA ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) clears **IGNORE\_SERVER\_IDS**.

**Note**

A deprecation warning is issued if **SET GTID\_MODE=ON** is issued when any channel has existing server IDs set with **IGNORE\_SERVER\_IDS**. Before starting GTID-based replication, check for and clear all ignored server ID lists on the servers involved. The [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement displays the list of ignored IDs, if there is one. If you do receive the deprecation warning, you can still clear a list after [**gtid\_mode=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is set by issuing a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement containing the **IGNORE\_SERVER\_IDS** option with an empty list.

**NETWORK\_NAMESPACE**

The network namespace to use for TCP/IP connections to the replication source server. If this option is omitted, connections from the replica use the default (global) namespace. On platforms that do not implement network namespace support, failure occurs when the replica attempts to connect to the source. For information about network namespaces, see [Section 5.1.14, “Network Namespace Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#network-namespace-support). **NETWORK\_NAMESPACE** is available as of MySQL 8.0.22.

**PRIVILEGE\_CHECKS\_USER**

Names a user account that supplies a security context for the specified channel. **NULL**, which is the default, means no security context is used. **PRIVILEGE\_CHECKS\_USER** is available as of MySQL 8.0.18.

The user name and host name for the user account must follow the syntax described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names), and the user must not be an anonymous user (with a blank user name) or the **CURRENT\_USER**. The account must have the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege, plus the required privileges to execute the transactions replicated on the channel. For details of the privileges required by the account, see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks). When you restart the replication channel, the privilege checks are applied from that point on. If you do not specify a channel and no other channels exist, the statement is applied to the default channel.

The use of row-based binary logging is strongly recommended when **PRIVILEGE\_CHECKS\_USER** is set, and you can set **REQUIRE\_ROW\_FORMAT** to enforce this. For example, to start privilege checks on the channel **channel\_1** on a running replica, issue the following statements:

mysql> STOP REPLICA FOR CHANNEL 'channel\_1';

mysql> CHANGE REPLICATION SOURCE TO

PRIVILEGE\_CHECKS\_USER = '***priv\_repl***'@'***%.example.com***',

REQUIRE\_ROW\_FORMAT = 1,

FOR CHANNEL 'channel\_1';

mysql> START REPLICA FOR CHANNEL 'channel\_1';

**RELAY\_LOG\_FILE, RELAY\_LOG\_POS**

The relay log file name, and the location in that file, at which the replication SQL thread begins reading from the replica's relay log the next time the thread starts. **RELAY\_LOG\_FILE** can use either an absolute or relative path, and uses the same base name as **SOURCE\_LOG\_FILE**.

A **CHANGE REPLICATION SOURCE TO** statement using **RELAY\_LOG\_FILE**, **RELAY\_LOG\_POS**, or both options can be executed on a running replica when the replication SQL thread is stopped. Relay logs are preserved if at least one of the replication SQL thread and the replication I/O thread is running. If both threads are stopped, all relay log files are deleted unless at least one of **RELAY\_LOG\_FILE** or **RELAY\_LOG\_POS** is specified. For the Group Replication applier channel (**group\_replication\_applier**), which only has an SQL thread and no I/O thread, this is the case if the SQL thread is stopped, but with that channel you cannot use the **RELAY\_LOG\_FILE** and **RELAY\_LOG\_POS** options.

**REQUIRE\_ROW\_FORMAT**

Permits only row-based replication events to be processed by the replication channel. This option prevents the replication applier from taking actions such as creating temporary tables and executing **LOAD DATA INFILE** requests, which increases the security of the channel. Group Replication channels are automatically created with **REQUIRE\_ROW\_FORMAT** set, and you cannot change the option for those channels. For more information, see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks). **REQUIRE\_ROW\_FORMAT** is available as of MySQL 8.0.19.

**REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK**

Enables a replica to select its own policy for primary key checks. When the option is set to **ON** for a replication channel, the replica always uses the value **ON** for the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable in replication operations, requiring a primary key. When the option is set to **OFF**, the replica always uses the value **OFF** for the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable in replication operations, so that a primary key is never required, even if the source required one. When the **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** option is set to **STREAM**, which is the default, the replica uses whatever value is replicated from the source for each transaction. **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** is available as of MySQL 8.0.20.

For multisource replication, setting **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** to **ON** or **OFF** enables a replica to normalize behavior across the replication channels for different sources, and keep a consistent setting for the [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) system variable. Using **ON** safeguards against the accidental loss of primary keys when multiple sources update the same set of tables. Using **OFF** allows sources that can manipulate primary keys to work alongside sources that cannot.

When **PRIVILEGE\_CHECKS\_USER** is set, setting **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** to **ON** or **OFF** means that the user account does not need session administration level privileges to set restricted session variables, which are required to change the value of [**sql\_require\_primary\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_require_primary_key) to match the source's setting for each transaction. For more information, see [Section 17.3.3, “Replication Privilege Checks”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-privilege-checks).

**SOURCE\_AUTO\_POSITION**

Makes the replica attempt to connect to the source using the auto-positioning feature of GTID-based replication, rather than a binary log file based position. This option can be used with **CHANGE REPLICATION SOURCE TO** only if both the replication SQL thread and replication I/O thread are stopped.

Both the replica and the source must have GTIDs enabled ([**GTID\_MODE=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode), **ON\_PERMISSIVE,** or **OFF\_PERMISSIVE** on the replica, and [**GTID\_MODE=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) on the source). **SOURCE\_LOG\_FILE**, **SOURCE\_LOG\_POS**, **RELAY\_LOG\_FILE**, and **RELAY\_LOG\_POS** cannot be specified together with **SOURCE\_AUTO\_POSITION = 1**. If multi-source replication is enabled on the replica, you need to set the **SOURCE\_AUTO\_POSITION = 1** option for each applicable replication channel.

With **SOURCE\_AUTO\_POSITION = 1** set, in the initial connection handshake, the replica sends a GTID set containing the transactions that it has already received, committed, or both. The source responds by sending all transactions recorded in its binary log whose GTID is not included in the GTID set sent by the replica. This exchange ensures that the source only sends the transactions with a GTID that the replica has not already recorded or committed. If the replica receives transactions from more than one source, as in the case of a diamond topology, the auto-skip function ensures that the transactions are not applied twice. For details of how the GTID set sent by the replica is computed, see [Section 17.1.3.3, “GTID Auto-Positioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning).

If any of the transactions that should be sent by the source have been purged from the source's binary log, or added to the set of GTIDs in the [**gtid\_purged**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_purged) system variable by another method, the source sends the error **ER\_MASTER\_HAS\_PURGED\_REQUIRED\_GTIDS** to the replica, and replication does not start. The GTIDs of the missing purged transactions are identified and listed in the source's error log in the warning message **ER\_FOUND\_MISSING\_GTIDS**. Also, if during the exchange of transactions it is found that the replica has recorded or committed transactions with the source's UUID in the GTID, but the source itself has not committed them, the source sends the error **ER\_SLAVE\_HAS\_MORE\_GTIDS\_THAN\_MASTER** to the replica and replication does not start. For information on how to handle these situations, see [Section 17.1.3.3, “GTID Auto-Positioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-auto-positioning).

You can see whether replication is running with GTID auto-positioning enabled by checking the Performance Schema [**replication\_connection\_status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-status-table) table or the output of [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). Disabling the **SOURCE\_AUTO\_POSITION** option again makes the replica revert to file-based replication.

**SOURCE\_BIND**

Determines which of the replica's network interfaces is chosen for connecting to the source, for use on replicas that have multiple network interfaces. The address configured with this option, if any, can be seen in the **Source\_Bind** column of the output from [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). In the source metadata repository table **mysql.slave\_master\_info**, the value can be seen as the **Source\_bind** column. The ability to bind a replica to a specific network interface is also supported by NDB Cluster.

**SOURCE\_COMPRESSION\_ALGORITHMS**

Specifies the permitted compression algorithms for connections to the replication source server. The available algorithms are the same as for the [**protocol\_compression\_algorithms**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_protocol_compression_algorithms) system variable. The default value is **uncompressed**. **SOURCE\_COMPRESSION\_ALGORITHMS** is available as of MySQL 8.0.18.

The value of **SOURCE\_COMPRESSION\_ALGORITHMS** applies only if the [**slave\_compressed\_protocol**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_compressed_protocol) system variable is disabled. If [**slave\_compressed\_protocol**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_compressed_protocol) is enabled, it takes precedence over **SOURCE\_COMPRESSION\_ALGORITHMS** and connections to the source use **zlib** compression if both source and replica support that algorithm. For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

Binary log transaction compression (available as of MySQL 8.0.20), which is activated by the [**binlog\_transaction\_compression**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_transaction_compression) system variable, can also be used to save bandwidth. If you do this in combination with connection compression, connection compression has less opportunity to act on the data, but can still compress headers and those events and transaction payloads that are uncompressed. For more information on binary log transaction compression, see [Section 5.4.4.5, “Binary Log Transaction Compression”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#binary-log-transaction-compression).

**SOURCE\_CONNECT\_RETRY**

Specifies the interval between the reconnection attempts that the replica makes after the connection to the source times out. The attempts are limited by the **SOURCE\_RETRY\_COUNT** option. If both the default settings are used, the replica waits 60 seconds between reconnection attempts (**SOURCE\_CONNECT\_RETRY=60**), and keeps attempting to reconnect at this rate for 60 days (**SOURCE\_RETRY\_COUNT=86400**). These values are recorded in the source metadata repository and shown in the [**replication\_connection\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-configuration-table) Performance Schema table.

**SOURCE\_CONNECTION\_AUTO\_FAILOVER**

Activates the asynchronous connection failover mechanism for a replication channel if one or more alternative replication source servers are available (so when there are multiple MySQL servers or groups of servers that share the replicated data). **SOURCE\_CONNECTION\_AUTO\_FAILOVER** is available as of MySQL 8.0.22. The asynchronous connection failover mechanism takes over after the reconnection attempts controlled by **SOURCE\_CONNECT\_RETRY** and **SOURCE\_RETRY\_COUNT** are exhausted. It reconnects the replica to an alternative source chosen from a specified source list, which you manage using the [**asynchronous\_connection\_failover\_add\_source**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-source) and [**asynchronous\_connection\_failover\_delete\_source**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-source) UDFs. To add and remove managed groups of servers, use the [**asynchronous\_connection\_failover\_add\_managed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-managed) and [**asynchronous\_connection\_failover\_delete\_managed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-managed) UDFs instead. For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

**Important**

You can only set **SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1** when GTID auto-positioning is in use (**SOURCE\_AUTO\_POSITION = 1**).

When you set **SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1**, set **SOURCE\_RETRY\_COUNT** and **SOURCE\_CONNECT\_RETRY** to minimal numbers that just allow a few retry attempts with the same source, in case the connection failure is caused by a transient network outage. Otherwise the asynchronous connection failover mechanism cannot be activated promptly. Suitable values are **SOURCE\_RETRY\_COUNT=3** and **SOURCE\_CONNECT\_RETRY=10**, which make the replica retry the connection 3 times with 10-second intervals between.

When you set **SOURCE\_CONNECTION\_AUTO\_FAILOVER = 1**, the replication metadata repositories must contain the credentials for a replication user account that can be used to connect to all the servers on the source list for the replication channel. The account must also have **SELECT** permissions on the Performance Schema tables. These credentials can be set using the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement with the **SOURCE\_USER** and **SOURCE\_PASSWORD** options. For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

**SOURCE\_DELAY**

Specifies how many seconds behind the source the replica must lag. An event received from the source is not executed until at least ***interval*** seconds later than its execution on the source. The default is 0. An error occurs if ***interval*** is not a nonnegative integer in the range from 0 to 231−1. For more information, see [Section 17.4.11, “Delayed Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-delayed). A **CHANGE REPLICATION SOURCE TO** statement using the **SOURCE\_DELAY** option can be executed on a running replica when the replication SQL thread is stopped.

**SOURCE\_HEARTBEAT\_PERIOD**

Controls the heartbeat interval, which stops the connection timeout occurring in the absence of data if the connection is still good. A heartbeat signal is sent to the replica after that number of seconds, and the waiting period is reset whenever the source's binary log is updated with an event. Heartbeats are therefore sent by the source only if there are no unsent events in the binary log file for a period longer than this.

The heartbeat interval ***interval*** is a decimal value having the range 0 to 4294967 seconds and a resolution in milliseconds; the smallest nonzero value is 0.001. Setting ***interval*** to 0 disables heartbeats altogether. The heartbeat interval defaults to half the value of the [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) system variable. It is recorded in the source metadata repository and shown in the [**replication\_connection\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-configuration-table) Performance Schema table. Issuing [**RESET REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) resets the heartbeat interval to the default value.

The [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) system variable specifies the number of seconds that the replica waits for either more data or a heartbeat signal from the source, before the replica considers the connection broken, aborts the read, and tries to reconnect. The default value is 60 seconds (one minute). Note that a change to the value or default setting of [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) does not automatically change the heartbeat interval, whether that has been set explicitly or is using a previously calculated default. A warning is issued if you set **@@GLOBAL.slave\_net\_timeout** to a value less than that of the current heartbeat interval. If [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) is changed, you must also issue [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) to adjust the heartbeat interval to an appropriate value so that the heartbeat signal occurs before the connection timeout. If you do not do this, the heartbeat signal has no effect, and if no data is received from the source, the replica can make repeated reconnection attempts, creating zombie dump threads.

**SOURCE\_HOST**

The host name or IP address of the replication source server. The replica uses this to connect to the source.

If you specify **SOURCE\_HOST** or **SOURCE\_PORT**, the replica assumes that the source server is different from before (even if the option value is the same as its current value.) In this case, the old values for the source's binary log file name and position are considered no longer applicable, so if you do not specify **SOURCE\_LOG\_FILE** and **SOURCE\_LOG\_POS** in the statement, **SOURCE\_LOG\_FILE=''** and **SOURCE\_LOG\_POS=4** are silently appended to it.

Setting **SOURCE\_HOST=''** (that is, setting its value explicitly to an empty string) is not the same as not setting **SOURCE\_HOST** at all. Trying to set **SOURCE\_HOST** to an empty string fails with an error.

**SOURCE\_LOG\_FILE, SOURCE\_LOG\_POS**

The binary log file name, and the location in that file, at which the replication I/O thread begins reading from the source's binary log the next time the thread starts. Specify these options if you are using binary log file position based replication. **SOURCE\_LOG\_FILE** must include the numeric suffix of a specific binary log file that is available on the source server, for example, **SOURCE\_LOG\_FILE='binlog.000145'**. **SOURCE\_LOG\_POS** is the numeric position for the replica to start reading in that file. **SOURCE\_LOG\_POS=4** represents the start of the events in a binary log file.

If you specify either of **SOURCE\_LOG\_FILE** or **SOURCE\_LOG\_POS**, you cannot specify **SOURCE\_AUTO\_POSITION = 1**, which is for GTID-based replication.

If neither of **SOURCE\_LOG\_FILE** or **SOURCE\_LOG\_POS** is specified, the replica uses the last coordinates of the replication SQL thread before **CHANGE REPLICATION SOURCE TO** was issued. This ensures that there is no discontinuity in replication, even if the replication SQL thread was late compared to the replication I/O thread.

**SOURCE\_PASSWORD**

The password for the replication user account to use for connecting to the replication source server. If you specify **SOURCE\_PASSWORD**, **SOURCE\_USER** is also required.

The password used for a replication user account in a **CHANGE REPLICATION SOURCE TO** statement is limited to 32 characters in length. Trying to use a password of more than 32 characters causes **CHANGE REPLICATION SOURCE TO** to fail.

**SOURCE\_PORT**

The TCP/IP port number that the replica uses to connect to the replication source server.

**Note**

Replication cannot use Unix socket files. You must be able to connect to the replication source server using TCP/IP.

If you specify **SOURCE\_HOST** or **SOURCE\_PORT**, the replica assumes that the source server is different from before (even if the option value is the same as its current value.) In this case, the old values for the source's binary log file name and position are considered no longer applicable, so if you do not specify **SOURCE\_LOG\_FILE** and **SOURCE\_LOG\_POS** in the statement, **SOURCE\_LOG\_FILE=''** and **SOURCE\_LOG\_POS=4** are silently appended to it.

**SOURCE\_PUBLIC\_KEY\_PATH**

Enables RSA key pair-based password exchange by providing the path name to a file containing a replica-side copy of the public key required by the source. The file must be in PEM format. This option applies to replicas that authenticate with the **sha256\_password** or **caching\_sha2\_password** authentication plugin. (For **sha256\_password**, **SOURCE\_PUBLIC\_KEY\_PATH** can be used only if MySQL was built using OpenSSL.) If you are using a replication user account that authenticates with the **caching\_sha2\_password** plugin (which is the default from MySQL 8.0), and you are not using a secure connection, you must specify either this option or the **GET\_SOURCE\_PUBLIC\_KEY=1** option to provide the RSA public key to the replica.

**SOURCE\_RETRY\_COUNT**

Sets the maximum number of reconnection attempts that the replica makes after the connection to the source times out, as determined by the [**slave\_net\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_net_timeout) system variable. If the replica does need to reconnect, the first retry occurs immediately after the timeout. The interval between the attempts is specified by the **SOURCE\_CONNECT\_RETRY** option. If both the default settings are used, the replica waits 60 seconds between reconnection attempts (**SOURCE\_CONNECT\_RETRY=60**), and keeps attempting to reconnect at this rate for 60 days (**SOURCE\_RETRY\_COUNT=86400**). These values are recorded in the source metadata repository and shown in the [**replication\_connection\_configuration**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-connection-configuration-table) Performance Schema table. **SOURCE\_RETRY\_COUNT** supersedes the [--master-retry-count](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_master-retry-count) server startup option.

**SOURCE\_SSL\_*xxx*, SOURCE\_TLS\_*xxx***

Specify how the replica uses encryption and ciphers to secure the replication connection. These options can be changed even on replicas that are compiled without SSL support. They are saved to the source metadata repository, but are ignored if the replica does not have SSL support enabled. The **SOURCE\_SSL\_*xxx*** and **SOURCE\_TLS\_*xxx*** options perform the same functions as the **--ssl-*xxx*** and **--tls-*xxx*** client options described in [Command Options for Encrypted Connections](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#encrypted-connection-options). The correspondence between the two sets of options, and the use of the **SOURCE\_SSL\_*xxx*** and **SOURCE\_TLS\_*xxx*** options to set up a secure connection, is explained in [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections).

**SOURCE\_USER**

The user name for the replication user account to use for connecting to the replication source server.

**Important**

To connect to the source using a replication user account that authenticates with the **caching\_sha2\_password** plugin, you must either set up a secure connection as described in [Section 17.3.1, “Setting Up Replication to Use Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-encrypted-connections), or enable the unencrypted connection to support password exchange using an RSA key pair. The **caching\_sha2\_password** authentication plugin is the default for new users created from MySQL 8.0 (for details, see [Section 6.4.1.2, “Caching SHA-2 Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#caching-sha2-pluggable-authentication)). If the user account that you create or use for replication uses this authentication plugin, and you are not using a secure connection, you must enable RSA key pair-based password exchange for a successful connection. You can do this using either the **SOURCE\_PUBLIC\_KEY\_PATH** option or the **GET\_SOURCE\_PUBLIC\_KEY=1** option for this statement.

It is possible to set an empty user name by specifying **SOURCE\_USER=''**, but the replication channel cannot be started with an empty user name. In releases before MySQL 8.0.21, only set an empty **SOURCE\_USER** user name if you need to clear previously used credentials from the replication metadata repositories for security purposes. Do not use the channel afterwards, due to a bug in these releases that can substitute a default user name if an empty user name is read from the repositories (for example, during an automatic restart of a Group Replication channel). From MySQL 8.0.21, it is valid to set an empty **SOURCE\_USER** user name and use the channel afterwards if you always provide user credentials using the [**START REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement or [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement that starts the replication channel. This approach means that the replication channel always needs operator intervention to restart, but the user credentials are not recorded in the replication metadata repositories.

The text of a running [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement, including values for **SOURCE\_USER** and **SOURCE\_PASSWORD**, can be seen in the output of a concurrent [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) statement. (The complete text of a [**START REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement is also visible to [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist).)

**SOURCE\_ZSTD\_COMPRESSION\_LEVEL**

The compression level to use for connections to the replication source server that use the **zstd** compression algorithm. The permitted levels are from 1 to 22, with larger values indicating increasing levels of compression. The default **zstd** compression level is 3. The compression level setting has no effect on connections that do not use **zstd** compression. **SOURCE\_ZSTD\_COMPRESSION\_LEVEL** is available as of MySQL 8.0.18. For more information, see [Section 4.2.8, “Connection Compression Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#connection-compression-control).

The following table shows the maximum permissible length for the string-valued options.

| **Option** | **Maximum Length** |
| --- | --- |
| **SOURCE\_HOST** | 255 |
| **SOURCE\_USER** | 96 |
| **SOURCE\_PASSWORD** | 32 |
| **SOURCE\_LOG\_FILE** | 511 |
| **RELAY\_LOG\_FILE** | 511 |
| **SOURCE\_SSL\_CA** | 511 |
| **SOURCE\_SSL\_CAPATH** | 511 |
| **SOURCE\_SSL\_CERT** | 511 |
| **SOURCE\_SSL\_CRL** | 511 |
| **SOURCE\_SSL\_CRLPATH** | 511 |
| **SOURCE\_SSL\_KEY** | 511 |
| **SOURCE\_SSL\_CIPHER** | 511 |
| **SOURCE\_TLS\_VERSION** | 511 |
| **SOURCE\_TLS\_CIPHERSUITES** | 4000 |
| **SOURCE\_PUBLIC\_KEY\_PATH** | 511 |
| **SOURCE\_COMPRESSION\_ALGORITHMS** | 99 |
| **NETWORK\_NAMESPACE** | 64 |

#### 13.4.2.4 MASTER\_POS\_WAIT() Statement

SELECT MASTER\_POS\_WAIT('***source\_log\_file***', ***source\_log\_pos*** [, ***timeout***][, ***channel***])

This is actually a function, not a statement. It is used to ensure that the replica has read and executed events up to a given position in the source's binary log. See [Section 12.24, “Miscellaneous Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#miscellaneous-functions), for a full description.

#### 13.4.2.5 RESET REPLICA | SLAVE Statement

RESET {REPLICA | SLAVE} [ALL] [***channel\_option***]

***channel\_option***:

FOR CHANNEL ***channel***

**RESET REPLICA | SLAVE** makes the replica forget its position in the source's binary log. From MySQL 8.0.22, use [**RESET REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) in place of [**RESET SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-slave), which is deprecated from that release. In releases before MySQL 8.0.22, use [**RESET SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-slave).

This statement is meant to be used for a clean start; it clears the replication metadata repositories, deletes all the relay log files, and starts a new relay log file. It also resets to 0 the replication delay specified with the **SOURCE\_DELAY** | **MASTER\_DELAY** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23).

**Note**

All relay log files are deleted, even if they have not been completely executed by the replication SQL thread. (This is a condition likely to exist on a replica if you have issued a [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement or if the replica is highly loaded.)

For a server where GTIDs are in use ([**gtid\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_mode) is **ON**), issuing **RESET REPLICA | SLAVE** has no effect on the GTID execution history. The statement does not change the values of **gtid\_executed** or **gtid\_purged**, or the **mysql.gtid\_executed** table. If you need to reset the GTID execution history, use [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master), even if the GTID-enabled server is a replica where binary logging is disabled.

**RESET REPLICA | SLAVE** requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

To use **RESET REPLICA | SLAVE**, the replication SQL thread and replication I/O thread must be stopped, so on a running replica use [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) before issuing **RESET REPLICA | SLAVE**. To use **RESET REPLICA | SLAVE** on a Group Replication group member, the member status must be **OFFLINE**, meaning that the plugin is loaded but the member does not currently belong to any group. A group member can be taken offline by using a [**STOP GROUP REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) statement.

The optional **FOR CHANNEL *channel*** clause enables you to name which replication channel the statement applies to. Providing a **FOR CHANNEL *channel*** clause applies the **RESET REPLICA | SLAVE** statement to a specific replication channel. Combining a **FOR CHANNEL *channel*** clause with the **ALL** option deletes the specified channel. If no channel is named and no extra channels exist, the statement applies to the default channel. Issuing a **RESET REPLICA | SLAVE ALL** statement without a **FOR CHANNEL *channel*** clause when multiple replication channels exist deletes all replication channels and recreates only the default channel. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels) for more information.

**RESET REPLICA | SLAVE** does not change any replication connection parameters, which include the source's host name and port, the replication user account and its password, the **PRIVILEGE\_CHECKS\_USER** account, the **REQUIRE\_ROW\_FORMAT** option, the **REQUIRE\_TABLE\_PRIMARY\_KEY\_CHECK** option, and the **ASSIGN\_GTIDS\_TO\_ANONYMOUS\_TRANSACTIONS**option. If you want to change any of the replication connection parameters, you can do this using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) after the server start. If you want to remove all of the replication connection parameters, use **RESET REPLICA | SLAVE ALL**. **RESET REPLICA | SLAVE ALL** also clears the **IGNORE\_SERVER\_IDS** list set by [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to). When you have used **RESET REPLICA | SLAVE ALL**, if you want to use the instance as a replica again, you need to issue a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement after the server start to specify new connection parameters.

In the event of an unexpected server exit or deliberate restart after issuing **RESET REPLICA | SLAVE** but before issuing [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica), retention of the replication connection parameters depends on the repository used for the replication metadata:

When [**master\_info\_repository=TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository) and [**relay\_log\_info\_repository=TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository) are set on the server (which are the default settings from MySQL 8.0), replication connection parameters are preserved in the crash-safe **InnoDB** tables **mysql.slave\_master\_info** and **mysql.slave\_relay\_log\_info** as part of the **RESET REPLICA | SLAVE** operation. They are also retained in memory. In the event of an unexpected server exit or deliberate restart after issuing **RESET REPLICA | SLAVE** but before issuing [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica), the replication connection parameters are retrieved from the tables and reapplied to the channel. This situation applies from MySQL 8.0.13 for the connection metadata repository, and from MySQL 8.0.19 for the applier metadata repository.

If [**master\_info\_repository=FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_master_info_repository) and [**relay\_log\_info\_repository=FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_info_repository) are set on the server, which is deprecated from MySQL 8.0, or the MySQL Server release is earlier than those specified above, replication connection parameters are only retained in memory. If the replica [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is restarted immediately after issuing **RESET REPLICA | SLAVE** due to an unexpected server exit or deliberate restart, the connection parameters are lost. In that case, you must issue a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23) after the server start to respecify the connection parameters before issuing [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica).

**RESET REPLICA | SLAVE** does not change any replication filter settings (such as [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table)) for channels affected by the statement. However, **RESET REPLICA | SLAVE ALL** removes the replication filters that were set on the channels deleted by the statement. When the deleted channel or channels are recreated, any global replication filters specified for the replica are copied to them, and no channel specific replication filters are applied. For more information see [Section 17.2.5.4, “Replication Channel Based Filters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-rules-channel-based-filters).

**RESET REPLICA | SLAVE** causes an implicit commit of an ongoing transaction. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

If the replication SQL thread was in the middle of replicating temporary tables when it was stopped, and **RESET REPLICA | SLAVE** is issued, these replicated temporary tables are deleted on the replica.

**RESET REPLICA | SLAVE** does not reset the heartbeat period or **SSL\_VERIFY\_SERVER\_CERT**.

**Note**

When used on an NDB Cluster replica SQL node, **RESET REPLICA | SLAVE** clears the **mysql.ndb\_apply\_status** table. You should keep in mind when using this statement that **ndb\_apply\_status** uses the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine and so is shared by all SQL nodes attached to the cluster.

You can override this behavior by issuing [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) **GLOBAL @@**[**ndb\_clear\_apply\_status=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#sysvar_ndb_clear_apply_status) prior to executing **RESET REPLICA | SLAVE**, which keeps the replica from purging the **ndb\_apply\_status** table in such cases.

#### 13.4.2.6 RESET SLAVE | REPLICA Statement

RESET {SLAVE | REPLICA} [ALL] [***channel\_option***]

***channel\_option***:

FOR CHANNEL ***channel***

Makes the replica forget its position in the source's binary log. From MySQL 8.0.22, [**RESET SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-slave) is deprecated and the alias [**RESET REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) should be used instead. In releases before MySQL 8.0.22, use [**RESET SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-slave). The statement works in the same way as before, only the terminology used for the statement and its output has changed. Both versions of the statement update the same status variables when used. Please see the documentation for [**RESET REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) for a description of the statement.

#### 13.4.2.7 START REPLICA | SLAVE Statement

START {REPLICA | SLAVE} [***thread\_types***] [***until\_option***] [***connection\_options***] [***channel\_option***]

***thread\_types***:

[***thread\_type*** [, ***thread\_type***] ... ]

***thread\_type***:

IO\_THREAD | SQL\_THREAD

***until\_option***:

UNTIL { {SQL\_BEFORE\_GTIDS | SQL\_AFTER\_GTIDS} = ***gtid\_set***

| MASTER\_LOG\_FILE = '***log\_name***', MASTER\_LOG\_POS = ***log\_pos***

| SOURCE\_LOG\_FILE = '***log\_name***', SOURCE\_LOG\_POS = ***log\_pos***

| RELAY\_LOG\_FILE = '***log\_name***', RELAY\_LOG\_POS = ***log\_pos***

| SQL\_AFTER\_MTS\_GAPS }

***connection\_options***:

[USER='***user\_name***'] [PASSWORD='***user\_pass***'] [DEFAULT\_AUTH='***plugin\_name***'] [PLUGIN\_DIR='***plugin\_dir***']

***channel\_option***:

FOR CHANNEL ***channel***

***gtid\_set***:

***uuid\_set*** [, ***uuid\_set***] ...

| ''

***uuid\_set***:

***uuid***:***interval***[:***interval***]...

***uuid***:

***hhhhhhhh***-***hhhh***-***hhhh***-***hhhh***-***hhhhhhhhhhhh***

***h***:

[0-9,A-F]

***interval***:

***n***[-***n***]

(***n*** >= 1)

**START REPLICA | SLAVE** starts one or both of the replication threads. From MySQL 8.0.22, use [**START REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) in place of [**START SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-slave), which is deprecated from that release. In releases before MySQL 8.0.22, use [**START SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-slave).

**START REPLICA | SLAVE** with no ***thread\_type*** options starts both of the replication threads. The replication I/O thread reads events from the source server and stores them in the relay log. The replication SQL thread reads events from the relay log and executes them. **START REPLICA | SLAVE** requires the [**REPLICATION\_SLAVE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

If **START REPLICA | SLAVE** succeeds in starting the replication threads, it returns without any error. However, even in that case, it might be that the replication threads start and then later stop (for example, because they do not manage to connect to the source or read its binary log, or some other problem). **START REPLICA | SLAVE** does not warn you about this. You must check the replica's error log for error messages generated by the replication threads, or check that they are running satisfactorily with [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status).

**START REPLICA | SLAVE** causes an implicit commit of an ongoing transaction. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

[**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) must be set to **AUTOMATIC** before issuing this statement.

The optional **FOR CHANNEL *channel*** clause enables you to name which replication channel the statement applies to. Providing a **FOR CHANNEL *channel*** clause applies the **START REPLICA | SLAVE** statement to a specific replication channel. If no clause is named and no extra channels exist, the statement applies to the default channel. If a **START REPLICA | SLAVE** statement does not have a channel defined when using multiple channels, this statement starts the specified threads for all channels. This statement is disallowed for the **group\_replication\_recovery** channel. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels) for more information.

You can add **IO\_THREAD** and **SQL\_THREAD** options to the statement to name which of the threads to start. Note that the Group Replication applier channel (**group\_replication\_applier**) has no I/O thread, only an SQL thread. Specifying the **IO\_THREAD** or **SQL\_THREAD** options when you start this channel has no benefit.

**START REPLICA | SLAVE** supports pluggable user-password authentication with the **USER**, **PASSWORD**, **DEFAULT\_AUTH** and **PLUGIN\_DIR** options, as described in the following list:

**USER**: User name. Cannot be set to an empty or null string, or left unset if **PASSWORD** is used.

**PASSWORD**: Password.

**DEFAULT\_AUTH**: Name of plugin; default is MySQL native authentication.

**PLUGIN\_DIR**: Location of plugin.

You cannot use the **SQL\_THREAD** option when specifying any of **USER**, **PASSWORD**, **DEFAULT\_AUTH**, or **PLUGIN\_DIR**, unless the **IO\_THREAD** option is also provided.

For more information, see [Section 6.2.17, “Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#pluggable-authentication).

If an insecure connection is used with any these options, the server issues the warning Sending passwords in plain text without SSL/TLS is extremely insecure.

**START REPLICA | SLAVE ... UNTIL** supports two additional options for use with global transaction identifiers (GTIDs) (see [Section 17.1.3, “Replication with Global Transaction Identifiers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids)). Each of these takes a set of one or more global transaction identifiers ***gtid\_set*** as an argument (see [GTID Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts-gtid-sets), for more information).

When no ***thread\_type*** is specified, **START REPLICA | SLAVE UNTIL SQL\_BEFORE\_GTIDS** causes the replication SQL thread to process transactions until it has reached the first transaction whose GTID is listed in the ***gtid\_set***. **START REPLICA | SLAVE UNTIL SQL\_AFTER\_GTIDS** causes the replication threads to process all transactions until the ***last*** transaction in the ***gtid\_set*** has been processed by both threads. In other words, **START REPLICA | SLAVE UNTIL SQL\_BEFORE\_GTIDS** causes the replication SQL thread to process all transactions occurring before the first GTID in the ***gtid\_set*** is reached, and **START REPLICA | SLAVE UNTIL SQL\_AFTER\_GTIDS** causes the replication threads to handle all transactions, including those whose GTIDs are found in ***gtid\_set***, until each has encountered a transaction whose GTID is not part of the set. **SQL\_BEFORE\_GTIDS** and **SQL\_AFTER\_GTIDS** each support the **SQL\_THREAD** and **IO\_THREAD** options, although using **IO\_THREAD** with them currently has no effect.

For example, **START REPLICA | SLAVE SQL\_THREAD UNTIL SQL\_BEFORE\_GTIDS = 3E11FA47-71CA-11E1-9E33-C80AA9429562:11-56** causes the replication SQL thread to process all transactions originating from the source whose [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) is **3E11FA47-71CA-11E1-9E33-C80AA9429562** until it encounters the transaction having sequence number 11; it then stops without processing this transaction. In other words, all transactions up to and including the transaction with sequence number 10 are processed. Executing **START REPLICA | SLAVE SQL\_THREAD UNTIL SQL\_AFTER\_GTIDS = 3E11FA47-71CA-11E1-9E33-C80AA9429562:11-56**, on the other hand, would cause the replication SQL thread to obtain all transactions just mentioned from the source, including all of the transactions having the sequence numbers 11 through 56, and then to stop without processing any additional transactions; that is, the transaction having sequence number 56 would be the last transaction fetched by the replication SQL thread.

When using a multithreaded replica with **slave\_preserve\_commit\_order=0** set, there is a chance of gaps in the sequence of transactions that have been executed from the relay log in the following cases:

killing the coordinator thread

after an error occurs in the applier threads

[**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) shuts down unexpectedly

Use the **START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS** statement to cause a multithreaded replica's worker threads to only run until no more gaps are found in the relay log, and then to stop. This statement can take an **SQL\_THREAD** option, but the effects of the statement remain unchanged. It has no effect on the replication I/O thread (and cannot be used with the **IO\_THREAD** option).

Issuing **START REPLICA | SLAVE** on a multithreaded replica with gaps in the sequence of transactions executed from the relay log generates a warning. In such a situation, the solution is to use **START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS**, then issue [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) to remove any remaining relay logs. See [Section 17.5.1.34, “Replication and Transaction Inconsistencies”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transaction-inconsistencies) for more information.

To change a failed multithreaded replica to single-threaded mode, you can issue the following series of statements, in the order shown:

START {REPLICA | SLAVE} UNTIL SQL\_AFTER\_MTS\_GAPS;

SET @@GLOBAL.slave\_parallel\_workers = 0;

START {REPLICA | SLAVE} SQL\_THREAD;

**Note**

It is possible to view the entire text of a running **START REPLICA | SLAVE** statement, including any **USER** or **PASSWORD** values used, in the output of [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist). This is also true for the text of a running [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, including any values it employs for **SOURCE\_USER** | **MASTER\_USER** or **SOURCE\_PASSWORD** | **MASTER\_PASSWORD**.

**START REPLICA | SLAVE** sends an acknowledgment to the user after both the replication I/O thread and the replication SQL thread have started. However, the replication I/O thread may not yet have connected. For this reason, a successful **START REPLICA | SLAVE** causes [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) to show **Replica\_SQL\_Running=Yes**, but this does not guarantee that **Replica\_IO\_Running=Yes** (because **Replica\_IO\_Running=Yes** only if the I/O thread is running and connected). For more information, see [Section 13.7.7.35, “SHOW REPLICA | SLAVE STATUS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status), and [Section 17.1.7.1, “Checking Replication Status”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-status).

An **UNTIL** clause (***until\_option***, in the preceding grammar) may be added to specify that the replica should start and run until the replication SQL thread reaches a given point in the source's binary log or in the replica's relay log. Use one of the following pairs of options to specify the position:

**MASTER\_LOG\_POS** and **MASTER\_LOG\_FILE** for the binary log (to MySQL 8.0.22).

**SOURCE\_LOG\_POS** and **SOURCE\_LOG\_FILE** for the binary log (from MySQL 8.0.23).

**RELAY\_LOG\_POS** and **RELAY\_LOG\_FILE** for the relay log.

For compressed transaction payloads, the position must be based on the compressed **Transaction\_payload\_event**. When the SQL thread reaches the point specified, it stops. If the **SQL\_THREAD** option is specified in the statement, it starts only the SQL thread. Otherwise, it starts both replication threads. If the SQL thread is running, the **UNTIL** clause is ignored and a warning is issued. You cannot use an **UNTIL** clause with the **IO\_THREAD** option.

It is also possible with **START REPLICA | SLAVE UNTIL** to specify a stop point relative to a given GTID or set of GTIDs using one of the options **SQL\_BEFORE\_GTIDS** or **SQL\_AFTER\_GTIDS**, as explained previously in this section. When using one of these options, you can specify **SQL\_THREAD**, **IO\_THREAD**, both of these, or neither of them. If you specify only **SQL\_THREAD**, then only the replication SQL thread is affected by the statement; if only **IO\_THREAD** is used, then only the replication I/O thread is affected. If both **SQL\_THREAD** and **IO\_THREAD** are used, or if neither of them is used, then both the SQL and I/O threads are affected by the statement.

For an **UNTIL** clause, you must specify any one of the following:

Both a log file name and a position in that file

Either of **SQL\_BEFORE\_GTIDS** or **SQL\_AFTER\_GTIDS**

**SQL\_AFTER\_MTS\_GAPS**

Do not mix source and relay log options. Do not mix log file options with GTID options.

The **UNTIL** clause is not supported for multithreaded replicas except when also using **SQL\_AFTER\_MTS\_GAPS**. If **UNTIL** is used on a multithreaded replica without **SQL\_AFTER\_MTS\_GAPS**, the replica operates in single-threaded (sequential) mode for replication until the point specified by the **UNTIL** clause is reached.

Any **UNTIL** condition is reset by a subsequent [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) statement, a **START REPLICA | SLAVE** statement that includes no **UNTIL** clause, or a server restart.

When specifying a log file and position, you can use the **IO\_THREAD** option with **START REPLICA | SLAVE ... UNTIL** even though only the SQL thread is affected by this statement. The **IO\_THREAD** option is ignored in such cases. The preceding restriction does not apply when using one of the GTID options (**SQL\_BEFORE\_GTIDS** and **SQL\_AFTER\_GTIDS**); the GTID options support both **SQL\_THREAD** and **IO\_THREAD**, as explained previously in this section.

The **UNTIL** clause can be useful for debugging replication, or to cause replication to proceed until just before the point where you want to avoid having the replica replicate an event. For example, if an unwise [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement was executed on the source, you can use **UNTIL** to tell the replica to execute up to that point but no farther. To find what the event is, use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) with the source's binary log or the replica's relay log, or by using a [**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events) statement.

If you are using **UNTIL** to have the replica process replicated queries in sections, it is recommended that you start the replica with the [--skip-slave-start](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_skip-slave-start) option, or from MySQL 8.0.24, the [**skip\_slave\_start**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_skip_slave_start) system variable, to prevent the SQL thread from running when the replica server starts. Remove the option or system variable setting after the procedure is complete, so that it is not forgotten in the event of an unexpected server restart.

The [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement includes output fields that display the current values of the **UNTIL** condition.

#### 13.4.2.8 START SLAVE | REPLICA Statement

START {SLAVE | REPLICA} [***thread\_types***] [***until\_option***] [***connection\_options***] [***channel\_option***]

***thread\_types***:

[***thread\_type*** [, ***thread\_type***] ... ]

***thread\_type***:

IO\_THREAD | SQL\_THREAD

***until\_option***:

UNTIL { {SQL\_BEFORE\_GTIDS | SQL\_AFTER\_GTIDS} = ***gtid\_set***

| MASTER\_LOG\_FILE = '***log\_name***', MASTER\_LOG\_POS = ***log\_pos***

| SOURCE\_LOG\_FILE = '***log\_name***', SOURCE\_LOG\_POS = ***log\_pos***

| RELAY\_LOG\_FILE = '***log\_name***', RELAY\_LOG\_POS = ***log\_pos***

| SQL\_AFTER\_MTS\_GAPS }

***connection\_options***:

[USER='***user\_name***'] [PASSWORD='***user\_pass***'] [DEFAULT\_AUTH='***plugin\_name***'] [PLUGIN\_DIR='***plugin\_dir***']

***channel\_option***:

FOR CHANNEL ***channel***

***gtid\_set***:

***uuid\_set*** [, ***uuid\_set***] ...

| ''

***uuid\_set***:

***uuid***:***interval***[:***interval***]...

***uuid***:

***hhhhhhhh***-***hhhh***-***hhhh***-***hhhh***-***hhhhhhhhhhhh***

***h***:

[0-9,A-F]

***interval***:

***n***[-***n***]

(***n*** >= 1)

Starts the replication threads. From MySQL 8.0.22, [**START SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-slave) is deprecated and the alias [**START REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) should be used instead. The statement works in the same way as before, only the terminology used for the statement and its output has changed. Both versions of the statement update the same status variables when used. Please see the documentation for [**START REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) for a description of the statement.

#### 13.4.2.9 STOP REPLICA | SLAVE Statement

STOP {REPLICA | SLAVE} [***thread\_types***] [***channel\_option***]

***thread\_types***:

[***thread\_type*** [, ***thread\_type***] ... ]

***thread\_type***: IO\_THREAD | SQL\_THREAD

***channel\_option***:

FOR CHANNEL ***channel***

Stops the replication threads. From MySQL 8.0.22, use [**STOP REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) in place of [**STOP SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-slave), which is now deprecated. In releases before MySQL 8.0.22, use [**STOP SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-slave).

**STOP REPLICA | SLAVE** requires the [**REPLICATION\_SLAVE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege). Recommended best practice is to execute **STOP REPLICA | SLAVE** on the replica before stopping the replica server (see [Section 5.1.19, “The Server Shutdown Process”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-shutdown), for more information).

Like [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica), this statement may be used with the **IO\_THREAD** and **SQL\_THREAD** options to name the replication thread or threads to be stopped. Note that the Group Replication applier channel (**group\_replication\_applier**) has no replication I/O thread, only a replication SQL thread. Using the **SQL\_THREAD** option therefore stops this channel completely.

**STOP REPLICA | SLAVE** causes an implicit commit of an ongoing transaction. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

[**gtid\_next**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_next) must be set to **AUTOMATIC** before issuing this statement.

You can control how long **STOP REPLICA | SLAVE** waits before timing out by setting the [**rpl\_stop\_slave\_timeout**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_rpl_stop_slave_timeout) system variable. This can be used to avoid deadlocks between **STOP REPLICA | SLAVE** and other SQL statements using different client connections to the replica. When the timeout value is reached, the issuing client returns an error message and stops waiting, but the **STOP REPLICA | SLAVE** instruction remains in effect. Once the replication threads are no longer busy, the **STOP REPLICA | SLAVE** statement is executed and the replica stops.

Some [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statements are allowed while the replica is running, depending on the states of the replication threads. However, using **STOP REPLICA | SLAVE** prior to executing a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement in such cases is still supported. See [Section 13.4.2.3, “CHANGE REPLICATION SOURCE TO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to), [Section 13.4.2.1, “CHANGE MASTER TO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to), and [Section 17.4.8, “Switching Sources During Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-solutions-switch), for more information.

The optional **FOR CHANNEL *channel*** clause enables you to name which replication channel the statement applies to. Providing a **FOR CHANNEL *channel*** clause applies the **STOP REPLICA | SLAVE** statement to a specific replication channel. If no channel is named and no extra channels exist, the statement applies to the default channel. If a **STOP REPLICA | SLAVE** statement does not name a channel when using multiple channels, this statement stops the specified threads for all channels. This statement cannot be used with the **group\_replication\_recovery** channel. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels) for more information.

When the replica is multithreaded ([**slave\_parallel\_workers**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_slave_parallel_workers) is a nonzero value), any gaps in the sequence of transactions executed from the relay log are closed as part of stopping the worker threads. If the replica is stopped unexpectedly (for example due to an error in a worker thread, or another thread issuing [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill)) while a **STOP REPLICA | SLAVE** statement is executing, the sequence of executed transactions from the relay log may become inconsistent. See [Section 17.5.1.34, “Replication and Transaction Inconsistencies”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transaction-inconsistencies), for more information.

When the source is using the row-based binary logging format, you should execute **STOP REPLICA | SLAVE** or **STOP REPLICA | SLAVE SQL\_THREAD** on the replica prior to shutting down the replica server if you are replicating any tables that use a nontransactional storage engine. If the current replication event group has modified one or more nontransactional tables, **STOP REPLICA | SLAVE** waits for up to 60 seconds for the event group to complete, unless you issue a [**KILL QUERY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) or [**KILL CONNECTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) statement for the replication SQL thread. If the event group remains incomplete after the timeout, an error message is logged.

When the source is using the statement-based binary logging format, changing the source while it has open temporary tables is potentially unsafe. This is one of the reasons why statement-based replication of temporary tables is not recommended. You can find out whether there are any temporary tables on the replica by checking the value of [**Slave\_open\_temp\_tables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Slave_open_temp_tables); when using statement-based replication, this value should be 0 before executing [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to). If there are any temporary tables open on the replica, issuing a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement after issuing a **STOP REPLICA | SLAVE** causes an [**ER\_WARN\_OPEN\_TEMP\_TABLES\_MUST\_BE\_ZERO**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_warn_open_temp_tables_must_be_zero) warning.

#### 13.4.2.10 STOP SLAVE | REPLICA Statement

STOP {SLAVE | REPLICA} [***thread\_types***] [***channel\_option***]

***thread\_types***:

[***thread\_type*** [, ***thread\_type***] ... ]

***thread\_type***: IO\_THREAD | SQL\_THREAD

***channel\_option***:

FOR CHANNEL ***channel***

Stops the replication threads. From MySQL 8.0.22, [**STOP SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-slave) is deprecated and the alias [**STOP REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) should be used instead. The statement works in the same way as before, only the terminology used for the statement and its output has changed. Both versions of the statement update the same status variables when used. Please see the documentation for [**STOP REPLICA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) for a description of the statement.

#### 13.4.2.11 Functions which Configure the Source List

The following functions, which are available from MySQL 8.0.22 for standard source to replica replication and from MySQL 8.0.23 for Group Replication, enable you to add and remove replication source servers from the source list for a replication channel. The asynchronous connection failover mechanism automatically establishes an asynchronous (source to replica) replication connection to a new source from the appropriate list after the existing connection from the replica to its source fails. From MySQL 8.0.23, the connection is also changed if the currently connected source does not have the highest weighted priority in the group. For Group Replication source servers that are defined as part of a managed group, the connection is also failed over to another group member if the currently connected source leaves the group or is no longer in the majority. For more information on the mechanism, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

The source lists are stored in the **mysql.replication\_asynchronous\_connection\_failover** and **mysql.replication\_asynchronous\_connection\_failover\_managed** tables, and can be viewed in the Performance Schema table [**replication\_asynchronous\_connection\_failover**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-asynchronous-connection-failover-table).

[**asynchronous\_connection\_failover\_add\_source()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-add-source)

Add configuration information for a replication source server to the source list for a replication channel.

Syntax:

asynchronous\_connection\_failover\_add\_source(***channel***, ***host***, ***port***, ***network\_namespace***, ***weight***)

Arguments:

***channel***: The replication channel for which this replication source server is part of the source list.

***host***: The host name for this replication source server.

***port***: The port number for this replication source server.

***network\_namespace***: The network namespace for this replication source server. Specify an empty string, as this parameter is reserved for future use.

***weight***: The priority of this replication source server in the replication channel's source list. The priority is from 1 to 100, with 100 being the highest, and 50 being the default. When the asynchronous connection failover mechanism activates, the source with the highest priority setting among the alternative sources listed in the source list for the channel is chosen for the first connection attempt. If this attempt does not work, the replica tries with all the listed sources in descending order of priority, then starts again from the highest priority source. If multiple sources have the same priority, the replica orders them randomly. From MySQL 8.0.23, the asynchronous connection failover mechanism activates if the currently connected source is not the highest weighted in the group.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

**SELECT asynchronous\_connection\_failover\_add\_source('channel2', '127.0.0.1', 3310, '', 80);**

+-------------------------------------------------------------------------------------------------+

| asynchronous\_connection\_failover\_add\_source('channel2', '127.0.0.1', 3310, '', 80) |

+-------------------------------------------------------------------------------------------------+

| Source configuration details successfully inserted. |

+-------------------------------------------------------------------------------------------------+

For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

[**asynchronous\_connection\_failover\_delete\_source()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_asynchronous-connection-failover-delete-source)

Remove configuration information for a replication source server from the source list for a replication channel.

Syntax:

asynchronous\_connection\_failover\_delete\_source(***channel***, ***host***, ***port***, ***network\_namespace***)

Arguments:

***channel***: The replication channel for which this replication source server was part of the source list.

***host***: The host name for this replication source server.

***port***: The port number for this replication source server.

***network\_namespace***: The network namespace for this replication source server. Specify an empty string, as this parameter is reserved for future use.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

**SELECT asynchronous\_connection\_failover\_delete\_source('channel2', '127.0.0.1', 3310, '');**

+------------------------------------------------------------------------------------------------+

| asynchronous\_connection\_failover\_delete\_source('channel2', '127.0.0.1', 3310, '') |

+------------------------------------------------------------------------------------------------+

| Source configuration details successfully deleted. |

+------------------------------------------------------------------------------------------------+

For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

**[asynchronous\_connection\_failover\_add\_managed()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_asynchronous-connection-failover-add-managed)**

Add configuration information for a replication source server that is part of a managed group (a Group Replication group member) to the source list for a replication channel. You only need to add one group member. The replica automatically adds the rest from the current group membership, then keeps the source list updated in line with membership change.

Syntax:

asynchronous\_connection\_failover\_add\_managed(***channel***, ***managed\_type***, ***managed\_name***, ***host***, ***port***, ***network\_namespace***, ***primary\_weight***, ***secondary\_weight***)

Arguments:

***channel***: The replication channel for which this replication source server is part of the source list.

***managed\_type***: The type of managed service that the asynchronous connection failover mechanism must provide for this server. The only value currently accepted is **GroupReplication**.

***managed\_name***: The identifier for the managed group that the server is a part of. For the **GroupReplication** managed service, the identifier is the value of the [**group\_replication\_group\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_name) system variable.

***host***: The host name for this replication source server.

***port***: The port number for this replication source server.

***network\_namespace***: The network namespace for this replication source server. Specify an empty string, as this parameter is reserved for future use.

***primary\_weight***: The priority of this replication source server in the replication channel's source list when it is acting as the primary for the managed group. The weight is from 1 to 100, with 100 being the highest. For the primary, 80 is a suitable weight. The asynchronous connection failover mechanism activates if the currently connected source is not the highest weighted in the group. Assuming that you set up the managed group to give a higher weight to a primary and a lower weight to a secondary, when the primary changes, its weight increases, and the replica changes over the connection to it.

***secondary\_weight***: The priority of this replication source server in the replication channel's source list when it is acting as a secondary in the managed group. The weight is from 1 to 100, with 100 being the highest. For a secondary, 60 is a suitable weight.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

**SELECT asynchronous\_connection\_failover\_add\_managed('channel2', 'GroupReplication', 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa', '127.0.0.1', 3310, '', 80, 60);**

+----------------------------------------------------------------------------------------------------------------------------------------------------+

| asynchronous\_connection\_failover\_add\_source('channel2', 'GroupReplication', 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa', '127.0.0.1', 3310, '', 80, 60) |

+----------------------------------------------------------------------------------------------------------------------------------------------------+

| Source managed configuration details successfully inserted. |

+----------------------------------------------------------------------------------------------------------------------------------------------------+

For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

**[asynchronous\_connection\_failover\_delete\_managed()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_asynchronous-connection-failover-delete-managed)**

Remove an entire managed group from the source list for a replication channel. When you use this UDF, all the replication source servers defined in the managed group are removed from the channel's source list.

Syntax:

asynchronous\_connection\_failover\_delete\_managed(***channel***, ***managed\_name***)

Arguments:

***channel***: The replication channel for which this replication source server was part of the source list.

***managed\_name***: The identifier for the managed group that the server is a part of. For the **GroupReplication** managed service, the identifier is the value of the [**group\_replication\_group\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_group_name) system variable.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

**SELECT asynchronous\_connection\_failover\_delete\_managed('channel2', 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa');**

+-----------------------------------------------------------------------------------------------------+

| asynchronous\_connection\_failover\_delete\_managed('channel2', 'aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa') |

+-----------------------------------------------------------------------------------------------------+

| Source managed configuration details successfully deleted. |

+-----------------------------------------------------------------------------------------------------+

For more information, see [Section 17.4.9, “Switching Sources with Asynchronous Connection Failover”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-asynchronous-connection-failover).

### 13.4.3 SQL Statements for Controlling Group Replication

[13.4.3.1 START GROUP\_REPLICATION Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication)

[13.4.3.2 STOP GROUP\_REPLICATION Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication)

[13.4.3.3 Function which Configures Group Replication Primary](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#group-replication-functions-for-new-primary)

[13.4.3.4 Functions which Configure the Group Replication Mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#group-replication-functions-for-mode)

[13.4.3.5 Functions to Inspect and Configure the Maximum Consensus Instances of a Group](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#group-replication-functions-for-maximum-consensus)

[13.4.3.6 Functions to Inspect and Set the Group Replication Communication Protocol Version](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#group-replication-functions-for-communication-protocol)

This section provides information about the statements used for controlling group replication.

#### 13.4.3.1 START GROUP\_REPLICATION Statement

START GROUP\_REPLICATION

[USER='***user\_name***']

[, PASSWORD='***user\_pass***']

[, DEFAULT\_AUTH='***plugin\_name***']

Starts group replication. This statement requires the [**GROUP\_REPLICATION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_group-replication-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege). If [**super\_read\_only=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) is set and the member should join as a primary, [**super\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only) is set to **OFF** once Group Replication successfully starts.

From MySQL 8.0.21, you can specify user credentials for distributed recovery on the **START GROUP\_REPLICATION** statement using the **USER**, **PASSWORD**, and **DEFAULT\_AUTH** options, as follows:

**USER**: The replication user for distributed recovery. For instructions to set up this account, see [Section 18.2.1.3, “User Credentials For Distributed Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-user-credentials). You cannot specify an empty or null string, or omit the **USER** option if **PASSWORD** is specified.

**PASSWORD**: The password for the replication user account. The password cannot be encrypted, but it is masked in the query log.

**DEFAULT\_AUTH**: The name of the authentication plugin used for the replication user account. If you do not specify this option, MySQL native authentication (the **mysql\_native\_password** plugin) is assumed. This option acts as a hint to the server, and the donor for distributed recovery overrides it if a different plugin is associated with the user account on that server. The authentication plugin used by default when you create user accounts in MySQL 8 is the caching SHA-2 authentication plugin (**caching\_sha2\_password**). See [Section 6.2.17, “Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#pluggable-authentication) for more information on authentication plugins.

These credentials are used for distributed recovery on the **group\_replication\_recovery** channel. When you specify user credentials on **START GROUP\_REPLICATION**, the credentials are saved in memory only, and are removed by a **STOP GROUP\_REPLICATION** statement or server shutdown. You must issue a **START GROUP\_REPLICATION** statement to provide the credentials again. This method is therefore not compatible with starting Group Replication automatically on server start, as specified by the [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) system variable.

User credentials specified on **START GROUP\_REPLICATION** take precedence over any user credentials set for the **group\_replication\_recovery** channel using a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). Note that user credentials set using these statements are stored in the replication metadata repositories, and are used when **START GROUP\_REPLICATION** is specified without user credentials, including automatic starts if the [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) system variable is set to **ON**. To gain the security benefits of specifying user credentials on **START GROUP\_REPLICATION**, ensure that [**group\_replication\_start\_on\_boot**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_start_on_boot) is set to **OFF** (the default is **ON**), and clear any user credentials previously set for the **group\_replication\_recovery** channel, following the instructions in [Section 18.6.3, “Securing Distributed Recovery Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-distributed-recovery-securing).

#### 13.4.3.2 STOP GROUP\_REPLICATION Statement

STOP GROUP\_REPLICATION

Stops Group Replication. This statement requires the [**GROUP\_REPLICATION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_group-replication-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege). As soon as you issue [**STOP GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-group-replication) the member is set to [**super\_read\_only=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_super_read_only), which ensures that no writes can be made to the member while Group Replication stops. Any other replication channels running on the member are also stopped. Any user credentials that you specified on the [**START GROUP\_REPLICATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-group-replication) statement when starting Group Replication on this member are removed from memory, and must be supplied when you start Group Replication again.

**Warning**

Use this statement with extreme caution because it removes the server instance from the group, meaning it is no longer protected by Group Replication's consistency guarantee mechanisms. To be completely safe, ensure that your applications can no longer connect to the instance before issuing this statement to avoid any chance of stale reads.

#### 13.4.3.3 Function which Configures Group Replication Primary

The following function enables you to configure which member of a single-primary replication group is the primary.

**[group\_replication\_set\_as\_primary()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_group-replication-set-as-primary)**

Appoints a specific member of the group as the new primary, overriding any election process. Pass in ***member\_uuid*** which is the [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) of the member that you want to become the new primary. Must be issued on a member of a replication group running in single-primary mode.

Syntax:

STRING group\_replication\_set\_as\_primary(***member\_uuid***)

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

SELECT group\_replication\_set\_as\_primary(***member\_uuid***)

For more information, see [Section 18.5.1.1, “Changing a Group's Primary Member”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-changing-primary-member)

#### 13.4.3.4 Functions which Configure the Group Replication Mode

The following functions enable you to control the mode which a replication group is running in, either single-primary or multi-primary mode.

**[group\_replication\_switch\_to\_single\_primary\_mode()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_group-replication-switch-to-single-primary-mode)**

Changes a group running in multi-primary mode to single-primary mode, without the need to stop Group Replication. Must be issued on a member of a replication group running in multi-primary mode. When you change to single-primary mode, strict consistency checks are also disabled on all group members, as required in single-primary mode ([**group\_replication\_enforce\_update\_everywhere\_checks=OFF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#sysvar_group_replication_enforce_update_everywhere_checks)).

Syntax:

STRING group\_replication\_switch\_to\_single\_primary\_mode([***str***])

Arguments:

***str***: A string containing the UUID of a member of the group which should become the new single primary. Other members of the group become secondaries.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

**SELECT group\_replication\_switch\_to\_single\_primary\_mode(*member\_uuid*);**

For more information, see [Section 18.5.1.2, “Changing a Group's Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-changing-group-mode)

**[group\_replication\_switch\_to\_multi\_primary\_mode()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_group-replication-switch-to-multi-primary-mode)**

Changes a group running in single-primary mode to multi-primary mode. Must be issued on a member of a replication group running in single-primary mode.

Syntax:

STRING group\_replication\_switch\_to\_multi\_primary\_mode()

This function has no parameters.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

SELECT group\_replication\_switch\_to\_multi\_primary\_mode()

All members which belong to the group become primaries.

For more information, see [Section 18.5.1.2, “Changing a Group's Mode”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-changing-group-mode)

#### 13.4.3.5 Functions to Inspect and Configure the Maximum Consensus Instances of a Group

The following functions enable you to inspect and configure the maximum number of consensus instances that a group can execute in parallel.

**[group\_replication\_get\_write\_concurrency()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_group-replication-get-write-concurrency)**

Check the maximum number of consensus instances that a group can execute in parallel.

Syntax:

INT group\_replication\_get\_write\_concurrency()

This function has no parameters.

Return value:

The maximum number of consensus instances currently set for the group.

Example:

SELECT group\_replication\_get\_write\_concurrency()

For more information, see [Section 18.5.1.3, “Using Group Replication Group Write Consensus”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-group-write-consensus).

**[group\_replication\_set\_write\_concurrency()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_group-replication-set-write-concurrency)**

Configures the maximum number of consensus instances that a group can execute in parallel. The [**GROUP\_REPLICATION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_group-replication-admin) privilege is required to use this UDF.

Syntax:

STRING group\_replication\_set\_write\_concurrency(***instances***)

Arguments:

***members***: Sets the maximum number of consensus instances that a group can execute in parallel. Default value is 10, valid values are integers in the range of 10 to 200.

Return value:

Any resulting error as a string.

Example:

**SELECT group\_replication\_set\_write\_concurrency(*instances*);**

For more information, see [Section 18.5.1.3, “Using Group Replication Group Write Consensus”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-group-write-consensus).

#### 13.4.3.6 Functions to Inspect and Set the Group Replication Communication Protocol Version

The following functions enable you to inspect and configure the Group Replication communication protocol version that is used by a replication group.

**[group\_replication\_get\_communication\_protocol()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_group-replication-get-communication-protocol)**

Inspect the Group Replication communication protocol version that is currently in use for a group.

Syntax:

STRING group\_replication\_get\_communication\_protocol()

This function has no parameters.

Return value:

The oldest MySQL Server version that can join this group and use the group's communication protocol. Versions from MySQL 5.7.14 allow compression of messages, and versions from MySQL 8.0.16 also allow fragmentation of messages. Note that the [**group\_replication\_get\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-get-communication-protocol) UDF returns the minimum MySQL version that the group supports, which might differ from the version number that was passed to the [**group\_replication\_set\_communication\_protocol()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#udf_group-replication-set-communication-protocol) UDF, and from the MySQL Server version that is installed on the member where you use the UDF.

If the protocol cannot be inspected because this server instance does not belong to a replication group, an error is returned as a string.

Example:

**SELECT group\_replication\_get\_communication\_protocol();**

+------------------------------------------------+

| group\_replication\_get\_communication\_protocol() |

+------------------------------------------------+

| 8.0.16 |

+------------------------------------------------+

For more information, see [Section 18.5.1.4, “Setting a Group's Communication Protocol Version”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-communication-protocol).

**[group\_replication\_set\_communication\_protocol()](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "udf_group-replication-set-communication-protocol)**

Downgrade the Group Replication communication protocol version of a group so that members at earlier releases can join, or upgrade the Group Replication communication protocol version of a group after upgrading MySQL Server on all members. The [**GROUP\_REPLICATION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_group-replication-admin) privilege is required to use this UDF, and all existing group members must be online when you issue the statement, with no loss of majority.

**Note**

For MySQL InnoDB cluster, the communication protocol version is managed automatically whenever the cluster topology is changed using AdminAPI operations. You do not have to use these UDFs yourself for an InnoDB cluster.

Syntax:

STRING group\_replication\_set\_communication\_protocol(***version***)

Arguments:

***version***: For a downgrade, specify the MySQL Server version of the prospective group member that has the oldest installed server version. In this case, the command makes the group fall back to a communication protocol compatible with that server version if possible. The minimum server version that you can specify is MySQL 5.7.14. For an upgrade, specify the new MySQL Server version to which the existing group members have been upgraded.

Return value:

A string containing the result of the operation, for example whether it was successful or not.

Example:

**SELECT group\_replication\_set\_communication\_protocol("5.7.25");**

For more information, see [Section 18.5.1.4, “Setting a Group's Communication Protocol Version”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\group-replication.html#group-replication-communication-protocol).

## 13.5 Prepared Statements

[13.5.1 PREPARE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare)

[13.5.2 EXECUTE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute)

[13.5.3 DEALLOCATE PREPARE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#deallocate-prepare)

MySQL 8.0 provides support for server-side prepared statements. This support takes advantage of the efficient client/server binary protocol. Using prepared statements with placeholders for parameter values has the following benefits:

Less overhead for parsing the statement each time it is executed. Typically, database applications process large volumes of almost-identical statements, with only changes to literal or variable values in clauses such as **WHERE** for queries and deletes, **SET** for updates, and **VALUES** for inserts.

Protection against SQL injection attacks. The parameter values can contain unescaped SQL quote and delimiter characters.

The following sections provide an overview of the characteristics of prepared statements:

[Prepared Statements in Application Programs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepared-statements-in-applications)

[Prepared Statements in SQL Scripts](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepared-statements-in-scripts)

[PREPARE, EXECUTE, and DEALLOCATE PREPARE Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepared-statement-types)

[SQL Syntax Permitted in Prepared Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepared-statements-permitted)

### Prepared Statements in Application Programs

You can use server-side prepared statements through client programming interfaces, including the [MySQL C API client library](https://dev.mysql.com/doc/c-api/8.0/en/) for C programs, [MySQL Connector/J](https://dev.mysql.com/doc/connector-j/8.0/en/) for Java programs, and [MySQL Connector/NET](https://dev.mysql.com/doc/connector-net/en/) for programs using .NET technologies. For example, the C API provides a set of function calls that make up its prepared statement API. See [C API Prepared Statement Interface](https://dev.mysql.com/doc/c-api/8.0/en/c-api-prepared-statement-interface.html). Other language interfaces can provide support for prepared statements that use the binary protocol by linking in the C client library, one example being the [**mysqli** extension](http://php.net/mysqli), available in PHP 5.0 and higher.

### Prepared Statements in SQL Scripts

An alternative SQL interface to prepared statements is available. This interface is not as efficient as using the binary protocol through a prepared statement API, but requires no programming because it is available directly at the SQL level:

You can use it when no programming interface is available to you.

You can use it from any program that can send SQL statements to the server to be executed, such as the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client program.

You can use it even if the client is using an old version of the client library.

SQL syntax for prepared statements is intended to be used for situations such as these:

To test how prepared statements work in your application before coding it.

To use prepared statements when you do not have access to a programming API that supports them.

To interactively troubleshoot application issues with prepared statements.

To create a test case that reproduces a problem with prepared statements, so that you can file a bug report.

### PREPARE, EXECUTE, and DEALLOCATE PREPARE Statements

SQL syntax for prepared statements is based on three SQL statements:

[**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare) prepares a statement for execution (see [Section 13.5.1, “PREPARE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare)).

[**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute) executes a prepared statement (see [Section 13.5.2, “EXECUTE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute)).

[**DEALLOCATE PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#deallocate-prepare) releases a prepared statement (see [Section 13.5.3, “DEALLOCATE PREPARE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#deallocate-prepare)).

The following examples show two equivalent ways of preparing a statement that computes the hypotenuse of a triangle given the lengths of the two sides.

The first example shows how to create a prepared statement by using a string literal to supply the text of the statement:

mysql> **PREPARE stmt1 FROM 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';**

mysql> **SET @a = 3;**

mysql> **SET @b = 4;**

mysql> **EXECUTE stmt1 USING @a, @b;**

+------------+

| hypotenuse |

+------------+

| 5 |

+------------+

mysql> **DEALLOCATE PREPARE stmt1;**

The second example is similar, but supplies the text of the statement as a user variable:

mysql> **SET @s = 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';**

mysql> **PREPARE stmt2 FROM @s;**

mysql> **SET @a = 6;**

mysql> **SET @b = 8;**

mysql> **EXECUTE stmt2 USING @a, @b;**

+------------+

| hypotenuse |

+------------+

| 10 |

+------------+

mysql> **DEALLOCATE PREPARE stmt2;**

Here is an additional example that demonstrates how to choose the table on which to perform a query at runtime, by storing the name of the table as a user variable:

mysql> **USE test;**

mysql> **CREATE TABLE t1 (a INT NOT NULL);**

mysql> **INSERT INTO t1 VALUES (4), (8), (11), (32), (80);**

mysql> **SET @table = 't1';**

mysql> **SET @s = CONCAT('SELECT \* FROM ', @table);**

mysql> **PREPARE stmt3 FROM @s;**

mysql> **EXECUTE stmt3;**

+----+

| a |

+----+

| 4 |

| 8 |

| 11 |

| 32 |

| 80 |

+----+

mysql> **DEALLOCATE PREPARE stmt3;**

A prepared statement is specific to the session in which it was created. If you terminate a session without deallocating a previously prepared statement, the server deallocates it automatically.

A prepared statement is also global to the session. If you create a prepared statement within a stored routine, it is not deallocated when the stored routine ends.

To guard against too many prepared statements being created simultaneously, set the [**max\_prepared\_stmt\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_prepared_stmt_count) system variable. To prevent the use of prepared statements, set the value to 0.

### SQL Syntax Permitted in Prepared Statements

The following SQL statements can be used as prepared statements:

ALTER TABLE

ALTER USER

ANALYZE TABLE

CACHE INDEX

CALL

CHANGE MASTER

CHECKSUM {TABLE | TABLES}

COMMIT

{CREATE | DROP} INDEX

{CREATE | RENAME | DROP} DATABASE

{CREATE | DROP} TABLE

{CREATE | RENAME | DROP} USER

{CREATE | DROP} VIEW

DELETE

DO

FLUSH {TABLE | TABLES | TABLES WITH READ LOCK | HOSTS | PRIVILEGES

| LOGS | STATUS | MASTER | SLAVE | USER\_RESOURCES}

GRANT

INSERT

INSTALL PLUGIN

KILL

LOAD INDEX INTO CACHE

OPTIMIZE TABLE

RENAME TABLE

REPAIR TABLE

REPLACE

RESET {MASTER | SLAVE}

REVOKE

SELECT

SET

SHOW BINLOG EVENTS

SHOW CREATE {PROCEDURE | FUNCTION | EVENT | TABLE | VIEW}

SHOW {MASTER | BINARY} LOGS

SHOW {MASTER | SLAVE} STATUS

SLAVE {START | STOP}

TRUNCATE TABLE

UNINSTALL PLUGIN

UPDATE

Other statements are not supported.

For compliance with the SQL standard, which states that diagnostics statements are not preparable, MySQL does not support the following as prepared statements:

**SHOW WARNINGS**, **SHOW COUNT(\*) WARNINGS**

**SHOW ERRORS**, **SHOW COUNT(\*) ERRORS**

Statements containing any reference to the [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) or [**error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_error_count) system variable.

Generally, statements not permitted in SQL prepared statements are also not permitted in stored programs. Exceptions are noted in [Section 25.8, “Restrictions on Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-program-restrictions).

Metadata changes to tables or views referred to by prepared statements are detected and cause automatic repreparation of the statement when it is next executed. For more information, see [Section 8.10.3, “Caching of Prepared Statements and Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#statement-caching).

Placeholders can be used for the arguments of the **LIMIT** clause when using prepared statements. See [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

In prepared [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) statements used with [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare) and [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute), placeholder support for **OUT** and **INOUT** parameters is available beginning with MySQL 8.0. See [Section 13.2.1, “CALL Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call), for an example and a workaround for earlier versions. Placeholders can be used for **IN** parameters regardless of version.

SQL syntax for prepared statements cannot be used in nested fashion. That is, a statement passed to [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare) cannot itself be a [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare), [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute), or [**DEALLOCATE PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#deallocate-prepare) statement.

SQL syntax for prepared statements is distinct from using prepared statement API calls. For example, you cannot use the [**mysql\_stmt\_prepare()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-stmt-prepare.html) C API function to prepare a [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare), [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute), or [**DEALLOCATE PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#deallocate-prepare) statement.

SQL syntax for prepared statements can be used within stored procedures, but not in stored functions or triggers. However, a cursor cannot be used for a dynamic statement that is prepared and executed with [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare) and [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute). The statement for a cursor is checked at cursor creation time, so the statement cannot be dynamic.

SQL syntax for prepared statements does not support multi-statements (that is, multiple statements within a single string separated by **;** characters).

To write C programs that use the [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) SQL statement to execute stored procedures that contain prepared statements, the **CLIENT\_MULTI\_RESULTS** flag must be enabled. This is because each [**CALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) returns a result to indicate the call status, in addition to any result sets that might be returned by statements executed within the procedure.

**CLIENT\_MULTI\_RESULTS** can be enabled when you call [**mysql\_real\_connect()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-real-connect.html), either explicitly by passing the **CLIENT\_MULTI\_RESULTS** flag itself, or implicitly by passing **CLIENT\_MULTI\_STATEMENTS** (which also enables **CLIENT\_MULTI\_RESULTS**). For additional information, see [Section 13.2.1, “CALL Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call).

### 13.5.1 PREPARE Statement

PREPARE ***stmt\_name*** FROM ***preparable\_stmt***

The [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare) statement prepares a SQL statement and assigns it a name, ***stmt\_name***, by which to refer to the statement later. The prepared statement is executed with [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute) and released with [**DEALLOCATE PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#deallocate-prepare). For examples, see [Section 13.5, “Prepared Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#sql-prepared-statements).

Statement names are not case-sensitive. ***preparable\_stmt*** is either a string literal or a user variable that contains the text of the SQL statement. The text must represent a single 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 quotation marks, even if you intend to bind them to string values. Parameter markers can be used only where data values should appear, not for SQL keywords, identifiers, and so forth.

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.

The scope of a prepared statement is the session within which it is created, which as several implications:

A prepared statement created in one session is not available to other sessions.

When a session ends, whether normally or abnormally, its prepared statements no longer exist. If auto-reconnect is enabled, the client is not notified that the connection was lost. For this reason, clients may wish to disable auto-reconnect. See [Automatic Reconnection Control](https://dev.mysql.com/doc/c-api/8.0/en/c-api-auto-reconnect.html).

A prepared statement created within a stored program continues to exist after the program finishes executing and can be executed outside the program later.

A statement prepared in stored program context cannot refer to stored procedure or function parameters or local variables because they go out of scope when the program ends and would be unavailable were the statement to be executed later outside the program. As a workaround, refer instead to user-defined variables, which also have session scope; see [Section 9.4, “User-Defined Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#user-variables).

Beginning with MySQL 8.0.22, a parameter used in a prepared statement has its type determined when the statement is first prepared, and retains this type whenever [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#execute) is invoked for this prepared statement (unless the statement is reprepared, as explained later in this section). Rules for determining a parameter's type are listed here:

A parameter which is an operand of a binary arithmetic operator has the same data type as the other operand.

If both operands of a binary arithmetic operator are parameters, the type of the parameters is decided by the context of the operator.

If a parameter is the operand of a unary arithmetic operator, the parameter's type is decided by the context of the operator.

If an arithmetic operator has no type-determining context, the derived type for any parameters involved is [**DOUBLE PRECISION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types). This can happen, for example, when the parameter is a top-level node in a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) list, or when it is part of a comparison operator.

A parameter which is an operand of a character string operator has the same derived type as the aggregated type of the other operands. If all operands of the operator are parameters, the derived type is [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char); its collation is determined by the value of [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection).

A parameter which is an operand of a temporal operator has type [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) if the operator returns a **DATETIME**, [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) if the operator returns a **TIME**, and [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) if the operator returns a **DATE**.

A parameter which is an operand of a binary comparison operator has the same derived type as the other operand of the comparison.

A parameter that is an operand of a ternary comparison operator such as [**BETWEEN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_between) has the same derived type as the aggregated type of the other operands.

If all operands of a comparison operator are parameters, the derived type for each of them is [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), with collation determined by the value of [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection).

A parameter that is an output operand of any of [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case), [**COALESCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coalesce), [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if), [**IFNULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ifnull), or [**NULLIF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nullif) has the same derived type as the aggregated type of the operator's other output operands.

If all output operands of any of [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case), [**COALESCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coalesce), [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if), [**IFNULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ifnull), or [**NULLIF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_nullif) are parameters, or they are all **NULL**, the type of the parameter is decided by the context of the operator.

If the parameter is an operand of any of of [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case), [**COALESCE()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_coalesce), [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if), or [**IFNULL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ifnull), and has no type-determining context, the derived type for each of the parameters involved is [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and its collation is determined by the value of [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection).

A parameter which is the operand of a [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast) has the same type as specified by the **CAST()**.

If a parameter is an immediate member of a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) list that is not part of an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement, the derived type of the parameter is [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), and its collation is determined by the value of [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection).

If a parameter is an immediate member of a **SELECT** list that is part of an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement, the derived type of the parameter is the type of the corresponding column into which the parameter is inserted.

If a parameter is used as source for an assignment in a **SET** clause of an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement or in the **ON DUPLICATE KEY UPDATE** clause of an [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statement, the derived type of the parameter is the type of the corresponding column which is updated by the **SET** or **ON DUPLICATE KEY UPDATE** clause.

If a parameter is an argument of a function, the derived type depends on the function's return type.

For some combinations of actual type and derived type, an automatic repreparation of the statement is triggered, to ensure closer compatibility with previous versions of MySQL. Repreparation does not occur if any of the following conditions are true:

**NULL** is used as the actual parameter value.

A parameter is an operand of a [**CAST()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_cast). (Instead, a cast to the derived type is attempted, and an exception raised if the cast fails.)

A parameter is a string. (In this case, an implicit **CAST(? AS *derived\_type*)** is performed.)

The derived type and actual type of the parameter are both [**INTEGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types) and have the same sign.

The parameter's derived type is [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) and its actual type is either **DECIMAL** or [**INTEGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#integer-types).

The derived type is [**DOUBLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#floating-point-types) and the actual type is any numeric type.

Both the derived type and the actual type are string types.

If the derived type is temporal and the actual type is temporal. Exceptions: The derived type is [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time) and the actual type is not **TIME**; the derived type is [**DATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) and the actual type is not **DATE**.

The derived type is temporal and the actual type is numeric.

For cases other than those just listed, the statement is reprepared and the actual parameter types are used instead of the derived parameter types.

These rules also apply to a user variable referenced in a prepared statement.

Using a different data type for a given parameter or user variable within a prepared statement for executions of the statement subsequent to the first execution causes the statement to be reprepared. This is less efficient; it may also lead to the paremeter's (or variable's) actual type to vary, and thus for results to be inconsistent, with subsequent executions of the prepared statement. For these reasons, it is advisable to use the same data type for a given parameter when re-executing a prepared statement.

### 13.5.2 EXECUTE Statement

EXECUTE ***stmt\_name***

[USING @***var\_name*** [, @***var\_name***] ...]

After preparing a statement with [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare), you execute it with an [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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.

For examples, see [Section 13.5, “Prepared Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#sql-prepared-statements).

### 13.5.3 DEALLOCATE PREPARE Statement

{DEALLOCATE | DROP} PREPARE ***stmt\_name***

To deallocate a prepared statement produced with [**PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare), use a [**DEALLOCATE PREPARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#deallocate-prepare) statement that refers to the prepared statement name. Attempting to execute a prepared statement after deallocating it results in an error. If too many prepared statements are created and not deallocated by either the **DEALLOCATE PREPARE** statement or the end of the session, you might encounter the upper limit enforced by the [**max\_prepared\_stmt\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_prepared_stmt_count) system variable.

For examples, see [Section 13.5, “Prepared Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#sql-prepared-statements).

## 13.6 Compound Statement Syntax

[13.6.1 BEGIN ... END Compound Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end)

[13.6.2 Statement Labels](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#statement-labels)

[13.6.3 DECLARE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare)

[13.6.4 Variables in Stored Programs](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stored-program-variables)

[13.6.5 Flow Control Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flow-control-statements)

[13.6.6 Cursors](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursors)

[13.6.7 Condition Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#condition-handling)

[13.6.8 Restrictions on Condition Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#condition-handling-restrictions)

This section describes the syntax for the [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) compound statement and other statements that can be used in the body of stored programs: Stored procedures and functions, triggers, and events. These objects are defined in terms of SQL code that is stored on the server for later invocation (see [Chapter 25, *Stored Objects*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html)).

A compound statement is a block that can contain other blocks; declarations for variables, condition handlers, and cursors; and flow control constructs such as loops and conditional tests.

### 13.6.1 BEGIN ... END Compound Statement

[***begin\_label***:] BEGIN

[***statement\_list***]

END [***end\_label***]

[**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) syntax is used for writing compound statements, which can appear within stored programs (stored procedures and functions, triggers, and events). 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 (**;**) statement delimiter. The ***statement\_list*** itself is optional, so the empty compound statement (**BEGIN END**) is legal.

[**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) blocks can be nested.

Use of multiple statements requires that a client is able to send statement strings containing the **;** statement delimiter. In the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) command-line client, this is handled with the **delimiter** command. Changing the **;** end-of-statement delimiter (for example, to **//**) permit **;** to be used in a program body. For an example, see [Section 25.1, “Defining Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-programs-defining).

A [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block can be labeled. See [Section 13.6.2, “Statement Labels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#statement-labels).

The optional **[NOT] ATOMIC** clause is not supported. This means that no transactional savepoint is set at the start of the instruction block and the **BEGIN** clause used in this context has no effect on the current transaction.

**Note**

Within all stored programs, the parser treats [**BEGIN [WORK]**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) as the beginning of a [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block. To begin a transaction in this context, use [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) instead.

### 13.6.2 Statement Labels

[***begin\_label***:] BEGIN

[***statement\_list***]

END [***end\_label***]

[***begin\_label***:] LOOP

***statement\_list***

END LOOP [***end\_label***]

[***begin\_label***:] REPEAT

***statement\_list***

UNTIL ***search\_condition***

END REPEAT [***end\_label***]

[***begin\_label***:] WHILE ***search\_condition*** DO

***statement\_list***

END WHILE [***end\_label***]

Labels are permitted for [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) blocks and for the [**LOOP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#loop), [**REPEAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat), and [**WHILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#while) statements. Label use for those statements follows these rules:

***begin\_label*** must be followed by a colon.

***begin\_label*** can be given without ***end\_label***. If ***end\_label*** is present, it must be the same as ***begin\_label***.

***end\_label*** cannot be given without ***begin\_label***.

Labels at the same nesting level must be distinct.

Labels can be up to 16 characters long.

To refer to a label within the labeled construct, use an [**ITERATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#iterate) or [**LEAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#leave) statement. The following example uses those statements to continue iterating or terminate the loop:

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;

END;

The scope of a block label does not include the code for handlers declared within the block. For details, see [Section 13.6.7.2, “DECLARE ... HANDLER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler).

### 13.6.3 DECLARE Statement

The [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare) statement is used to define various items local to a program:

Local variables. See [Section 13.6.4, “Variables in Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stored-program-variables).

Conditions and handlers. See [Section 13.6.7, “Condition Handling”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#condition-handling).

Cursors. See [Section 13.6.6, “Cursors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursors).

[**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare) is permitted only inside a [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) compound statement and must be at its start, before any other statements.

Declarations must follow a certain order. Cursor declarations must appear before handler declarations. Variable and condition declarations must appear before cursor or handler declarations.

### 13.6.4 Variables in Stored Programs

[13.6.4.1 Local Variable DECLARE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-local-variable)

[13.6.4.2 Local Variable Scope and Resolution](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#local-variable-scope)

System variables and user-defined variables can be used in stored programs, just as they can be used outside stored-program context. In addition, stored programs can use **DECLARE** to define local variables, and stored routines (procedures and functions) can be declared to take parameters that communicate values between the routine and its caller.

To declare local variables, use the [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-local-variable) statement, as described in [Section 13.6.4.1, “Local Variable DECLARE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-local-variable).

Variables can be set directly with the [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement. See [Section 13.7.6.1, “SET Syntax for Variable Assignment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable).

Results from queries can be retrieved into local variables using [**SELECT ... INTO *var\_list***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into) or by opening a cursor and using [**FETCH ... INTO *var\_list***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#fetch). See [Section 13.2.10.1, “SELECT ... INTO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select-into), and [Section 13.6.6, “Cursors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursors).

For information about the scope of local variables and how MySQL resolves ambiguous names, see [Section 13.6.4.2, “Local Variable Scope and Resolution”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#local-variable-scope).

It is not permitted to assign the value **DEFAULT** to stored procedure or function parameters or stored program local variables (for example with a **SET *var\_name* = DEFAULT** statement). In MySQL 8.0, this results in a syntax error.

#### 13.6.4.1 Local Variable DECLARE Statement

DECLARE ***var\_name*** [, ***var\_name***] ... ***type*** [DEFAULT ***value***]

This statement declares local variables within stored programs. To provide a default value for a variable, include a **DEFAULT** clause. The value can be specified as an expression; 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 [Section 13.1.17, “CREATE PROCEDURE and CREATE FUNCTION Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure).

Variable declarations must appear before cursor or handler declarations.

Local variable names are not case-sensitive. Permissible characters and quoting rules are the same as for other identifiers, as described in [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers).

The scope of a local variable is the [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block within which 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.

For examples of variable declarations, see [Section 13.6.4.2, “Local Variable Scope and Resolution”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#local-variable-scope).

#### 13.6.4.2 Local Variable Scope and Resolution

The scope of a local variable is the [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block within which 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.

Because local variables are in scope only during stored program execution, references to them are not permitted in prepared statements created within a stored program. Prepared statement scope is the current session, not the stored program, so the statement could be executed after the program ends, at which point the variables would no longer be in scope. For example, **SELECT ... INTO *local\_var*** cannot be used as a prepared statement. This restriction also applies to stored procedure and function parameters. See [Section 13.5.1, “PREPARE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#prepare).

A local variable should not have the same name as a table column. If an SQL statement, such as a [**SELECT ... INTO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, contains a reference to a column and a declared local variable with the same name, MySQL currently interprets the reference as the name of a variable. Consider the following procedure definition:

CREATE PROCEDURE sp1 (x VARCHAR(5))

BEGIN

DECLARE xname VARCHAR(5) DEFAULT 'bob';

DECLARE newname VARCHAR(5);

DECLARE xid INT;

SELECT xname, id INTO newname, xid

FROM table1 WHERE xname = xname;

SELECT newname;

END;

MySQL interprets **xname** in the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement as a reference to the **xname** variable rather than the **xname** column. Consequently, when the procedure **sp1()**is called, the **newname** variable returns the value **'bob'** regardless of the value of the **table1.xname** column.

Similarly, the cursor definition in the following procedure contains a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement that refers to **xname**. MySQL interprets this as a reference to the variable of that name rather than a column reference.

CREATE PROCEDURE sp2 (x VARCHAR(5))

BEGIN

DECLARE xname VARCHAR(5) DEFAULT 'bob';

DECLARE newname VARCHAR(5);

DECLARE xid INT;

DECLARE done TINYINT DEFAULT 0;

DECLARE cur1 CURSOR FOR SELECT xname, id FROM table1;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur1;

read\_loop: LOOP

FETCH FROM cur1 INTO newname, xid;

IF done THEN LEAVE read\_loop; END IF;

SELECT newname;

END LOOP;

CLOSE cur1;

END;

See also [Section 25.8, “Restrictions on Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-program-restrictions).

### 13.6.5 Flow Control Statements

[13.6.5.1 CASE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case)

[13.6.5.2 IF Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if)

[13.6.5.3 ITERATE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#iterate)

[13.6.5.4 LEAVE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#leave)

[13.6.5.5 LOOP Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#loop)

[13.6.5.6 REPEAT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat)

[13.6.5.7 RETURN Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return)

[13.6.5.8 WHILE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#while)

MySQL supports the [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if), [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case), [**ITERATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#iterate), [**LEAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#leave) [**LOOP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#loop), [**WHILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#while), and [**REPEAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat) constructs for flow control within stored programs. It also supports [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) within stored functions.

Many of these constructs contain other statements, as indicated by the grammar specifications in the following sections. Such constructs may be nested. For example, an [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) statement might contain a [**WHILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#while) loop, which itself contains a [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case) statement.

MySQL does not support **FOR** loops.

#### 13.6.5.1 CASE Statement

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

The [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case) statement for stored programs implements a complex conditional construct.

**Note**

There is also a [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_case) operator, which differs from the [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case) statement described here. See [Section 12.5, “Flow Control Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#flow-control-functions). The [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case) statement cannot have an **ELSE NULL** clause, and it is terminated with **END CASE** instead of **END**.

For the first syntax, ***case\_value*** is an expression. This value is compared to the ***when\_value*** expression in each **WHEN** clause until one of them is equal. When an equal ***when\_value*** is found, the corresponding **THEN** clause ***statement\_list*** executes. If no ***when\_value*** is equal, the **ELSE** clause ***statement\_list*** executes, if there is one.

This syntax cannot be used to test for equality with **NULL** because **NULL = NULL** is false. See [Section 3.3.4.6, “Working with NULL Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\tutorial.html#working-with-null).

For the second syntax, each **WHEN** clause ***search\_condition*** expression is evaluated until one is true, at which point its corresponding **THEN** clause ***statement\_list*** executes. If no ***search\_condition*** is equal, the **ELSE** clause ***statement\_list*** executes, if there is one.

If no ***when\_value*** or ***search\_condition*** matches the value tested and the [**CASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#case) statement contains no **ELSE** clause, a Case not found for CASE statement error results.

Each ***statement\_list*** consists of one or more SQL statements; an empty ***statement\_list*** is not permitted.

To handle situations where no value is matched by any **WHEN** clause, use an **ELSE** containing an empty [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block, as shown in this example. (The indentation used here in the **ELSE** clause is for purposes of clarity only, and is not otherwise significant.)

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;

|

#### 13.6.5.2 IF Statement

IF ***search\_condition*** THEN ***statement\_list***

[ELSEIF ***search\_condition*** THEN ***statement\_list***] ...

[ELSE ***statement\_list***]

END IF

The [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) statement for stored programs implements a basic conditional construct.

**Note**

There is also an [**IF()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_if) function, which differs from the [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) statement described here. See [Section 12.5, “Flow Control Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#flow-control-functions). The [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) statement can have **THEN**, **ELSE**, and **ELSEIF** clauses, and it is terminated with **END IF**.

If a given ***search\_condition*** evaluates to true, the corresponding **THEN** or **ELSEIF** clause ***statement\_list*** executes. If no ***search\_condition*** matches, the **ELSE** clause ***statement\_list*** executes.

Each ***statement\_list*** consists of one or more SQL statements; an empty ***statement\_list*** is not permitted.

An **IF ... END IF** block, like all other flow-control blocks used within stored programs, must be terminated with a semicolon, as shown in this example:

DELIMITER //

CREATE FUNCTION SimpleCompare(n INT, m INT)

RETURNS VARCHAR(20)

BEGIN

DECLARE s VARCHAR(20);

IF n > m THEN SET s = '>';

ELSEIF n = m THEN SET s = '=';

ELSE SET s = '<';

END IF;

SET s = CONCAT(n, ' ', s, ' ', m);

RETURN s;

END //

DELIMITER ;

As with other flow-control constructs, **IF ... END IF** blocks may be nested within other flow-control constructs, including other [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) statements. Each [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) must be terminated by its own **END IF** followed by a semicolon. You can use indentation to make nested flow-control blocks more easily readable by humans (although this is not required by MySQL), as shown here:

DELIMITER //

CREATE FUNCTION VerboseCompare (n INT, m INT)

RETURNS VARCHAR(50)

BEGIN

DECLARE s VARCHAR(50);

IF n = m THEN SET s = 'equals';

ELSE

IF n > m THEN SET s = 'greater';

ELSE SET s = 'less';

END IF;

SET s = CONCAT('is ', s, ' than');

END IF;

SET s = CONCAT(n, ' ', s, ' ', m, '.');

RETURN s;

END //

DELIMITER ;

In this example, the inner [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) is evaluated only if **n** is not equal to **m**.

#### 13.6.5.3 ITERATE Statement

ITERATE ***label***

[**ITERATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#iterate) can appear only within [**LOOP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#loop), [**REPEAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat), and [**WHILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#while) statements. [**ITERATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#iterate) means “start the loop again.”

For an example, see [Section 13.6.5.5, “LOOP Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#loop).

#### 13.6.5.4 LEAVE Statement

LEAVE ***label***

This statement is used to exit the flow control construct that has the given label. If the label is for the outermost stored program block, [**LEAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#leave) exits the program.

[**LEAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#leave) can be used within [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) or loop constructs ([**LOOP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#loop), [**REPEAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat), [**WHILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#while)).

For an example, see [Section 13.6.5.5, “LOOP Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#loop).

#### 13.6.5.5 LOOP Statement

[***begin\_label***:] LOOP

***statement\_list***

END LOOP [***end\_label***]

[**LOOP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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 (**;**) statement delimiter. The statements within the loop are repeated until the loop is terminated. Usually, this is accomplished with a [**LEAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#leave) statement. Within a stored function, [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) can also be used, which exits the function entirely.

Neglecting to include a loop-termination statement results in an infinite loop.

A [**LOOP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#loop) statement can be labeled. For the rules regarding label use, see [Section 13.6.2, “Statement Labels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#statement-labels).

Example:

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;

#### 13.6.5.6 REPEAT Statement

[***begin\_label***:] REPEAT

***statement\_list***

UNTIL ***search\_condition***

END REPEAT [***end\_label***]

The statement list within a [**REPEAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat) statement is repeated until the ***search\_condition*** expression is true. Thus, a [**REPEAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat) always enters the loop at least once. ***statement\_list*** consists of one or more statements, each terminated by a semicolon (**;**) statement delimiter.

A [**REPEAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat) statement can be labeled. For the rules regarding label use, see [Section 13.6.2, “Statement Labels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#statement-labels).

Example:

mysql> **delimiter //**

mysql> **CREATE PROCEDURE dorepeat(p1 INT)**

**BEGIN**

**SET @x = 0;**

**REPEAT**

**SET @x = @x + 1;**

**UNTIL @x > p1 END REPEAT;**

**END**

**//**

Query OK, 0 rows affected (0.00 sec)

mysql> **CALL dorepeat(1000)//**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @x//**

+------+

| @x |

+------+

| 1001 |

+------+

1 row in set (0.00 sec)

#### 13.6.5.7 RETURN Statement

RETURN ***expr***

The [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) statement terminates execution of a stored function and returns the value ***expr*** to the function caller. There must be at least one [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) statement in a stored function. There may be more than one if the function has multiple exit points.

This statement is not used in stored procedures, triggers, or events. The [**LEAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#leave) statement can be used to exit a stored program of those types.

#### 13.6.5.8 WHILE Statement

[***begin\_label***:] WHILE ***search\_condition*** DO

***statement\_list***

END WHILE [***end\_label***]

The statement list within a [**WHILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#while) statement is repeated as long as the ***search\_condition*** expression is true. ***statement\_list*** consists of one or more SQL statements, each terminated by a semicolon (**;**) statement delimiter.

A [**WHILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#while) statement can be labeled. For the rules regarding label use, see [Section 13.6.2, “Statement Labels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#statement-labels).

Example:

CREATE PROCEDURE dowhile()

BEGIN

DECLARE v1 INT DEFAULT 5;

WHILE v1 > 0 DO

...

SET v1 = v1 - 1;

END WHILE;

END;

### 13.6.6 Cursors

[13.6.6.1 Cursor CLOSE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#close)

[13.6.6.2 Cursor DECLARE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-cursor)

[13.6.6.3 Cursor FETCH Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#fetch)

[13.6.6.4 Cursor OPEN Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#open)

[13.6.6.5 Restrictions on Server-Side Cursors](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursor-restrictions)

MySQL supports cursors inside stored programs. The syntax is as in embedded SQL. Cursors have these properties:

Asensitive: The server may or may not make a copy of its result table

Read only: Not updatable

Nonscrollable: Can be traversed only in one direction and cannot skip rows

Cursor declarations must appear before handler declarations and after variable and condition declarations.

Example:

CREATE PROCEDURE curdemo()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE a CHAR(16);

DECLARE b, c INT;

DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;

DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur1;

OPEN cur2;

read\_loop: LOOP

FETCH cur1 INTO a, b;

FETCH cur2 INTO c;

IF done THEN

LEAVE read\_loop;

END IF;

IF b < c THEN

INSERT INTO test.t3 VALUES (a,b);

ELSE

INSERT INTO test.t3 VALUES (a,c);

END IF;

END LOOP;

CLOSE cur1;

CLOSE cur2;

END;

#### 13.6.6.1 Cursor CLOSE Statement

CLOSE ***cursor\_name***

This statement closes a previously opened cursor. For an example, see [Section 13.6.6, “Cursors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursors).

An error occurs if the cursor is not open.

If not closed explicitly, a cursor is closed at the end of the [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block in which it was declared.

#### 13.6.6.2 Cursor DECLARE Statement

DECLARE ***cursor\_name*** CURSOR FOR ***select\_statement***

This statement declares a cursor and associates it with a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement that retrieves the rows to be traversed by the cursor. To fetch the rows later, use a [**FETCH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#fetch) statement. The number of columns retrieved by the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement must match the number of output variables specified in the [**FETCH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#fetch) statement.

The [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement cannot have an **INTO** clause.

Cursor declarations must appear before handler declarations and after variable and condition declarations.

A stored program may contain multiple cursor declarations, but each cursor declared in a given block must have a unique name. For an example, see [Section 13.6.6, “Cursors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursors).

For information available through [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statements, it is possible in many cases to obtain equivalent information by using a cursor with an **INFORMATION\_SCHEMA** table.

#### 13.6.6.3 Cursor FETCH Statement

FETCH [[NEXT] FROM] ***cursor\_name*** INTO ***var\_name*** [, ***var\_name***] ...

This statement fetches the next row for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement associated with the specified cursor (which must be open), and advances the cursor pointer. If a row exists, the fetched columns are stored in the named variables. The number of columns retrieved by the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement must match the number of output variables specified in the [**FETCH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#fetch) statement.

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). For an example, see [Section 13.6.6, “Cursors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursors).

Be aware that another operation, such as a **SELECT** or another **FETCH**, may also cause the handler to execute by raising the same condition. If it is necessary to distinguish which operation raised the condition, place the operation within its own [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block so that it can be associated with its own handler.

#### 13.6.6.4 Cursor OPEN Statement

OPEN ***cursor\_name***

This statement opens a previously declared cursor. For an example, see [Section 13.6.6, “Cursors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursors).

#### 13.6.6.5 Restrictions on Server-Side Cursors

Server-side cursors are implemented in the C API using the [**mysql\_stmt\_attr\_set()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-stmt-attr-set.html) function. The same implementation is used for cursors in stored routines. A server-side cursor enables a result set to be generated on the server side, but not transferred to the client except for those rows that the client requests. For example, if a client executes a query but is only interested in the first row, the remaining rows are not transferred.

In MySQL, a server-side cursor is materialized into an internal temporary table. Initially, this is a **MEMORY** table, but is converted to a **MyISAM** table when its size exceeds the minimum value of the [**max\_heap\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_heap_table_size) and [**tmp\_table\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_tmp_table_size) system variables. The same restrictions apply to internal temporary tables created to hold the result set for a cursor as for other uses of internal temporary tables. See [Section 8.4.4, “Internal Temporary Table Use in MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#internal-temporary-tables). One limitation of the implementation is that for a large result set, retrieving its rows through a cursor might be slow.

Cursors are read only; you cannot use a cursor to update rows.

**UPDATE WHERE CURRENT OF** and **DELETE WHERE CURRENT OF** are not implemented, because updatable cursors are not supported.

Cursors are nonholdable (not held open after a commit).

Cursors are asensitive.

Cursors are nonscrollable.

Cursors are not named. The statement handler acts as the cursor ID.

You can have open only a single cursor per prepared statement. If you need several cursors, you must prepare several statements.

You cannot use a cursor for a statement that generates a result set if the statement is not supported in prepared mode. This includes statements such as [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table), **HANDLER READ**, and [**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events).

### 13.6.7 Condition Handling

[13.6.7.1 DECLARE ... CONDITION Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition)

[13.6.7.2 DECLARE ... HANDLER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler)

[13.6.7.3 GET DIAGNOSTICS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics)

[13.6.7.4 RESIGNAL Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal)

[13.6.7.5 SIGNAL Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal)

[13.6.7.6 Scope Rules for Handlers](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler-scope)

[13.6.7.7 The MySQL Diagnostics Area](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area)

[13.6.7.8 Condition Handling and OUT or INOUT Parameters](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#conditions-and-parameters)

Conditions may arise during stored program execution that require special handling, such as exiting the current program block or continuing execution. Handlers can be defined for general conditions such as warnings or exceptions, or for specific conditions such as a particular error code. Specific conditions can be assigned names and referred to that way in handlers.

To name a condition, use the [**DECLARE ... CONDITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition) statement. To declare a handler, use the [**DECLARE ... HANDLER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler) statement. See [Section 13.6.7.1, “DECLARE ... CONDITION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition), and [Section 13.6.7.2, “DECLARE ... HANDLER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler). For information about how the server chooses handlers when a condition occurs, see [Section 13.6.7.6, “Scope Rules for Handlers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler-scope).

To raise a condition, use the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement. To modify condition information within a condition handler, use [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal). See [Section 13.6.7.1, “DECLARE ... CONDITION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition), and [Section 13.6.7.2, “DECLARE ... HANDLER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler).

To retrieve information from the diagnostics area, use the [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statement (see [Section 13.6.7.3, “GET DIAGNOSTICS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics)). For information about the diagnostics area, see [Section 13.6.7.7, “The MySQL Diagnostics Area”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area).

#### 13.6.7.1 DECLARE ... CONDITION Statement

DECLARE ***condition\_name*** CONDITION FOR ***condition\_value***

***condition\_value***: {

***mysql\_error\_code***

| SQLSTATE [VALUE] ***sqlstate\_value***

}

The [**DECLARE ... CONDITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition) statement declares a named error condition, associating a name with a condition that needs specific handling. The name can be referred to in a subsequent [**DECLARE ... HANDLER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler) statement (see [Section 13.6.7.2, “DECLARE ... HANDLER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler)).

Condition declarations must appear before cursor or handler declarations.

The ***condition\_value*** for [**DECLARE ... CONDITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition) indicates the specific condition or class of conditions to associate with the condition name. It can take the following forms:

***mysql\_error\_code***: An integer literal indicating a MySQL error code.

Do not use MySQL error code 0 because that indicates success rather than an error condition. For a list of MySQL error codes, see [Server Error Message Reference](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html).

SQLSTATE [VALUE] ***sqlstate\_value***: A 5-character string literal indicating an SQLSTATE value.

Do not use SQLSTATE values that begin with **'00'** because those indicate success rather than an error condition. For a list of SQLSTATE values, see [Server Error Message Reference](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html).

Condition names referred to in [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) or use [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statements must be associated with SQLSTATE values, not MySQL error codes.

Using names for conditions can help make stored program code clearer. For example, this handler applies to attempts to drop a nonexistent table, but that is apparent only if you know that 1051 is the MySQL error code for “unknown table”:

DECLARE CONTINUE HANDLER FOR 1051

BEGIN

-- body of handler

END;

By declaring a name for the condition, the purpose of the handler is more readily seen:

DECLARE no\_such\_table CONDITION FOR 1051;

DECLARE CONTINUE HANDLER FOR no\_such\_table

BEGIN

-- body of handler

END;

Here is a named condition for the same condition, but based on the corresponding SQLSTATE value rather than the MySQL error code:

DECLARE no\_such\_table CONDITION FOR SQLSTATE '42S02';

DECLARE CONTINUE HANDLER FOR no\_such\_table

BEGIN

-- body of handler

END;

#### 13.6.7.2 DECLARE ... HANDLER Statement

DECLARE ***handler\_action*** HANDLER

FOR ***condition\_value*** [, ***condition\_value***] ...

***statement***

***handler\_action***: {

CONTINUE

| EXIT

| UNDO

}

***condition\_value***: {

***mysql\_error\_code***

| SQLSTATE [VALUE] ***sqlstate\_value***

| ***condition\_name***

| SQLWARNING

| NOT FOUND

| SQLEXCEPTION

}

The [**DECLARE ... HANDLER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler) statement specifies a handler that deals with one or more conditions. If one of these conditions occurs, the specified ***statement*** executes. ***statement*** can be a simple statement such as **SET *var\_name* = *value***, or a compound statement written using **BEGIN** and **END** (see [Section 13.6.1, “BEGIN ... END Compound Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end)).

Handler declarations must appear after variable or condition declarations.

The ***handler\_action*** value indicates what action the handler takes after execution of the handler statement:

**CONTINUE**: Execution of the current program continues.

**EXIT**: Execution terminates for the [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) compound statement in which the handler is declared. This is true even if the condition occurs in an inner block.

**UNDO**: Not supported.

The ***condition\_value*** for [**DECLARE ... HANDLER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler) indicates the specific condition or class of conditions that activates the handler. It can take the following forms:

***mysql\_error\_code***: An integer literal indicating a MySQL error code, such as 1051 to specify “unknown table”:

DECLARE CONTINUE HANDLER FOR 1051

BEGIN

-- body of handler

END;

Do not use MySQL error code 0 because that indicates success rather than an error condition. For a list of MySQL error codes, see [Server Error Message Reference](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html).

SQLSTATE [VALUE] ***sqlstate\_value***: A 5-character string literal indicating an SQLSTATE value, such as **'42S01'** to specify “unknown table”:

DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'

BEGIN

-- body of handler

END;

Do not use SQLSTATE values that begin with **'00'** because those indicate success rather than an error condition. For a list of SQLSTATE values, see [Server Error Message Reference](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html).

***condition\_name***: A condition name previously specified with [**DECLARE ... CONDITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition). A condition name can be associated with a MySQL error code or SQLSTATE value. See [Section 13.6.7.1, “DECLARE ... CONDITION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition).

**SQLWARNING**: Shorthand for the class of SQLSTATE values that begin with **'01'**.

DECLARE CONTINUE HANDLER FOR SQLWARNING

BEGIN

-- body of handler

END;

**NOT FOUND**: Shorthand for the class of SQLSTATE values that begin with **'02'**. This is relevant within 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.

DECLARE CONTINUE HANDLER FOR NOT FOUND

BEGIN

-- body of handler

END;

For another example, see [Section 13.6.6, “Cursors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cursors). The **NOT FOUND** condition also occurs for **SELECT ... INTO *var\_list*** statements that retrieve no rows.

**SQLEXCEPTION**: Shorthand for the class of SQLSTATE values that do not begin with **'00'**, **'01'**, or **'02'**.

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

BEGIN

-- body of handler

END;

For information about how the server chooses handlers when a condition occurs, see [Section 13.6.7.6, “Scope Rules for Handlers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler-scope).

If a condition occurs for which no handler has been declared, the action taken depends on the condition class:

For **SQLEXCEPTION** conditions, the stored program terminates at the statement that raised the condition, as if there were an **EXIT** handler. If the program was called by another stored program, the calling program handles the condition using the handler selection rules applied to its own handlers.

For **SQLWARNING** conditions, the program continues executing, as if there were a **CONTINUE** handler.

For **NOT FOUND** conditions, if the condition was raised normally, the action is **CONTINUE**. If it was raised by [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) or [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal), the action is **EXIT**.

The following example uses a handler for **SQLSTATE '23000'**, which occurs for a duplicate-key error:

mysql> **CREATE TABLE test.t (s1 INT, PRIMARY KEY (s1));**

Query OK, 0 rows affected (0.00 sec)

mysql> **delimiter //**

mysql> **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;**

**//**

Query OK, 0 rows affected (0.00 sec)

mysql> **CALL handlerdemo()//**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @x//**

+------+

| @x |

+------+

| 3 |

+------+

1 row in set (0.00 sec)

Notice that **@x** is **3** after the procedure executes, which shows that execution continued to the end of the procedure after the error occurred. If the [**DECLARE ... HANDLER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler) statement had not been present, MySQL would have taken the default action (**EXIT**) after the second [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) failed due to the **PRIMARY KEY** constraint, and **SELECT @x** would have returned **2**.

To ignore a condition, declare a **CONTINUE** handler for it and associate it with an empty block. For example:

DECLARE CONTINUE HANDLER FOR SQLWARNING BEGIN END;

The scope of a block label does not include the code for handlers declared within the block. Therefore, the statement associated with a handler cannot use [**ITERATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#iterate) or [**LEAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#leave) to refer to labels for blocks that enclose the handler declaration. Consider the following example, where the [**REPEAT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repeat) block has a label of **retry**:

CREATE PROCEDURE p ()

BEGIN

DECLARE i INT DEFAULT 3;

retry:

REPEAT

BEGIN

DECLARE CONTINUE HANDLER FOR SQLWARNING

BEGIN

ITERATE retry; # illegal

END;

IF i < 0 THEN

LEAVE retry; # legal

END IF;

SET i = i - 1;

END;

UNTIL FALSE END REPEAT;

END;

The **retry** label is in scope for the [**IF**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#if) statement within the block. It is not in scope for the **CONTINUE** handler, so the reference there is invalid and results in an error:

ERROR 1308 (42000): LEAVE with no matching label: retry

To avoid references to outer labels in handlers, use one of these strategies:

To leave the block, use an **EXIT** handler. If no block cleanup is required, the [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) handler body can be empty:

DECLARE EXIT HANDLER FOR SQLWARNING BEGIN END;

Otherwise, put the cleanup statements in the handler body:

DECLARE EXIT HANDLER FOR SQLWARNING

BEGIN

***block cleanup statements***

END;

To continue execution, set a status variable in a **CONTINUE** handler that can be checked in the enclosing block to determine whether the handler was invoked. The following example uses the variable **done** for this purpose:

CREATE PROCEDURE p ()

BEGIN

DECLARE i INT DEFAULT 3;

DECLARE done INT DEFAULT FALSE;

retry:

REPEAT

BEGIN

DECLARE CONTINUE HANDLER FOR SQLWARNING

BEGIN

SET done = TRUE;

END;

IF done OR i < 0 THEN

LEAVE retry;

END IF;

SET i = i - 1;

END;

UNTIL FALSE END REPEAT;

END;

#### 13.6.7.3 GET DIAGNOSTICS Statement

GET [CURRENT | STACKED] DIAGNOSTICS {

***statement\_information\_item***

[, ***statement\_information\_item***] ...

| CONDITION ***condition\_number***

***condition\_information\_item***

[, ***condition\_information\_item***] ...

}

***statement\_information\_item***:

***target*** = ***statement\_information\_item\_name***

***condition\_information\_item***:

***target*** = ***condition\_information\_item\_name***

***statement\_information\_item\_name***: {

NUMBER

| ROW\_COUNT

}

***condition\_information\_item\_name***: {

CLASS\_ORIGIN

| SUBCLASS\_ORIGIN

| RETURNED\_SQLSTATE

| MESSAGE\_TEXT

| MYSQL\_ERRNO

| CONSTRAINT\_CATALOG

| CONSTRAINT\_SCHEMA

| CONSTRAINT\_NAME

| CATALOG\_NAME

| SCHEMA\_NAME

| TABLE\_NAME

| COLUMN\_NAME

| CURSOR\_NAME

}

***condition\_number***, ***target***:

(see following discussion)

SQL statements produce diagnostic information that populates the diagnostics area. The [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statement enables applications to inspect this information. (You can also use [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) or [**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors) to see conditions or errors.)

No special privileges are required to execute [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics).

The keyword **CURRENT** means to retrieve information from the current diagnostics area. The keyword **STACKED** means to retrieve information from the second diagnostics area, which is available only if the current context is a condition handler. If neither keyword is given, the default is to use the current diagnostics area.

The [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statement is typically used in a handler within a stored program. It is a MySQL extension that [**GET [CURRENT] DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) is permitted outside handler context to check the execution of any SQL statement. For example, if you invoke the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client program, you can enter these statements at the prompt:

mysql> **DROP TABLE test.no\_such\_table;**

ERROR 1051 (42S02): Unknown table 'test.no\_such\_table'

mysql> **GET DIAGNOSTICS CONDITION 1**

**@p1 = RETURNED\_SQLSTATE, @p2 = MESSAGE\_TEXT;**

mysql> **SELECT @p1, @p2;**

+-------+------------------------------------+

| @p1 | @p2 |

+-------+------------------------------------+

| 42S02 | Unknown table 'test.no\_such\_table' |

+-------+------------------------------------+

This extension applies only to the current diagnostics area. It does not apply to the second diagnostics area because **GET STACKED DIAGNOSTICS** is permitted only if the current context is a condition handler. If that is not the case, a **GET STACKED DIAGNOSTICS when handler not active** error occurs.

For a description of the diagnostics area, see [Section 13.6.7.7, “The MySQL Diagnostics Area”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area). Briefly, it contains two kinds of information:

Statement information, such as the number of conditions that occurred or the affected-rows count.

Condition information, such as the error code and message. If a statement raises multiple conditions, this part of the diagnostics area has a condition area for each one. If a statement raises no conditions, this part of the diagnostics area is empty.

For a statement that produces three conditions, the diagnostics area contains statement and condition information like this:

Statement information:

row count

... other statement information items ...

Condition area list:

Condition area 1:

error code for condition 1

error message for condition 1

... other condition information items ...

Condition area 2:

error code for condition 2:

error message for condition 2

... other condition information items ...

Condition area 3:

error code for condition 3

error message for condition 3

... other condition information items ...

[**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) can obtain either statement or condition information, but not both in the same statement:

To obtain statement information, retrieve the desired statement items into target variables. This instance of [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) assigns the number of available conditions and the rows-affected count to the user variables **@p1** and **@p2**:

GET DIAGNOSTICS @p1 = NUMBER, @p2 = ROW\_COUNT;

To obtain condition information, specify the condition number and retrieve the desired condition items into target variables. This instance of [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) assigns the SQLSTATE value and error message to the user variables **@p3** and **@p4**:

GET DIAGNOSTICS CONDITION 1

@p3 = RETURNED\_SQLSTATE, @p4 = MESSAGE\_TEXT;

The retrieval list specifies one or more ***target* = *item\_name*** assignments, separated by commas. Each assignment names a target variable and either a ***statement\_information\_item\_name*** or ***condition\_information\_item\_name*** designator, depending on whether the statement retrieves statement or condition information.

Valid ***target*** designators for storing item information can be stored procedure or function parameters, stored program local variables declared with [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare), or user-defined variables.

Valid ***condition\_number*** designators can be stored procedure or function parameters, stored program local variables declared with [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare), user-defined variables, system variables, or literals. A character literal may include a ***\_charset*** introducer. A warning occurs if the condition number is not in the range from 1 to the number of condition areas that have information. In this case, the warning is added to the diagnostics area without clearing it.

When a condition occurs, MySQL does not populate all condition items recognized by [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics). For example:

mysql> **GET DIAGNOSTICS CONDITION 1**

**@p5 = SCHEMA\_NAME, @p6 = TABLE\_NAME;**

mysql> **SELECT @p5, @p6;**

+------+------+

| @p5 | @p6 |

+------+------+

| | |

+------+------+

In standard SQL, if there are multiple conditions, the first condition relates to the **SQLSTATE** value returned for the previous SQL statement. In MySQL, this is not guaranteed. To get the main error, you cannot do this:

GET DIAGNOSTICS CONDITION 1 @errno = MYSQL\_ERRNO;

Instead, retrieve the condition count first, then use it to specify which condition number to inspect:

GET DIAGNOSTICS @cno = NUMBER;

GET DIAGNOSTICS CONDITION @cno @errno = MYSQL\_ERRNO;

For information about permissible statement and condition information items, and which ones are populated when a condition occurs, see [Diagnostics Area Information Items](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-information-items).

Here is an example that uses [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) and an exception handler in stored procedure context to assess the outcome of an insert operation. If the insert was successful, the procedure uses [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) to get the rows-affected count. This shows that you can use [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) multiple times to retrieve information about a statement as long as the current diagnostics area has not been cleared.

CREATE PROCEDURE do\_insert(value INT)

BEGIN

-- Declare variables to hold diagnostics area information

DECLARE code CHAR(5) DEFAULT '00000';

DECLARE msg TEXT;

DECLARE nrows INT;

DECLARE result TEXT;

-- Declare exception handler for failed insert

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

BEGIN

GET DIAGNOSTICS CONDITION 1

code = RETURNED\_SQLSTATE, msg = MESSAGE\_TEXT;

END;

-- Perform the insert

INSERT INTO t1 (int\_col) VALUES(value);

-- Check whether the insert was successful

IF code = '00000' THEN

GET DIAGNOSTICS nrows = ROW\_COUNT;

SET result = CONCAT('insert succeeded, row count = ',nrows);

ELSE

SET result = CONCAT('insert failed, error = ',code,', message = ',msg);

END IF;

-- Say what happened

SELECT result;

END;

Suppose that **t1.int\_col** is an integer column that is declared as **NOT NULL**. The procedure produces these results when invoked to insert non-**NULL** and **NULL** values, respectively:

mysql> **CALL do\_insert(1);**

+---------------------------------+

| result |

+---------------------------------+

| insert succeeded, row count = 1 |

+---------------------------------+

mysql> **CALL do\_insert(NULL);**

+-------------------------------------------------------------------------+

| result |

+-------------------------------------------------------------------------+

| insert failed, error = 23000, message = Column 'int\_col' cannot be null |

+-------------------------------------------------------------------------+

When a condition handler activates, a push to the diagnostics area stack occurs:

The first (current) diagnostics area becomes the second (stacked) diagnostics area and a new current diagnostics area is created as a copy of it.

[**GET [CURRENT] DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) and [**GET STACKED DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) can be used within the handler to access the contents of the current and stacked diagnostics areas.

Initially, both diagnostics areas return the same result, so it is possible to get information from the current diagnostics area about the condition that activated the handler, as long as you execute no statements within the handler that change its current diagnostics area.

However, statements executing within the handler can modify the current diagnostics area, clearing and setting its contents according to the normal rules (see [How the Diagnostics Area is Cleared and Populated](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-populating)).

A more reliable way to obtain information about the handler-activating condition is to use the stacked diagnostics area, which cannot be modified by statements executing within the handler except [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal). For information about when the current diagnostics area is set and cleared, see [Section 13.6.7.7, “The MySQL Diagnostics Area”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area).

The next example shows how **GET STACKED DIAGNOSTICS** can be used within a handler to obtain information about the handled exception, even after the current diagnostics area has been modified by handler statements.

Within a stored procedure **p()**, we attempt to insert two values into a table that contains a **TEXT NOT NULL** column. The first value is a non-**NULL** string and the second is **NULL**. The column prohibits **NULL** values, so the first insert succeeds but the second causes an exception. The procedure includes an exception handler that maps attempts to insert **NULL** into inserts of the empty string:

DROP TABLE IF EXISTS t1;

CREATE TABLE t1 (c1 TEXT NOT NULL);

DROP PROCEDURE IF EXISTS p;

delimiter //

CREATE PROCEDURE p ()

BEGIN

-- Declare variables to hold diagnostics area information

DECLARE errcount INT;

DECLARE errno INT;

DECLARE msg TEXT;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

-- Here the current DA is nonempty because no prior statements

-- executing within the handler have cleared it

GET CURRENT DIAGNOSTICS CONDITION 1

errno = MYSQL\_ERRNO, msg = MESSAGE\_TEXT;

SELECT 'current DA before mapped insert' AS op, errno, msg;

GET STACKED DIAGNOSTICS CONDITION 1

errno = MYSQL\_ERRNO, msg = MESSAGE\_TEXT;

SELECT 'stacked DA before mapped insert' AS op, errno, msg;

-- Map attempted NULL insert to empty string insert

INSERT INTO t1 (c1) VALUES('');

-- Here the current DA should be empty (if the INSERT succeeded),

-- so check whether there are conditions before attempting to

-- obtain condition information

GET CURRENT DIAGNOSTICS errcount = NUMBER;

IF errcount = 0

THEN

SELECT 'mapped insert succeeded, current DA is empty' AS op;

ELSE

GET CURRENT DIAGNOSTICS CONDITION 1

errno = MYSQL\_ERRNO, msg = MESSAGE\_TEXT;

SELECT 'current DA after mapped insert' AS op, errno, msg;

END IF ;

GET STACKED DIAGNOSTICS CONDITION 1

errno = MYSQL\_ERRNO, msg = MESSAGE\_TEXT;

SELECT 'stacked DA after mapped insert' AS op, errno, msg;

END;

INSERT INTO t1 (c1) VALUES('string 1');

INSERT INTO t1 (c1) VALUES(NULL);

END;

//

delimiter ;

CALL p();

SELECT \* FROM t1;

When the handler activates, a copy of the current diagnostics area is pushed to the diagnostics area stack. The handler first displays the contents of the current and stacked diagnostics areas, which are both the same initially:

+---------------------------------+-------+----------------------------+

| op | errno | msg |

+---------------------------------+-------+----------------------------+

| current DA before mapped insert | 1048 | Column 'c1' cannot be null |

+---------------------------------+-------+----------------------------+

+---------------------------------+-------+----------------------------+

| op | errno | msg |

+---------------------------------+-------+----------------------------+

| stacked DA before mapped insert | 1048 | Column 'c1' cannot be null |

+---------------------------------+-------+----------------------------+

Statements executing after the [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statements may reset the current diagnostics area. statements may reset the current diagnostics area. For example, the handler maps the **NULL** insert to an empty-string insert and displays the result. The new insert succeeds and clears the current diagnostics area, but the stacked diagnostics area remains unchanged and still contains information about the condition that activated the handler:

+----------------------------------------------+

| op |

+----------------------------------------------+

| mapped insert succeeded, current DA is empty |

+----------------------------------------------+

+--------------------------------+-------+----------------------------+

| op | errno | msg |

+--------------------------------+-------+----------------------------+

| stacked DA after mapped insert | 1048 | Column 'c1' cannot be null |

+--------------------------------+-------+----------------------------+

When the condition handler ends, its current diagnostics area is popped from the stack and the stacked diagnostics area becomes the current diagnostics area in the stored procedure.

After the procedure returns, the table contains two rows. The empty row results from the attempt to insert **NULL** that was mapped to an empty-string insert:

+----------+

| c1 |

+----------+

| string 1 |

| |

+----------+

In the preceding example, the first two [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statements within the condition handler that retrieve information from the current and stacked diagnostics areas return the same values. This is not the case if statements that reset the current diagnostics area execute earlier within the handler. Suppose that **p()** is rewritten to place the [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare) statements within the handler definition rather than preceding it:

CREATE PROCEDURE p ()

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

-- Declare variables to hold diagnostics area information

DECLARE errcount INT;

DECLARE errno INT;

DECLARE msg TEXT;

GET CURRENT DIAGNOSTICS CONDITION 1

errno = MYSQL\_ERRNO, msg = MESSAGE\_TEXT;

SELECT 'current DA before mapped insert' AS op, errno, msg;

GET STACKED DIAGNOSTICS CONDITION 1

errno = MYSQL\_ERRNO, msg = MESSAGE\_TEXT;

SELECT 'stacked DA before mapped insert' AS op, errno, msg;

...

In this case, the result is version dependent:

Before MySQL 5.7.2, [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare) does not change the current diagnostics area, so the first two [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statements return the same result, just as in the original version of **p()**.

In MySQL 5.7.2, work was done to ensure that all nondiagnostic statements populate the diagnostics area, per the SQL standard. [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare) is one of them, so in 5.7.2 and higher, [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare) statements executing at the beginning of the handler clear the current diagnostics area and the [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statements produce different results:

+---------------------------------+-------+------+

| op | errno | msg |

+---------------------------------+-------+------+

| current DA before mapped insert | NULL | NULL |

+---------------------------------+-------+------+

+---------------------------------+-------+----------------------------+

| op | errno | msg |

+---------------------------------+-------+----------------------------+

| stacked DA before mapped insert | 1048 | Column 'c1' cannot be null |

+---------------------------------+-------+----------------------------+

To avoid this issue within a condition handler when seeking to obtain information about the condition that activated the handler, be sure to access the stacked diagnostics area, not the current diagnostics area.

#### 13.6.7.4 RESIGNAL Statement

RESIGNAL [***condition\_value***]

[SET ***signal\_information\_item***

[, ***signal\_information\_item***] ...]

***condition\_value***: {

SQLSTATE [VALUE] ***sqlstate\_value***

| ***condition\_name***

}

***signal\_information\_item***:

***condition\_information\_item\_name*** = ***simple\_value\_specification***

***condition\_information\_item\_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

}

***condition\_name***, ***simple\_value\_specification***:

(see following discussion)

[**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) passes on the error condition information that is available during execution of a condition handler within a compound statement inside a stored procedure or function, trigger, or event. [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) may change some or all information before passing it on. [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) is related to [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal), but instead of originating a condition as [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) does, [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) relays existing condition information, possibly after modifying it.

[**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) makes it possible to both handle an error and return the error information. Otherwise, by executing an SQL statement within the handler, information that caused the handler's activation is destroyed. [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) also can make some procedures shorter if a given handler can handle part of a situation, then pass the condition “up the line” to another handler.

No privileges are required to execute the [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statement.

All forms of [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) require that the current context be a condition handler. Otherwise, [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) is illegal and a **RESIGNAL when handler not active** error occurs.

To retrieve information from the diagnostics area, use the [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statement (see [Section 13.6.7.3, “GET DIAGNOSTICS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics)). For information about the diagnostics area, see [Section 13.6.7.7, “The MySQL Diagnostics Area”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area).

[RESIGNAL Overview](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal-overview)

[RESIGNAL Alone](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal-alone)

[RESIGNAL with New Signal Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal-with-new-signal)

[RESIGNAL with a Condition Value and Optional New Signal Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal-with-condition)

[RESIGNAL Requires Condition Handler Context](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal-handler)

##### RESIGNAL Overview

For ***condition\_value*** and ***signal\_information\_item***, the definitions and rules are the same for [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) as for [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal). For example, the ***condition\_value*** can be an **SQLSTATE** value, and the value can indicate errors, warnings, or “not found.” For additional information, see [Section 13.6.7.5, “SIGNAL Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal).

The [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statement takes ***condition\_value*** and **SET** clauses, both of which are optional. This leads to several possible uses:

[**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) alone:

RESIGNAL;

[**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) with new signal information:

RESIGNAL SET ***signal\_information\_item*** [, ***signal\_information\_item***] ...;

[**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) with a condition value and possibly new signal information:

RESIGNAL ***condition\_value***

[SET ***signal\_information\_item*** [, ***signal\_information\_item***] ...];

These use cases all cause changes to the diagnostics and condition areas:

A diagnostics area contains one or more condition areas.

A condition area contains condition information items, such as the **SQLSTATE** value, **MYSQL\_ERRNO**, or **MESSAGE\_TEXT**.

There is a stack of diagnostics areas. When a handler takes control, it pushes a diagnostics area to the top of the stack, so there are two diagnostics areas during handler execution:

The first (current) diagnostics area, which starts as a copy of the last diagnostics area, but is overwritten by the first statement in the handler that changes the current diagnostics area.

The last (stacked) diagnostics area, which has the condition areas that were set up before the handler took control.

The maximum number of condition areas in a diagnostics area is determined by the value of the [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) system variable. See [Diagnostics Area-Related System Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-system-variables).

##### RESIGNAL Alone

A simple [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) alone means “pass on the error with no change.” It restores the last diagnostics area and makes it the current diagnostics area. That is, it “pops” the diagnostics area stack.

Within a condition handler that catches a condition, one use for [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) alone is to perform some other actions, and then pass on without change the original condition information (the information that existed before entry into the handler).

Example:

DROP TABLE IF EXISTS xx;

delimiter //

CREATE PROCEDURE p ()

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

SET @error\_count = @error\_count + 1;

IF @a = 0 THEN RESIGNAL; END IF;

END;

DROP TABLE xx;

END//

delimiter ;

SET @error\_count = 0;

SET @a = 0;

CALL p();

Suppose that the **DROP TABLE xx** statement fails. The diagnostics area stack looks like this:

DA 1. ERROR 1051 (42S02): Unknown table 'xx'

Then execution enters the **EXIT** handler. It starts by pushing a diagnostics area to the top of the stack, which now looks like this:

DA 1. ERROR 1051 (42S02): Unknown table 'xx'

DA 2. ERROR 1051 (42S02): Unknown table 'xx'

At this point, the contents of the first (current) and second (stacked) diagnostics areas are the same. The first diagnostics area may be modified by statements executing subsequently within the handler.

Usually a procedure statement clears the first diagnostics area. **BEGIN** is an exception, it does not clear, it does nothing. **SET** is not an exception, it clears, performs the operation, and produces a result of “success.” The diagnostics area stack now looks like this:

DA 1. ERROR 0000 (00000): Successful operation

DA 2. ERROR 1051 (42S02): Unknown table 'xx'

At this point, if **@a = 0**, [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) pops the diagnostics area stack, which now looks like this:

DA 1. ERROR 1051 (42S02): Unknown table 'xx'

And that is what the caller sees.

If **@a** is not 0, the handler simply ends, which means that there is no more use for the current diagnostics area (it has been “handled”), so it can be thrown away, causing the stacked diagnostics area to become the current diagnostics area again. The diagnostics area stack looks like this:

DA 1. ERROR 0000 (00000): Successful operation

The details make it look complex, but the end result is quite useful: Handlers can execute without destroying information about the condition that caused activation of the handler.

##### RESIGNAL with New Signal Information

[**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) with a **SET** clause provides new signal information, so the statement means “pass on the error with changes”:

RESIGNAL SET ***signal\_information\_item*** [, ***signal\_information\_item***] ...;

As with [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) alone, the idea is to pop the diagnostics area stack so that the original information goes out. Unlike [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) alone, anything specified in the **SET** clause changes.

Example:

DROP TABLE IF EXISTS xx;

delimiter //

CREATE PROCEDURE p ()

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

SET @error\_count = @error\_count + 1;

IF @a = 0 THEN RESIGNAL SET MYSQL\_ERRNO = 5; END IF;

END;

DROP TABLE xx;

END//

delimiter ;

SET @error\_count = 0;

SET @a = 0;

CALL p();

Remember from the previous discussion that [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) alone results in a diagnostics area stack like this:

DA 1. ERROR 1051 (42S02): Unknown table 'xx'

The **RESIGNAL SET MYSQL\_ERRNO = 5** statement results in this stack instead, which is what the caller sees:

DA 1. ERROR 5 (42S02): Unknown table 'xx'

In other words, it changes the error number, and nothing else.

The [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statement can change any or all of the signal information items, making the first condition area of the diagnostics area look quite different.

##### RESIGNAL with a Condition Value and Optional New Signal Information

[**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) with a condition value means “push a condition into the current diagnostics area.” If the **SET** clause is present, it also changes the error information.

RESIGNAL ***condition\_value***

[SET ***signal\_information\_item*** [, ***signal\_information\_item***] ...];

This form of [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) restores the last diagnostics area and makes it the current diagnostics area. That is, it “pops” the diagnostics area stack, which is the same as what a simple [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) alone would do. However, it also changes the diagnostics area depending on the condition value or signal information.

Example:

DROP TABLE IF EXISTS xx;

delimiter //

CREATE PROCEDURE p ()

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

SET @error\_count = @error\_count + 1;

IF @a = 0 THEN RESIGNAL SQLSTATE '45000' SET MYSQL\_ERRNO=5; END IF;

END;

DROP TABLE xx;

END//

delimiter ;

SET @error\_count = 0;

SET @a = 0;

SET @@max\_error\_count = 2;

CALL p();

SHOW ERRORS;

This is similar to the previous example, and the effects are the same, except that if [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) happens, the current condition area looks different at the end. (The reason the condition adds to rather than replaces the existing condition is the use of a condition value.)

The [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statement includes a condition value (**SQLSTATE '45000'**), so it adds a new condition area, resulting in a diagnostics area stack that looks like this:

DA 1. (condition 2) ERROR 1051 (42S02): Unknown table 'xx'

(condition 1) ERROR 5 (45000) Unknown table 'xx'

The result of [**CALL p()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#call) and [**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors) for this example is:

mysql> **CALL p();**

ERROR 5 (45000): Unknown table 'xx'

mysql> **SHOW ERRORS;**

+-------+------+----------------------------------+

| Level | Code | Message |

+-------+------+----------------------------------+

| Error | 1051 | Unknown table 'xx' |

| Error | 5 | Unknown table 'xx' |

+-------+------+----------------------------------+

##### RESIGNAL Requires Condition Handler Context

All forms of [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) require that the current context be a condition handler. Otherwise, [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) is illegal and a **RESIGNAL when handler not active** error occurs. For example:

mysql> **CREATE PROCEDURE p () RESIGNAL;**

Query OK, 0 rows affected (0.00 sec)

mysql> **CALL p();**

ERROR 1645 (0K000): RESIGNAL when handler not active

Here is a more difficult example:

delimiter //

CREATE FUNCTION f () RETURNS INT

BEGIN

RESIGNAL;

RETURN 5;

END//

CREATE PROCEDURE p ()

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION SET @a=f();

SIGNAL SQLSTATE '55555';

END//

delimiter ;

CALL p();

[**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) occurs within the stored function **f()**. Although **f()** itself is invoked within the context of the **EXIT** handler, execution within **f()** has its own context, which is not handler context. Thus, **RESIGNAL** within **f()** results in a “handler not active” error.

#### 13.6.7.5 SIGNAL Statement

SIGNAL ***condition\_value***

[SET ***signal\_information\_item***

[, ***signal\_information\_item***] ...]

***condition\_value***: {

SQLSTATE [VALUE] ***sqlstate\_value***

| ***condition\_name***

}

***signal\_information\_item***:

***condition\_information\_item\_name*** = ***simple\_value\_specification***

***condition\_information\_item\_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

}

***condition\_name***, ***simple\_value\_specification***:

(see following discussion)

[**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) is the way to “return” an error. [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) provides error information to a handler, to an outer portion of the application, or to the client. Also, it provides control over the error's characteristics (error number, **SQLSTATE** value, message). Without [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal), it is necessary to resort to workarounds such as deliberately referring to a nonexistent table to cause a routine to return an error.

No privileges are required to execute the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement.

To retrieve information from the diagnostics area, use the [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statement (see [Section 13.6.7.3, “GET DIAGNOSTICS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics)). For information about the diagnostics area, see [Section 13.6.7.7, “The MySQL Diagnostics Area”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area).

[SIGNAL Overview](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal-overview)

[Signal Condition Information Items](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal-condition-information-items)

[Effect of Signals on Handlers, Cursors, and Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal-effects)

##### SIGNAL Overview

The ***condition\_value*** in a [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement indicates the error value to be returned. It can be an **SQLSTATE** value (a 5-character string literal) or a ***condition\_name*** that refers to a named condition previously defined with [**DECLARE ... CONDITION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition) (see [Section 13.6.7.1, “DECLARE ... CONDITION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-condition)).

An **SQLSTATE** value can indicate errors, warnings, or “not found.” The first two characters of the value indicate its error class, as discussed in [Signal Condition Information Items](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal-condition-information-items). Some signal values cause statement termination; see [Effect of Signals on Handlers, Cursors, and Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal-effects).

The **SQLSTATE** value for a [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement should not start with **'00'** because such values indicate success and are not valid for signaling an error. This is true whether the **SQLSTATE** value is specified directly in the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement or in a named condition referred to in the statement. If the value is invalid, a **Bad SQLSTATE** error occurs.

To signal a generic **SQLSTATE** value, use **'45000'**, which means “unhandled user-defined exception.”

The [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement optionally includes a **SET** clause that contains multiple signal items, in a list of ***condition\_information\_item\_name*** = ***simple\_value\_specification*** assignments, separated by commas.

Each ***condition\_information\_item\_name*** may be specified only once in the **SET** clause. Otherwise, a **Duplicate condition information item** error occurs.

Valid ***simple\_value\_specification*** designators can be specified using stored procedure or function parameters, stored program local variables declared with [**DECLARE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare), user-defined variables, system variables, or literals. A character literal may include a ***\_charset*** introducer.

For information about permissible ***condition\_information\_item\_name*** values, see [Signal Condition Information Items](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal-condition-information-items).

The following procedure signals an error or warning depending on the value of **pval**, its input parameter:

CREATE PROCEDURE p (pval INT)

BEGIN

DECLARE specialty CONDITION FOR SQLSTATE '45000';

IF pval = 0 THEN

SIGNAL SQLSTATE '01000';

ELSEIF pval = 1 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'An error occurred';

ELSEIF pval = 2 THEN

SIGNAL specialty

SET MESSAGE\_TEXT = 'An error occurred';

ELSE

SIGNAL SQLSTATE '01000'

SET MESSAGE\_TEXT = 'A warning occurred', MYSQL\_ERRNO = 1000;

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'An error occurred', MYSQL\_ERRNO = 1001;

END IF;

END;

If **pval** is 0, **p()** signals a warning because **SQLSTATE** values that begin with **'01'** are signals in the warning class. The warning does not terminate the procedure, and can be seen with [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) after the procedure returns.

If **pval** is 1, **p()** signals an error and sets the **MESSAGE\_TEXT** condition information item. The error terminates the procedure, and the text is returned with the error information.

If **pval** is 2, the same error is signaled, although the **SQLSTATE** value is specified using a named condition in this case.

If **pval** is anything else, **p()** first signals a warning and sets the message text and error number condition information items. This warning does not terminate the procedure, so execution continues and **p()** then signals an error. The error does terminate the procedure. The message text and error number set by the warning are replaced by the values set by the error, which are returned with the error information.

[**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) is typically used within stored programs, but it is a MySQL extension that it is permitted outside handler context. For example, if you invoke the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client program, you can enter any of these statements at the prompt:

SIGNAL SQLSTATE '77777';

CREATE TRIGGER t\_bi BEFORE INSERT ON t

FOR EACH ROW SIGNAL SQLSTATE '77777';

CREATE EVENT e ON SCHEDULE EVERY 1 SECOND

DO SIGNAL SQLSTATE '77777';

[**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) executes according to the following rules:

If the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement indicates a particular **SQLSTATE** value, that value is used to signal the condition specified. Example:

CREATE PROCEDURE p (divisor INT)

BEGIN

IF divisor = 0 THEN

SIGNAL SQLSTATE '22012';

END IF;

END;

If the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement uses a named condition, the condition must be declared in some scope that applies to the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement, and must be defined using an **SQLSTATE** value, not a MySQL error number. Example:

CREATE PROCEDURE p (divisor INT)

BEGIN

DECLARE divide\_by\_zero CONDITION FOR SQLSTATE '22012';

IF divisor = 0 THEN

SIGNAL divide\_by\_zero;

END IF;

END;

If the named condition does not exist in the scope of the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement, an **Undefined CONDITION** error occurs.

If [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) refers to a named condition that is defined with a MySQL error number rather than an **SQLSTATE** value, a **SIGNAL/RESIGNAL can only use a CONDITION defined with SQLSTATE** error occurs. The following statements cause that error because the named condition is associated with a MySQL error number:

DECLARE no\_such\_table CONDITION FOR 1051;

SIGNAL no\_such\_table;

If a condition with a given name is declared multiple times in different scopes, the declaration with the most local scope applies. Consider the following procedure:

CREATE PROCEDURE p (divisor INT)

BEGIN

DECLARE my\_error CONDITION FOR SQLSTATE '45000';

IF divisor = 0 THEN

BEGIN

DECLARE my\_error CONDITION FOR SQLSTATE '22012';

SIGNAL my\_error;

END;

END IF;

SIGNAL my\_error;

END;

If **divisor** is 0, the first [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement executes. The innermost **my\_error** condition declaration applies, raising **SQLSTATE** **'22012'**.

If **divisor** is not 0, the second [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement executes. The outermost **my\_error** condition declaration applies, raising **SQLSTATE** **'45000'**.

For information about how the server chooses handlers when a condition occurs, see [Section 13.6.7.6, “Scope Rules for Handlers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler-scope).

Signals can be raised within exception handlers:

CREATE PROCEDURE p ()

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

SIGNAL SQLSTATE VALUE '99999'

SET MESSAGE\_TEXT = 'An error occurred';

END;

DROP TABLE no\_such\_table;

END;

**CALL p()** reaches the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement. There is no table named **no\_such\_table**, so the error handler is activated. The error handler destroys the original error (“no such table”) and makes a new error with **SQLSTATE** **'99999'** and message **An error occurred**.

##### Signal Condition Information Items

The following table lists the names of diagnostics area condition information items that can be set in a [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) (or [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal)) statement. All items are standard SQL except **MYSQL\_ERRNO**, which is a MySQL extension. For more information about these items see [Section 13.6.7.7, “The MySQL Diagnostics Area”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area).

Item Name Definition

--------- ----------

CLASS\_ORIGIN VARCHAR(64)

SUBCLASS\_ORIGIN VARCHAR(64)

CONSTRAINT\_CATALOG VARCHAR(64)

CONSTRAINT\_SCHEMA VARCHAR(64)

CONSTRAINT\_NAME VARCHAR(64)

CATALOG\_NAME VARCHAR(64)

SCHEMA\_NAME VARCHAR(64)

TABLE\_NAME VARCHAR(64)

COLUMN\_NAME VARCHAR(64)

CURSOR\_NAME VARCHAR(64)

MESSAGE\_TEXT VARCHAR(128)

MYSQL\_ERRNO SMALLINT UNSIGNED

The character set for character items is UTF-8.

It is illegal to assign **NULL** to a condition information item in a [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement.

A [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement always specifies an **SQLSTATE** value, either directly, or indirectly by referring to a named condition defined with an **SQLSTATE** value. The first two characters of an **SQLSTATE** value are its class, and the class determines the default value for the condition information items:

Class = **'00'** (success)

Illegal. **SQLSTATE** values that begin with **'00'** indicate success and are not valid for [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal).

Class = **'01'** (warning)

MESSAGE\_TEXT = 'Unhandled user-defined warning condition';

MYSQL\_ERRNO = [**ER\_SIGNAL\_WARN**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_signal_warn)

Class = **'02'** (not found)

MESSAGE\_TEXT = 'Unhandled user-defined not found condition';

MYSQL\_ERRNO = [**ER\_SIGNAL\_NOT\_FOUND**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_signal_not_found)

Class > **'02'** (exception)

MESSAGE\_TEXT = 'Unhandled user-defined exception condition';

MYSQL\_ERRNO = [**ER\_SIGNAL\_EXCEPTION**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_signal_exception)

For legal classes, the other condition information items are set as follows:

CLASS\_ORIGIN = SUBCLASS\_ORIGIN = '';

CONSTRAINT\_CATALOG = CONSTRAINT\_SCHEMA = CONSTRAINT\_NAME = '';

CATALOG\_NAME = SCHEMA\_NAME = TABLE\_NAME = COLUMN\_NAME = '';

CURSOR\_NAME = '';

The error values that are accessible after [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) executes are the **SQLSTATE** value raised by the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement and the **MESSAGE\_TEXT** and **MYSQL\_ERRNO** items. These values are available from the C API:

[**mysql\_sqlstate()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-sqlstate.html) returns the **SQLSTATE** value.

[**mysql\_errno()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-errno.html) returns the **MYSQL\_ERRNO** value.

[**mysql\_error()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-error.html) returns the **MESSAGE\_TEXT** value.

At the SQL level, the output from [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) and [**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors) indicates the **MYSQL\_ERRNO** and **MESSAGE\_TEXT** values in the **Code** and **Message** columns.

To retrieve information from the diagnostics area, use the [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statement (see [Section 13.6.7.3, “GET DIAGNOSTICS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics)). For information about the diagnostics area, see [Section 13.6.7.7, “The MySQL Diagnostics Area”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area).

##### Effect of Signals on Handlers, Cursors, and Statements

Signals have different effects on statement execution depending on the signal class. The class determines how severe an error is. MySQL ignores the value of the [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) system variable; in particular, strict SQL mode does not matter. MySQL also ignores **IGNORE**: The intent of [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) is to raise a user-generated error explicitly, so a signal is never ignored.

In the following descriptions, “unhandled” means that no handler for the signaled **SQLSTATE** value has been defined with [**DECLARE ... HANDLER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#declare-handler).

Class = **'00'** (success)

Illegal. **SQLSTATE** values that begin with **'00'** indicate success and are not valid for [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal).

Class = **'01'** (warning)

The value of the [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) system variable goes up. [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) shows the signal. **SQLWARNING** handlers catch the signal.

Warnings cannot be returned from stored functions because the [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) statement that causes the function to return clears the diagnostic area. The statement thus clears any warnings that may have been present there (and resets [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) to 0).

Class = **'02'** (not found)

**NOT FOUND** handlers catch the signal. There is no effect on cursors. If the signal is unhandled in a stored function, statements end.

Class > **'02'** (exception)

**SQLEXCEPTION** handlers catch the signal. If the signal is unhandled in a stored function, statements end.

Class = **'40'**

Treated as an ordinary exception.

#### 13.6.7.6 Scope Rules for Handlers

A stored program may include handlers to be invoked when certain conditions occur within the program. The applicability of each handler depends on its location within the program definition and on the condition or conditions that it handles:

A handler declared in a [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block is in scope only for the SQL statements following the handler declarations in the block. If the handler itself raises a condition, it cannot handle that condition, nor can any other handlers declared in the block. In the following example, handlers **H1** and **H2** are in scope for conditions raised by statements ***stmt1*** and ***stmt2***. But neither **H1** nor **H2** are in scope for conditions raised in the body of **H1** or **H2**.

BEGIN -- outer block

DECLARE EXIT HANDLER FOR ...; -- handler H1

DECLARE EXIT HANDLER FOR ...; -- handler H2

***stmt1***;

***stmt2***;

END;

A handler is in scope only for the block in which it is declared, and cannot be activated for conditions occurring outside that block. In the following example, handler **H1** is in scope for ***stmt1*** in the inner block, but not for ***stmt2*** in the outer block:

BEGIN -- outer block

BEGIN -- inner block

DECLARE EXIT HANDLER FOR ...; -- handler H1

***stmt1***;

END;

***stmt2***;

END;

A handler can be specific or general. A specific handler is for a MySQL error code, **SQLSTATE** value, or condition name. A general handler is for a condition in the **SQLWARNING**, **SQLEXCEPTION**, or **NOT FOUND** class. Condition specificity is related to condition precedence, as described later.

Multiple handlers can be declared in different scopes and with different specificities. For example, there might be a specific MySQL error code handler in an outer block, and a general **SQLWARNING** handler in an inner block. Or there might be handlers for a specific MySQL error code and the general **SQLWARNING** class in the same block.

Whether a handler is activated depends not only on its own scope and condition value, but on what other handlers are present. When a condition occurs in a stored program, the server searches for applicable handlers in the current scope (current [**BEGIN ... END**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#begin-end) block). If there are no applicable handlers, the search continues outward with the handlers in each successive containing scope (block). When the server finds one or more applicable handlers at a given scope, it chooses among them based on condition precedence:

A MySQL error code handler takes precedence over an **SQLSTATE** value handler.

An **SQLSTATE** value handler takes precedence over general **SQLWARNING**, **SQLEXCEPTION**, or **NOT FOUND** handlers.

An **SQLEXCEPTION** handler takes precedence over an **SQLWARNING** handler.

It is possible to have several applicable handlers with the same precedence. For example, a statement could generate multiple warnings with different error codes, for each of which an error-specific handler exists. In this case, the choice of which handler the server activates is nondeterministic, and may change depending on the circumstances under which the condition occurs.

One implication of the handler selection rules is that if multiple applicable handlers occur in different scopes, handlers with the most local scope take precedence over handlers in outer scopes, even over those for more specific conditions.

If there is no appropriate handler when a condition occurs, the action taken depends on the class of the condition:

For **SQLEXCEPTION** conditions, the stored program terminates at the statement that raised the condition, as if there were an **EXIT** handler. If the program was called by another stored program, the calling program handles the condition using the handler selection rules applied to its own handlers.

For **SQLWARNING** conditions, the program continues executing, as if there were a **CONTINUE** handler.

For **NOT FOUND** conditions, if the condition was raised normally, the action is **CONTINUE**. If it was raised by [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) or [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal), the action is **EXIT**.

The following examples demonstrate how MySQL applies the handler selection rules.

This procedure contains two handlers, one for the specific **SQLSTATE** value (**'42S02'**) that occurs for attempts to drop a nonexistent table, and one for the general **SQLEXCEPTION** class:

CREATE PROCEDURE p1()

BEGIN

DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'

SELECT 'SQLSTATE handler was activated' AS msg;

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

SELECT 'SQLEXCEPTION handler was activated' AS msg;

DROP TABLE test.t;

END;

Both handlers are declared in the same block and have the same scope. However, **SQLSTATE** handlers take precedence over **SQLEXCEPTION** handlers, so if the table **t** is nonexistent, the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement raises a condition that activates the **SQLSTATE** handler:

mysql> **CALL p1();**

+--------------------------------+

| msg |

+--------------------------------+

| SQLSTATE handler was activated |

+--------------------------------+

This procedure contains the same two handlers. But this time, the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement and **SQLEXCEPTION** handler are in an inner block relative to the **SQLSTATE** handler:

CREATE PROCEDURE p2()

BEGIN -- outer block

DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'

SELECT 'SQLSTATE handler was activated' AS msg;

BEGIN -- inner block

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

SELECT 'SQLEXCEPTION handler was activated' AS msg;

DROP TABLE test.t; -- occurs within inner block

END;

END;

In this case, the handler that is more local to where the condition occurs takes precedence. The **SQLEXCEPTION** handler activates, even though it is more general than the **SQLSTATE** handler:

mysql> **CALL p2();**

+------------------------------------+

| msg |

+------------------------------------+

| SQLEXCEPTION handler was activated |

+------------------------------------+

In this procedure, one of the handlers is declared in a block inner to the scope of the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement:

CREATE PROCEDURE p3()

BEGIN -- outer block

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

SELECT 'SQLEXCEPTION handler was activated' AS msg;

BEGIN -- inner block

DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'

SELECT 'SQLSTATE handler was activated' AS msg;

END;

DROP TABLE test.t; -- occurs within outer block

END;

Only the **SQLEXCEPTION** handler applies because the other one is not in scope for the condition raised by the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table):

mysql> **CALL p3();**

+------------------------------------+

| msg |

+------------------------------------+

| SQLEXCEPTION handler was activated |

+------------------------------------+

In this procedure, both handlers are declared in a block inner to the scope of the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement:

CREATE PROCEDURE p4()

BEGIN -- outer block

BEGIN -- inner block

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

SELECT 'SQLEXCEPTION handler was activated' AS msg;

DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02'

SELECT 'SQLSTATE handler was activated' AS msg;

END;

DROP TABLE test.t; -- occurs within outer block

END;

Neither handler applies because they are not in scope for the [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table). The condition raised by the statement goes unhandled and terminates the procedure with an error:

mysql> **CALL p4();**

ERROR 1051 (42S02): Unknown table 'test.t'

#### 13.6.7.7 The MySQL Diagnostics Area

SQL statements produce diagnostic information that populates the diagnostics area. Standard SQL has a diagnostics area stack, containing a diagnostics area for each nested execution context. Standard SQL also supports [**GET STACKED DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) syntax for referring to the second diagnostics area during condition handler execution.

The following discussion describes the structure of the diagnostics area in MySQL, the information items recognized by MySQL, how statements clear and set the diagnostics area, and how diagnostics areas are pushed to and popped from the stack.

[Diagnostics Area Structure](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-structure)

[Diagnostics Area Information Items](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-information-items)

[How the Diagnostics Area is Cleared and Populated](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-populating)

[How the Diagnostics Area Stack Works](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-stack)

[Diagnostics Area-Related System Variables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-system-variables)

##### Diagnostics Area Structure

The diagnostics area contains two kinds of information:

Statement information, such as the number of conditions that occurred or the affected-rows count.

Condition information, such as the error code and message. If a statement raises multiple conditions, this part of the diagnostics area has a condition area for each one. If a statement raises no conditions, this part of the diagnostics area is empty.

For a statement that produces three conditions, the diagnostics area contains statement and condition information like this:

Statement information:

row count

... other statement information items ...

Condition area list:

Condition area 1:

error code for condition 1

error message for condition 1

... other condition information items ...

Condition area 2:

error code for condition 2:

error message for condition 2

... other condition information items ...

Condition area 3:

error code for condition 3

error message for condition 3

... other condition information items ...

##### Diagnostics Area Information Items

The diagnostics area contains statement and condition information items. Numeric items are integers. The character set for character items is UTF-8. No item can be **NULL**. If a statement or condition item is not set by a statement that populates the diagnostics area, its value is 0 or the empty string, depending on the item data type.

The statement information part of the diagnostics area contains these items:

**NUMBER**: An integer indicating the number of condition areas that have information.

**ROW\_COUNT**: An integer indicating the number of rows affected by the statement. **ROW\_COUNT** has the same value as the [**ROW\_COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_row-count) function (see [Section 12.16, “Information Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#information-functions)).

The condition information part of the diagnostics area contains a condition area for each condition. Condition areas are numbered from 1 to the value of the **NUMBER** statement condition item. If **NUMBER** is 0, there are no condition areas.

Each condition area contains the items in the following list. All items are standard SQL except **MYSQL\_ERRNO**, which is a MySQL extension. The definitions apply for conditions generated other than by a signal (that is, by a [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) or [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statement). For nonsignal conditions, MySQL populates only those condition items not described as always empty. The effects of signals on the condition area are described later.

**CLASS\_ORIGIN**: A string containing the class of the **RETURNED\_SQLSTATE** value. If the **RETURNED\_SQLSTATE** value begins with a class value defined in SQL standards document ISO 9075-2 (section 24.1, SQLSTATE), **CLASS\_ORIGIN** is **'ISO 9075'**. Otherwise, **CLASS\_ORIGIN** is **'MySQL'**.

**SUBCLASS\_ORIGIN**: A string containing the subclass of the **RETURNED\_SQLSTATE** value. If **CLASS\_ORIGIN** is **'ISO 9075'** or **RETURNED\_SQLSTATE** ends with **'000'**, **SUBCLASS\_ORIGIN** is **'ISO 9075'**. Otherwise, **SUBCLASS\_ORIGIN** is **'MySQL'**.

**RETURNED\_SQLSTATE**: A string that indicates the **SQLSTATE** value for the condition.

**MESSAGE\_TEXT**: A string that indicates the error message for the condition.

**MYSQL\_ERRNO**: An integer that indicates the MySQL error code for the condition.

**CONSTRAINT\_CATALOG**, **CONSTRAINT\_SCHEMA**, **CONSTRAINT\_NAME**: Strings that indicate the catalog, schema, and name for a violated constraint. They are always empty.

**CATALOG\_NAME**, **SCHEMA\_NAME**, **TABLE\_NAME**, **COLUMN\_NAME**: Strings that indicate the catalog, schema, table, and column related to the condition. They are always empty.

**CURSOR\_NAME**: A string that indicates the cursor name. This is always empty.

For the **RETURNED\_SQLSTATE**, **MESSAGE\_TEXT**, and **MYSQL\_ERRNO** values for particular errors, see [Server Error Message Reference](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html).

If a [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) (or [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal)) statement populates the diagnostics area, its **SET** clause can assign to any condition information item except **RETURNED\_SQLSTATE** any value that is legal for the item data type. [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) also sets the **RETURNED\_SQLSTATE** value, but not directly in its **SET** clause. That value comes from the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) statement **SQLSTATE** argument.

[**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) also sets statement information items. It sets **NUMBER** to 1. It sets **ROW\_COUNT** to −1 for errors and 0 otherwise.

##### How the Diagnostics Area is Cleared and Populated

Nondiagnostic SQL statements populate the diagnostics area automatically, and its contents can be set explicitly with the [**SIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#signal) and [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statements. The diagnostics area can be examined with [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) to extract specific items, or with [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) or [**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors) to see conditions or errors.

SQL statements clear and set the diagnostics area as follows:

When the server starts executing a statement after parsing it, it clears the diagnostics area for nondiagnostic statements. Diagnostic statements do not clear the diagnostics area. These statements are diagnostic:

[**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics)

[**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors)

[**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings)

If a statement raises a condition, the diagnostics area is cleared of conditions that belong to earlier statements. The exception is that conditions raised by [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) and [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) are added to the diagnostics area without clearing it.

Thus, even a statement that does not normally clear the diagnostics area when it begins executing clears it if the statement raises a condition.

The following example shows the effect of various statements on the diagnostics area, using [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) to display information about conditions stored there.

This [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) statement clears the diagnostics area and populates it when the condition occurs:

mysql> **DROP TABLE IF EXISTS test.no\_such\_table;**

Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> **SHOW WARNINGS;**

+-------+------+------------------------------------+

| Level | Code | Message |

+-------+------+------------------------------------+

| Note | 1051 | Unknown table 'test.no\_such\_table' |

+-------+------+------------------------------------+

1 row in set (0.00 sec)

This [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement generates an error, so it clears and populates the diagnostics area:

mysql> **SET @x = @@x;**

ERROR 1193 (HY000): Unknown system variable 'x'

mysql> **SHOW WARNINGS;**

+-------+------+-----------------------------+

| Level | Code | Message |

+-------+------+-----------------------------+

| Error | 1193 | Unknown system variable 'x' |

+-------+------+-----------------------------+

1 row in set (0.00 sec)

The previous [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement produced a single condition, so 1 is the only valid condition number for [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) at this point. The following statement uses a condition number of 2, which produces a warning that is added to the diagnostics area without clearing it:

mysql> **GET DIAGNOSTICS CONDITION 2 @p = MESSAGE\_TEXT;**

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS;**

+-------+------+------------------------------+

| Level | Code | Message |

+-------+------+------------------------------+

| Error | 1193 | Unknown system variable 'xx' |

| Error | 1753 | Invalid condition number |

+-------+------+------------------------------+

2 rows in set (0.00 sec)

Now there are two conditions in the diagnostics area, so the same [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) statement succeeds:

mysql> **GET DIAGNOSTICS CONDITION 2 @p = MESSAGE\_TEXT;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SELECT @p;**

+--------------------------+

| @p |

+--------------------------+

| Invalid condition number |

+--------------------------+

1 row in set (0.01 sec)

##### How the Diagnostics Area Stack Works

When a push to the diagnostics area stack occurs, the first (current) diagnostics area becomes the second (stacked) diagnostics area and a new current diagnostics area is created as a copy of it. Diagnostics areas are pushed to and popped from the stack under the following circumstances:

Execution of a stored program

A push occurs before the program executes and a pop occurs afterward. If the stored program ends while handlers are executing, there can be more than one diagnostics area to pop; this occurs due to an exception for which there are no appropriate handlers or due to [**RETURN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#return) in the handler.

Any warning or error conditions in the popped diagnostics areas then are added to the current diagnostics area, except that, for triggers, only errors are added. When the stored program ends, the caller sees these conditions in its current diagonstics area.

Execution of a condition handler within a stored program

When a push occurs as a result of condition handler activation, the stacked diagnostics area is the area that was current within the stored program prior to the push. The new now-current diagnostics area is the handler's current diagnostics area. [**GET [CURRENT] DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) and [**GET STACKED DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) can be used within the handler to access the contents of the current (handler) and stacked (stored program) diagnostics areas. Initially, they return the same result, but statements executing within the handler modify the current diagnostics area, clearing and setting its contents according to the normal rules (see [How the Diagnostics Area is Cleared and Populated](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#diagnostics-area-populating)). The stacked diagnostics area cannot be modified by statements executing within the handler except [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal).

If the handler executes successfully, the current (handler) diagnostics area is popped and the stacked (stored program) diagnostics area again becomes the current diagnostics area. Conditions added to the handler diagnostics area during handler execution are added to the current diagnostics area.

Execution of [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal)

The [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statement passes on the error condition information that is available during execution of a condition handler within a compound statement inside a stored program. [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) may change some or all information before passing it on, modifying the diagnostics stack as described in [Section 13.6.7.4, “RESIGNAL Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal).

##### Diagnostics Area-Related System Variables

Certain system variables control or are related to some aspects of the diagnostics area:

[**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) controls the number of condition areas in the diagnostics area. If more conditions than this occur, MySQL silently discards information for the excess conditions. (Conditions added by [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) are always added, with older conditions being discarded as necessary to make room.)

[**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) indicates the number of conditions that occurred. This includes errors, warnings, and notes. Normally, **NUMBER** and [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) are the same. However, as the number of conditions generated exceeds [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count), the value of [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) continues to rise whereas **NUMBER** remains capped at [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) because no additional conditions are stored in the diagnostics area.

[**error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_error_count) indicates the number of errors that occurred. This value includes “not found” and exception conditions, but excludes warnings and notes. Like [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count), its value can exceed [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count).

If the [**sql\_notes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_notes) system variable is set to 0, notes are not stored and do not increment [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count).

Example: If [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) is 10, the diagnostics area can contain a maximum of 10 condition areas. Suppose that a statement raises 20 conditions, 12 of which are errors. In that case, the diagnostics area contains the first 10 conditions, **NUMBER** is 10, [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) is 20, and [**error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_error_count) is 12.

Changes to the value of [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) have no effect until the next attempt to modify the diagnostics area. If the diagnostics area contains 10 condition areas and [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) is set to 5, that has no immediate effect on the size or content of the diagnostics area.

#### 13.6.7.8 Condition Handling and OUT or INOUT Parameters

If a stored procedure exits with an unhandled exception, modified values of **OUT** and **INOUT** parameters are not propogated back to the caller.

If an exception is handled by a **CONTINUE** or **EXIT** handler that contains a [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) statement, execution of [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal) pops the Diagnostics Area stack, thus signalling the exception (that is, the information that existed before entry into the handler). If the exception is an error, the values of **OUT** and **INOUT** parameters are not propogated back to the caller.

### 13.6.8 Restrictions on Condition Handling

**[SIGNAL](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "signal" \o "13.6.7.5 SIGNAL Statement)**, [**RESIGNAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resignal), and [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) are not permissible as prepared statements. For example, this statement is invalid:

PREPARE stmt1 FROM 'SIGNAL SQLSTATE "02000"';

**SQLSTATE** values in class **'04'** are not treated specially. They are handled the same as other exceptions.

In standard SQL, the first condition relates to the **SQLSTATE** value returned for the previous SQL statement. In MySQL, this is not guaranteed, so to get the main error, you cannot do this:

GET DIAGNOSTICS CONDITION 1 @errno = MYSQL\_ERRNO;

Instead, do this:

GET DIAGNOSTICS @cno = NUMBER;

GET DIAGNOSTICS CONDITION @cno @errno = MYSQL\_ERRNO;

## 13.7 Database Administration Statements

[13.7.1 Account Management Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#account-management-statements)

[13.7.2 Resource Group Management Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#resource-group-statements)

[13.7.3 Table Maintenance Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table-maintenance-statements)

[13.7.4 Component, Plugin, and User-Defined Function Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#component-statements)

[13.7.5 CLONE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#clone)

[13.7.6 SET Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-statement)

[13.7.7 SHOW Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show)

[13.7.8 Other Administrative Statements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#other-administrative-statements)

### 13.7.1 Account Management Statements

[13.7.1.1 ALTER USER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user)

[13.7.1.2 CREATE ROLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-role)

[13.7.1.3 CREATE USER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user)

[13.7.1.4 DROP ROLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-role)

[13.7.1.5 DROP USER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user)

[13.7.1.6 GRANT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant)

[13.7.1.7 RENAME USER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user)

[13.7.1.8 REVOKE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke)

[13.7.1.9 SET DEFAULT ROLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role)

[13.7.1.10 SET PASSWORD Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password)

[13.7.1.11 SET ROLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role)

MySQL account information is stored in the tables of the **mysql** system schema. This database and the access control system are discussed extensively in [Chapter 5, *MySQL Server Administration*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html), which you should consult for additional details.

**Important**

Some MySQL releases introduce changes to the grant tables to add new privileges or features. To make sure that you can take advantage of any new capabilities, update your grant tables to the current structure whenever you upgrade MySQL. See [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading).

When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, account-management statements require the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege), in addition to any other required privileges. This is because they modify tables in the **mysql** system schema.

Account management statements are atomic and crash safe. For more information, see [Section 13.1.1, “Atomic Data Definition Statement Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#atomic-ddl).

#### 13.7.1.1 ALTER USER Statement

ALTER USER [IF EXISTS]

***user*** [***auth\_option***] [, ***user*** [***auth\_option***]] ...

[REQUIRE {NONE | ***tls\_option*** [[AND] ***tls\_option***] ...}]

[WITH ***resource\_option*** [***resource\_option***] ...]

[***password\_option*** | ***lock\_option***] ...

[COMMENT '***comment\_string***' | ATTRIBUTE '***json\_object***']

ALTER USER [IF EXISTS] USER() ***user\_func\_auth\_option***

ALTER USER [IF EXISTS]

***user*** DEFAULT ROLE

{NONE | ALL | ***role*** [, ***role*** ] ...}

***user***:

(see [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names))

***auth\_option***: {

IDENTIFIED BY '***auth\_string***'

[REPLACE '***current\_auth\_string***']

[RETAIN CURRENT PASSWORD]

| IDENTIFIED BY RANDOM PASSWORD

[REPLACE '***current\_auth\_string***']

[RETAIN CURRENT PASSWORD]

| IDENTIFIED WITH ***auth\_plugin***

| IDENTIFIED WITH ***auth\_plugin*** BY '***auth\_string***'

[REPLACE '***current\_auth\_string***']

[RETAIN CURRENT PASSWORD]

| IDENTIFIED WITH ***auth\_plugin*** BY RANDOM PASSWORD

[REPLACE '***current\_auth\_string***']

[RETAIN CURRENT PASSWORD]

| IDENTIFIED WITH ***auth\_plugin*** AS '***auth\_string***'

| DISCARD OLD PASSWORD

}

***user\_func\_auth\_option***: {

IDENTIFIED BY '***auth\_string***'

[REPLACE '***current\_auth\_string***']

[RETAIN CURRENT PASSWORD]

| DISCARD OLD 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***

}

***password\_option***: {

PASSWORD EXPIRE [DEFAULT | NEVER | INTERVAL ***N*** DAY]

| PASSWORD HISTORY {DEFAULT | ***N***}

| PASSWORD REUSE INTERVAL {DEFAULT | ***N*** DAY}

| PASSWORD REQUIRE CURRENT [DEFAULT | OPTIONAL]

| FAILED\_LOGIN\_ATTEMPTS ***N***

| PASSWORD\_LOCK\_TIME {***N*** | UNBOUNDED}

}

***lock\_option***: {

ACCOUNT LOCK

| ACCOUNT UNLOCK

}

The [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statement modifies MySQL accounts. It enables authentication, role, SSL/TLS, resource-limit, and password-management properties to be modified for existing accounts. It can also be used to lock and unlock accounts.

In most cases, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) requires the global [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege, or the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the **mysql** system schema. The exceptions are:

Any client who connects to the server using a nonanonymous account can change the password for that account. (In particular, you can change your own password.) To see which account the server authenticated you as, invoke the [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) function:

SELECT CURRENT\_USER();

For **DEFAULT ROLE** syntax, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) requires these privileges:

Setting the default roles for another user requires the global [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege, or the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the **mysql.default\_roles** system table.

Setting the default roles for yourself requires no special privileges, as long as the roles you want as the default have been granted to you.

Statements that modify secondary passwords require these privileges:

The [**APPLICATION\_PASSWORD\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_application-password-admin) privilege is required to use the **RETAIN CURRENT PASSWORD** or **DISCARD OLD PASSWORD** clause for [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statements that apply to your own account. The privilege is required to manipulate your own secondary password because most users require only one password.

If an account is to be permitted to manipulate secondary passwords for all accounts, it requires the [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege rather than [**APPLICATION\_PASSWORD\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_application-password-admin).

When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) additionally requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

By default, an error occurs if you try to modify a user that does not exist. If the **IF EXISTS** clause is given, the statement produces a warning for each named user that does not exist, rather than an error.

**Important**

Under some circumstances, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) may be recorded in server logs or on the client side in a history file such as ~/.mysql\_history, which means that cleartext passwords may be read by anyone having read access to that information. For information about the conditions under which this occurs for the server logs and how to control it, see [Section 6.1.2.3, “Passwords and Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-logging). For similar information about client-side logging, see [Section 4.5.1.3, “mysql Client Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-logging).

There are several aspects to the [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statement, described under the following topics:

[ALTER USER Overview](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-overview)

[ALTER USER Authentication Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-authentication)

[ALTER USER Role Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-role)

[ALTER USER SSL/TLS Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-tls)

[ALTER USER Resource-Limit Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-resource-limits)

[ALTER USER Password-Management Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-password-management)

[ALTER USER Account-Locking Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-account-locking)

[ALTER USER Binary Logging](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-binary-logging)

##### ALTER USER Overview

For each affected account, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) modifies the corresponding row in the **mysql.user** system table to reflect the properties specified in the statement. Unspecified properties retain their current values.

Each account name uses the format described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names). The host name part of the account name, if omitted, defaults to **'%'**. It is also possible to specify [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) or [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) to refer to the account associated with the current session.

For one syntax only, the account may be specified with the [**USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_user) function:

ALTER USER USER() IDENTIFIED BY '***auth\_string***';

This syntax enables changing your own password without naming your account literally. (The syntax also supports the **REPLACE**, **RETAIN CURRENT PASSWORD**, and **DISCARD OLD PASSWORD** clauses described at [ALTER USER Authentication Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user-authentication).)

MySQL 8.0.21 and later supports user comments and user attributes, as described in [Section 13.7.1.3, “CREATE USER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user). These can be modified employing **ALTER USER** by means of the **COMMENT** and **ATTRIBUTE** options, respectively. You cannot specify both options in the same **ALTER USER** statement; attempting to do so results in a syntax error.

The user comment and user attribute are stored in the [**INFORMATION\_SCHEMA.USER\_ATTRIBUTES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-user-attributes-table) table as a JSON object; the user comment is stored as the value for a **comment** key in the ATTRIBUTE column of this table, as shown later in this discussion. The **COMMENT** text can be any arbtitrary quoted text, and replaces any existing user comment. The **ATTRIBUTE** value must be the valid string represntation of a JSON object. This is merged with any existing user attribute as if the [**JSON\_MERGE\_PATCH()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_json-merge-patch) function had been used on the existing user attribute and the new one; for any keys that are re-used, the new value overwrites the old one, as shown here:

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER='bill' AND HOST='localhost';**

+------+-----------+----------------+

| USER | HOST | ATTRIBUTE |

+------+-----------+----------------+

| bill | localhost | {"foo": "bar"} |

+------+-----------+----------------+

1 row in set (0.11 sec)

mysql> **ALTER USER 'bill'@'localhost' ATTRIBUTE '{"baz": "faz", "foo": "moo"}';**

Query OK, 0 rows affected (0.22 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER='bill' AND HOST='localhost';**

+------+-----------+------------------------------+

| USER | HOST | ATTRIBUTE |

+------+-----------+------------------------------+

| bill | localhost | {"baz": "faz", "foo": "moo"} |

+------+-----------+------------------------------+

1 row in set (0.00 sec)

To remove a key and its value from the user attribute, set the key to JSON **null** (must be lowercase and unquoted), like this:

mysql> **ALTER USER 'bill'@'localhost' ATTRIBUTE '{"foo": null}';**

Query OK, 0 rows affected (0.08 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER='bill' AND HOST='localhost';**

+------+-----------+----------------+

| USER | HOST | ATTRIBUTE |

+------+-----------+----------------+

| bill | localhost | {"baz": "faz"} |

+------+-----------+----------------+

1 row in set (0.00 sec)

To set an existing user comment to an empty string, use **ALTER USER ... COMMENT ''**. This leaves an empty **comment** value in the [**USER\_ATTRIBUTES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-user-attributes-table) table; to remove the user comment completely, use **ALTER USER ... ATTRIBUTE ...** with the value for the column key set to JSON **null** (unquoted, in lower case). This is illustrated by the following sequence of SQL statements:

mysql> **ALTER USER 'bill'@'localhost' COMMENT 'Something about Bill';**

Query OK, 0 rows affected (0.06 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER='bill' AND HOST='localhost';**

+------+-----------+---------------------------------------------------+

| USER | HOST | ATTRIBUTE |

+------+-----------+---------------------------------------------------+

| bill | localhost | {"baz": "faz", "comment": "Something about Bill"} |

+------+-----------+---------------------------------------------------+

1 row in set (0.00 sec)

mysql> **ALTER USER 'bill'@'localhost' COMMENT '';**

Query OK, 0 rows affected (0.09 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER='bill' AND HOST='localhost';**

+------+-----------+-------------------------------+

| USER | HOST | ATTRIBUTE |

+------+-----------+-------------------------------+

| bill | localhost | {"baz": "faz", "comment": ""} |

+------+-----------+-------------------------------+

1 row in set (0.00 sec)

mysql> **ALTER USER 'bill'@'localhost' ATTRIBUTE '{"comment": null}';**

Query OK, 0 rows affected (0.07 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER='bill' AND HOST='localhost';**

+------+-----------+----------------+

| USER | HOST | ATTRIBUTE |

+------+-----------+----------------+

| bill | localhost | {"baz": "faz"} |

+------+-----------+----------------+

1 row in set (0.00 sec)

For [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) syntax that permits an ***auth\_option*** value to follow a ***user*** value, ***auth\_option*** indicates how the account authenticates by specifying an account authentication plugin, credentials (for example, a password), or both. Each ***auth\_option*** value applies only to the account named immediately preceding it.

Following the ***user*** specifications, the statement may include options for SSL/TLS, resource-limit, password-management, and locking properties. All such options are global to the statement and apply to all accounts named in the statement.

Example: Change an account's password and expire it. As a result, the user must connect with the named password and choose a new one at the next connection:

ALTER USER 'jeffrey'@'localhost'

IDENTIFIED BY '***new\_password***' PASSWORD EXPIRE;

Example: Modify an account to use the **caching\_sha2\_password** authentication plugin and the given password. Require that a new password be chosen every 180 days, and enable failed-login tracking, such that three consecutive incorrect passwords cause temporary account locking for two days:

ALTER USER 'jeffrey'@'localhost'

IDENTIFIED WITH caching\_sha2\_password BY '***new\_password***'

PASSWORD EXPIRE INTERVAL 180 DAY

FAILED\_LOGIN\_ATTEMPTS 3 PASSWORD\_LOCK\_TIME 2;

Example: Lock or unlock an account:

ALTER USER 'jeffrey'@'localhost' ACCOUNT LOCK;

ALTER USER 'jeffrey'@'localhost' ACCOUNT UNLOCK;

Example: Require an account to connect using SSL and establish a limit of 20 connections per hour:

ALTER USER 'jeffrey'@'localhost'

REQUIRE SSL WITH MAX\_CONNECTIONS\_PER\_HOUR 20;

Example: Alter multiple accounts, specifying some per-account properties and some global properties:

ALTER USER

'jeffrey'@'localhost'

IDENTIFIED BY '***jeffrey\_new\_password***',

'jeanne'@'localhost',

'josh'@'localhost'

IDENTIFIED BY '***josh\_new\_password***'

REPLACE '***josh\_current\_password***'

RETAIN CURRENT PASSWORD

REQUIRE SSL WITH MAX\_USER\_CONNECTIONS 2

PASSWORD HISTORY 5;

The **IDENTIFIED BY** value following **jeffrey** applies only to its immediately preceding account, so it changes the password to **'*jeffrey\_new\_password*'** only for **jeffrey**. For **jeanne**, there is no per-account value (thus leaving the password unchanged). For **josh**, **IDENTIFIED BY** establishes a new password (**'*josh\_new\_password*'**), **REPLACE** is specified to verify that the user issuing the [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statement knows the current password (**'*josh\_current\_password*'**), and that current password is also retained as the account secondary password. (As a result, **josh** can connect with either the primary or secondary password.)

The remaining properties apply globally to all accounts named in the statement, so for both accounts:

Connections are required to use SSL.

The account can be used for a maximum of two simultaneous connections.

Password changes cannot reuse any of the five most recent passwords.

Example: Discard the secondary password for **josh**, leaving the account with only its primary password:

ALTER USER 'josh'@'localhost' DISCARD OLD PASSWORD;

In the absence of a particular type of option, the account remains unchanged in that respect. For example, with no locking option, the locking state of the account is not changed.

##### ALTER USER Authentication Options

An account name may be followed by an ***auth\_option*** authentication option that specifies the account authentication plugin, credentials, or both. It may also include a password-verification clause that specifies the account current password to be replaced, and clauses that manage whether an account has a secondary password.

**Note**

Clauses for random password generation, password verification, and secondary passwords apply only to accounts that use an authentication plugin that stores credentials internally to MySQL. For accounts that use a plugin that performs authentication against a credentials system that is external to MySQL, password management must be handled externally against that system as well. For more information about internal credentials storage, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

***auth\_plugin*** names an authentication plugin. The plugin name can be a quoted string literal or an unquoted name. Plugin names are stored in the **plugin** column of the **mysql.user** system table.

For ***auth\_option*** syntax that does not specify an authentication plugin, the default plugin is indicated by the value of the [**default\_authentication\_plugin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_authentication_plugin) system variable. For descriptions of each plugin, see [Section 6.4.1, “Authentication Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#authentication-plugins).

Credentials that are stored internally are stored in the **mysql.user** system table. An **'*auth\_string*'** value or **RANDOM PASSWORD** specifies account credentials, either as a cleartext (unencrypted) string or hashed in the format expected by the authentication plugin associated with the account, respectively:

For syntax that uses **BY '*auth\_string*'**, the string is cleartext and is passed to the authentication plugin for possible hashing. The result returned by the plugin is stored in the **mysql.user** table. A plugin may use the value as specified, in which case no hashing occurs.

For syntax that uses **BY RANDOM PASSWORD**, MySQL generates a random password and as cleartext and passes it to the authentication plugin for possible hashing. The result returned by the plugin is stored in the **mysql.user** table. A plugin may use the value as specified, in which case no hashing occurs.

Randomly generated passwords are available as of MySQL 8.0.18 and have the characteristics described in [Random Password Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#random-password-generation).

For syntax that uses **AS '*auth\_string*'**, the string is assumed to be already in the format the authentication plugin requires, and is stored as is in the **mysql.user** table. If a plugin requires a hashed value, the value must be already hashed in a format appropriate for the plugin; otherwise, the value cannot be used by the plugin and correct authentication of client connections does not occur.

As of MySQL 8.0.17, a hashed string can be either a string literal or a hexadecimal value. The latter corresponds to the type of value displayed by [**SHOW CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-user) for password hashes containing unprintable characters when the [**print\_identified\_with\_as\_hex**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_print_identified_with_as_hex) system variable is enabled.

If an authentication plugin performs no hashing of the authentication string, the **BY '*auth\_string*'** and **AS '*auth\_string*'** clauses have the same effect: The authentication string is stored as is in the **mysql.user** system table.

The **REPLACE '*current\_auth\_string*'** clause performs password verification and is available as of MySQL 8.0.13. If given:

**REPLACE** specifies the account current password to be replaced, as a cleartext (unencrypted) string.

The clause must be given if password changes for the are required to specify the current password, as verification that the user attempting to make the change actually knows the current password.

The clause is optional if password changes for the account may but need not specify the current password.

The statement fails if the clause is given but does not match the current password, even if the clause is optional.

**REPLACE** can be specified only when changing the account password for the current user.

For more information about password verification by specifying the current password, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

The **RETAIN CURRENT PASSWORD** and **DISCARD OLD PASSWORD** clauses implement dual-password capability and are available as of MySQL 8.0.14. Both are optional, but if given, have the following effects:

**RETAIN CURRENT PASSWORD** retains an account current password as its secondary password, replacing any existing secondary password. The new password becomes the primary password, but clients can use the account to connect to the server using either the primary or secondary password. (Exception: If the new password specified by the [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statement is empty, the secondary password becomes empty as well, even if **RETAIN CURRENT PASSWORD** is given.)

If you specify **RETAIN CURRENT PASSWORD** for an account that has an empty primary password, the statement fails.

If an account has a secondary password and you change its primary password without specifying **RETAIN CURRENT PASSWORD**, the secondary password remains unchanged.

If you change the authentication plugin assigned to the account, the secondary password is discarded. If you change the authentication plugin and also specify **RETAIN CURRENT PASSWORD**, the statement fails.

**DISCARD OLD PASSWORD** discards the secondary password, if one exists. The account retains only its primary password, and clients can use the account to connect to the server only with the primary password.

For more information about use of dual passwords, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

[**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) permits these ***auth\_option*** syntaxes:

**IDENTIFIED BY '*auth\_string*' [REPLACE '*current\_auth\_string*'] [RETAIN CURRENT PASSWORD]**

Sets the account authentication plugin to the default plugin, passes the cleartext **'*auth\_string*'** value to the plugin for possible hashing, and stores the result in the account row in the **mysql.user** system table.

The **REPLACE** clause, if given, specifies the account current password, as described previously in this section.

The **RETAIN CURRENT PASSWORD** clause, if given, causes the account current password to be retained as its secondary password, as described previously in this section.

**IDENTIFIED BY RANDOM PASSWORD [REPLACE '*current\_auth\_string*'] [RETAIN CURRENT PASSWORD]**

Sets the account authentication plugin to the default plugin, generates a random password, passes the cleartext password value to the plugin for possible hashing, and stores the result in the account row in the **mysql.user** system table. The statement also returns the cleartext password in a result set to make it available to the user or application executing the statement. For details about the result set and characteristics of randomly generated passwords, see [Random Password Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#random-password-generation).

The **REPLACE** clause, if given, specifies the account current password, as described previously in this section.

The **RETAIN CURRENT PASSWORD** clause, if given, causes the account current password to be retained as its secondary password, as described previously in this section.

**IDENTIFIED WITH *auth\_plugin***

Sets the account authentication plugin to ***auth\_plugin***, clears the credentials to the empty string (the credentials are associated with the old authentication plugin, not the new one), and stores the result in the account row in the **mysql.user** system table.

In addition, the password is marked expired. The user must choose a new one when next connecting.

**IDENTIFIED WITH *auth\_plugin* BY '*auth\_string*' [REPLACE '*current\_auth\_string*'] [RETAIN CURRENT PASSWORD]**

Sets the account authentication plugin to ***auth\_plugin***, passes the cleartext **'*auth\_string*'** value to the plugin for possible hashing, and stores the result in the account row in the **mysql.user** system table.

The **REPLACE** clause, if given, specifies the account current password, as described previously in this section.

The **RETAIN CURRENT PASSWORD** clause, if given, causes the account current password to be retained as its secondary password, as described previously in this section.

**IDENTIFIED WITH *auth\_plugin* BY RANDOM PASSWORD [REPLACE '*current\_auth\_string*'] [RETAIN CURRENT PASSWORD]**

Sets the account authentication plugin to ***auth\_plugin***, generates a random password, passes the cleartext password value to the plugin for possible hashing, and stores the result in the account row in the **mysql.user** system table. The statement also returns the cleartext password in a result set to make it available to the user or application executing the statement. For details about the result set and characteristics of randomly generated passwords, see [Random Password Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#random-password-generation).

The **REPLACE** clause, if given, specifies the account current password, as described previously in this section.

The **RETAIN CURRENT PASSWORD** clause, if given, causes the account current password to be retained as its secondary password, as described previously in this section.

**IDENTIFIED WITH *auth\_plugin* AS '*auth\_string*'**

Sets the account authentication plugin to ***auth\_plugin*** and stores the **'*auth\_string*'** value as is in the **mysql.user** account row. If the plugin requires a hashed string, the string is assumed to be already hashed in the format the plugin requires.

**DISCARD OLD PASSWORD**

Discards the account secondary password, if there is one, as described previously in this section.

Example: Specify the password as cleartext; the default plugin is used:

ALTER USER 'jeffrey'@'localhost'

IDENTIFIED BY '***password***';

Example: Specify the authentication plugin, along with a cleartext password value:

ALTER USER 'jeffrey'@'localhost'

IDENTIFIED WITH mysql\_native\_password

BY '***password***';

Example: Like the preceding example, but in addition, specify the current password as a cleartext value to satisfy any account requirement that the user making the change knows that password:

ALTER USER 'jeffrey'@'localhost'

IDENTIFIED WITH mysql\_native\_password

BY '***password***'

REPLACE '***current\_password***';

The preceding statement fails unless the current user is **jeffrey** because **REPLACE** is permitted only for changes to the current user's password.

Example: Establish a new primary password and retain the existing password as the secondary password:

ALTER USER 'jeffrey'@'localhost'

IDENTIFIED BY '***new\_password***'

RETAIN CURRENT PASSWORD;

Example: Discard the secondary password, leaving the account with only its primary password:

ALTER USER 'jeffery'@'localhost' DISCARD OLD PASSWORD;

Example: Specify the authentication plugin, along with a hashed password value:

ALTER USER 'jeffrey'@'localhost'

IDENTIFIED WITH mysql\_native\_password

AS '\*6C8989366EAF75BB670AD8EA7A7FC1176A95CEF4';

For additional information about setting passwords and authentication plugins, see [Section 6.2.14, “Assigning Account Passwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#assigning-passwords), and [Section 6.2.17, “Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#pluggable-authentication).

##### ALTER USER Role Options

**[ALTER USER ... DEFAULT ROLE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "alter-user" \o "13.7.1.1 ALTER USER Statement)** defines which roles become active when the user connects to the server and authenticates, or when the user executes the [**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) statement during a session.

[**ALTER USER ... DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) is alternative syntax for [**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) (see [Section 13.7.1.9, “SET DEFAULT ROLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role)). However, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) can set the default for only a single user, whereas [**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) can set the default for multiple users. On the other hand, you can specify **CURRENT\_USER** as the user name for the [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statement, whereas you cannot for [**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role).

Each user account name uses the format described previously.

Each role name uses the format described in [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names). For example:

ALTER USER 'joe'@'10.0.0.1' DEFAULT ROLE administrator, developer;

The host name part of the role name, if omitted, defaults to **'%'**.

The clause following the **DEFAULT ROLE** keywords permits these values:

**NONE**: Set the default to **NONE** (no roles).

**ALL**: Set the default to all roles granted to the account.

***role* [, *role* ] ...**: Set the default to the named roles, which must exist and be granted to the account at the time [**ALTER USER ... DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) is executed.

##### ALTER USER SSL/TLS Options

MySQL can check X.509 certificate attributes in addition to the usual authentication that is based on the user name and credentials. For background information on the use of SSL/TLS with MySQL, see [Section 6.3, “Using Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connections).

To specify SSL/TLS-related options for a MySQL account, use a **REQUIRE** clause that specifies one or more ***tls\_option*** values.

Order of **REQUIRE** options does not matter, but no option can be specified twice. The **AND** keyword is optional between **REQUIRE** options.

[**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) permits these ***tls\_option*** values:

**NONE**

Indicates that all accounts named by the statement have no SSL or X.509 requirements. Unencrypted connections are permitted if the user name and password are valid. Encrypted connections can be used, at the client's option, if the client has the proper certificate and key files.

ALTER USER 'jeffrey'@'localhost' REQUIRE NONE;

Clients attempt to establish a secure connection by default. For clients that have **REQUIRE NONE**, the connection attempt falls back to an unencrypted connection if a secure connection cannot be established. To require an encrypted connection, a client need specify only the [--ssl-mode=REQUIRED](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-mode) option; the connection attempt fails if a secure connection cannot be established.

**SSL**

Tells the server to permit only encrypted connections for all accounts named by the statement.

ALTER USER 'jeffrey'@'localhost' REQUIRE SSL;

Clients attempt to establish a secure connection by default. For accounts that have **REQUIRE SSL**, the connection attempt fails if a secure connection cannot be established.

**X509**

For all accounts named by the statement, requires that clients present a valid certificate, but the exact certificate, issuer, and subject do not matter. The only requirement is that it should be possible to verify its signature with one of the CA certificates. Use of X.509 certificates always implies encryption, so the **SSL** option is unnecessary in this case.

ALTER USER 'jeffrey'@'localhost' REQUIRE X509;

For accounts with **REQUIRE X509**, clients must specify the [--ssl-key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-key) and [--ssl-cert](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-cert) options to connect. (It is recommended but not required that [--ssl-ca](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-ca) also be specified so that the public certificate provided by the server can be verified.) This is true for **ISSUER** and **SUBJECT** as well because those **REQUIRE** options imply the requirements of **X509**.

**ISSUER '*issuer*'**

For all accounts named by the statement, requires that clients present a valid X.509 certificate issued by CA **'*issuer*'**. If a client presents a certificate that is valid but has a different issuer, the server rejects the connection. Use of X.509 certificates always implies encryption, so the **SSL** option is unnecessary in this case.

ALTER USER 'jeffrey'@'localhost'

REQUIRE ISSUER '/C=SE/ST=Stockholm/L=Stockholm/

O=MySQL/CN=CA/emailAddress=ca@example.com';

Because **ISSUER** implies the requirements of **X509**, clients must specify the [--ssl-key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-key) and [--ssl-cert](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-cert) options to connect. (It is recommended but not required that [--ssl-ca](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-ca) also be specified so that the public certificate provided by the server can be verified.)

**SUBJECT '*subject*'**

For all accounts named by the statement, requires that clients present a valid X.509 certificate containing the subject ***subject***. If a client presents a certificate that is valid but has a different subject, the server rejects the connection. Use of X.509 certificates always implies encryption, so the **SSL** option is unnecessary in this case.

ALTER USER 'jeffrey'@'localhost'

REQUIRE SUBJECT '/C=SE/ST=Stockholm/L=Stockholm/

O=MySQL demo client certificate/

CN=client/emailAddress=client@example.com';

MySQL does a simple string comparison of the **'*subject*'** value to the value in the certificate, so lettercase and component ordering must be given exactly as present in the certificate.

Because **SUBJECT** implies the requirements of **X509**, clients must specify the [--ssl-key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-key) and [--ssl-cert](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-cert) options to connect. (It is recommended but not required that [--ssl-ca](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-ca) also be specified so that the public certificate provided by the server can be verified.)

**CIPHER '*cipher*'**

For all accounts named by the statement, requires a specific cipher method for encrypting connections. This option is needed to ensure that ciphers and key lengths of sufficient strength are used. Encryption can be weak if old algorithms using short encryption keys are used.

ALTER USER 'jeffrey'@'localhost'

REQUIRE CIPHER 'EDH-RSA-DES-CBC3-SHA';

The **SUBJECT**, **ISSUER**, and **CIPHER** options can be combined in the **REQUIRE** clause:

ALTER USER 'jeffrey'@'localhost'

REQUIRE SUBJECT '/C=SE/ST=Stockholm/L=Stockholm/

O=MySQL demo client certificate/

CN=client/emailAddress=client@example.com'

AND ISSUER '/C=SE/ST=Stockholm/L=Stockholm/

O=MySQL/CN=CA/emailAddress=ca@example.com'

AND CIPHER 'EDH-RSA-DES-CBC3-SHA';

##### ALTER USER Resource-Limit Options

It is possible to place limits on use of server resources by an account, as discussed in [Section 6.2.20, “Setting Account Resource Limits”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#user-resources). To do so, use a **WITH** clause that specifies one or more ***resource\_option*** values.

Order of **WITH** options does not matter, except that if a given resource limit is specified multiple times, the last instance takes precedence.

[**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) permits these ***resource\_option*** values:

**MAX\_QUERIES\_PER\_HOUR *count***, **MAX\_UPDATES\_PER\_HOUR *count***, **MAX\_CONNECTIONS\_PER\_HOUR *count***

For all accounts named by the statement, these options restrict how many queries, updates, and connections to the server are permitted to each account during any given one-hour period. If ***count*** is **0** (the default), this means that there is no limitation for the account.

**MAX\_USER\_CONNECTIONS *count***

For all accounts named by the statement, restricts the maximum number of simultaneous connections to the server by each account. A nonzero ***count*** specifies the limit for the account explicitly. If ***count*** is **0** (the default), the server determines the number of simultaneous connections for the account from the global value of the [**max\_user\_connections**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_user_connections) system variable. If [**max\_user\_connections**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_user_connections) is also zero, there is no limit for the account.

Example:

ALTER USER 'jeffrey'@'localhost'

WITH MAX\_QUERIES\_PER\_HOUR 500 MAX\_UPDATES\_PER\_HOUR 100;

##### ALTER USER Password-Management Options

[**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) supports several ***password\_option*** values for password management:

Password expiration options: You can expire an account password manually and establish its password expiration policy. Policy options do not expire the password. Instead, they determine how the server applies automatic expiration to the account based on password age, which is assessed from the date and time of the most recent account password change.

Password reuse options: You can restrict password reuse based on number of password changes, time elapsed, or both.

Password verification-required options: You can indicate whether attempts to change an account password must specify the current password, as verification that the user attempting to make the change actually knows the current password.

Incorrect-password failed-login tracking options: You can cause the server to track failed login attempts and temporarily lock accounts for which too many consecutive incorrect passwords are given. The required number of failures and the lock time are configurable.

This section describes the syntax for password-management options. For information about establishing policy for password management, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

If multiple password-management options of a given type are specified, the last one takes precedence. For example, **PASSWORD EXPIRE DEFAULT PASSWORD EXPIRE NEVER** is the same as **PASSWORD EXPIRE NEVER**.

**Note**

Except for the options that pertain to failed-login tracking, password-management options apply only to accounts that use an authentication plugin that stores credentials internally to MySQL. For accounts that use a plugin that performs authentication against a credentials system that is external to MySQL, password management must be handled externally against that system as well. For more information about internal credentials storage, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

A client has an expired password if the account password was expired manually or the password age is considered greater than its permitted lifetime per the automatic expiration policy. In this case, the server either disconnects the client or restricts the operations permitted to it (see [Section 6.2.16, “Server Handling of Expired Passwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#expired-password-handling)). Operations performed by a restricted client result in an error until the user establishes a new account password.

**Note**

Although it is possible to “reset” an expired password by setting it to its current value, it is preferable, as a matter of good policy, to choose a different password. DBAs can enforce non-reuse by establishing an appropriate password-reuse policy. See [Password Reuse Policy](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-reuse-policy).

[**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) permits these ***password\_option*** values for controlling password expiration:

**PASSWORD EXPIRE**

Immediately marks the password expired for all accounts named by the statement.

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE;

**PASSWORD EXPIRE DEFAULT**

Sets all accounts named by the statement so that the global expiration policy applies, as specified by the [**default\_password\_lifetime**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_password_lifetime) system variable.

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE DEFAULT;

**PASSWORD EXPIRE NEVER**

This expiration option overrides the global policy for all accounts named by the statement. For each, it disables password expiration so that the password never expires.

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE NEVER;

**PASSWORD EXPIRE INTERVAL *N* DAY**

This expiration option overrides the global policy for all accounts named by the statement. For each, it sets the password lifetime to ***N*** days. The following statement requires the password to be changed every 180 days:

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE INTERVAL 180 DAY;

[**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) permits these ***password\_option*** values for controlling reuse of previous passwords based on required minimum number of password changes:

**PASSWORD HISTORY DEFAULT**

Sets all accounts named by the statement so that the global policy about password history length applies, to prohibit reuse of passwords before the number of changes specified by the [**password\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_password_history) system variable.

ALTER USER 'jeffrey'@'localhost' PASSWORD HISTORY DEFAULT;

**PASSWORD HISTORY *N***

This history-length option overrides the global policy for all accounts named by the statement. For each, it sets the password history length to ***N*** passwords, to prohibit reusing any of the ***N*** most recently chosen passwords. The following statement prohibits reuse of any of the previous 6 passwords:

ALTER USER 'jeffrey'@'localhost' PASSWORD HISTORY 6;

[**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) permits these ***password\_option*** values for controlling reuse of previous passwords based on time elapsed:

**PASSWORD REUSE INTERVAL DEFAULT**

Sets all statements named by the account so that the global policy about time elapsed applies, to prohibit reuse of passwords newer than the number of days specified by the [**password\_reuse\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_password_reuse_interval) system variable.

ALTER USER 'jeffrey'@'localhost' PASSWORD REUSE INTERVAL DEFAULT;

**PASSWORD REUSE INTERVAL *N* DAY**

This time-elapsed option overrides the global policy for all accounts named by the statement. For each, it sets the password reuse interval to ***N*** days, to prohibit reuse of passwords newer than that many days. The following statement prohibits password reuse for 360 days:

ALTER USER 'jeffrey'@'localhost' PASSWORD REUSE INTERVAL 360 DAY;

[**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) permits these ***password\_option*** values for controlling whether attempts to change an account password must specify the current password, as verification that the user attempting to make the change actually knows the current password:

**PASSWORD REQUIRE CURRENT**

This verification option overrides the global policy for all accounts named by the statement. For each, it requires that password changes specify the current password.

ALTER USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT;

**PASSWORD REQUIRE CURRENT OPTIONAL**

This verification option overrides the global policy for all accounts named by the statement. For each, it does not require that password changes specify the current password. (The current password may but need not be given.)

ALTER USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT OPTIONAL;

**PASSWORD REQUIRE CURRENT DEFAULT**

Sets all statements named by the account so that the global policy about password verification applies, as specified by the [**password\_require\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_password_require_current) system variable.

ALTER USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT DEFAULT;

As of MySQL 8.0.19, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) permits these ***password\_option*** values for controlling failed-login tracking:

**FAILED\_LOGIN\_ATTEMPTS *N***

Whether to track account login attempts that specify an incorrect password. ***N*** must be a number from 0 to 32767. A value of 0 disables failed-login tracking. Values greater than 0 indicate how many consecutive password failures cause temporary account locking (if **PASSWORD\_LOCK\_TIME** is also nonzero).

**PASSWORD\_LOCK\_TIME {*N* | UNBOUNDED}**

How long to lock the account after too many consecutive login attempts provide an incorrect password. ***N*** must be a number from 0 to 32767, or **UNBOUNDED**. A value of 0 disables temporary account locking. Values greater than 0 indicate how long to lock the account in days. A value of **UNBOUNDED** causes the account locking duration to be unbounded; once locked, the account remains in a locked state until unlocked. For information about the conditions under which unlocking occurs, see [Failed-Login Tracking and Temporary Account Locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#failed-login-tracking).

For failed-login tracking and temporary locking to occur, an account's **FAILED\_LOGIN\_ATTEMPTS** and **PASSWORD\_LOCK\_TIME** options both must be nonzero. The following statement modifies an account such that it remains locked for two days after four consecutive password failures:

ALTER USER 'jeffrey'@'localhost'

FAILED\_LOGIN\_ATTEMPTS 4 PASSWORD\_LOCK\_TIME 2;

##### ALTER USER Account-Locking Options

MySQL supports account locking and unlocking using the **ACCOUNT LOCK** and **ACCOUNT UNLOCK** options, which specify the locking state for an account. For additional discussion, see [Section 6.2.19, “Account Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-locking).

If multiple account-locking options are specified, the last one takes precedence.

As of MySQL 8.0.19, [**ALTER USER ... UNLOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) unlocks any account named by the statement that is temporarily locked due to too many failed logins. See [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

##### ALTER USER Binary Logging

**[ALTER USER](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "alter-user" \o "13.7.1.1 ALTER USER Statement)** is written to the binary log if it succeeds, but not if it fails; in that case, rollback occurs and no changes are made. A statement written to the binary log includes all named users. If the **IF EXISTS** clause is given, this includes even users that do not exist and were not altered.

If the original statement changes the credentials for a user, the statement written to the binary log specifies the applicable authentication plugin for that user, determined as follows:

The plugin named in the original statement, if one was specified.

Otherwise, the plugin associated with the user account if the user exists, or the default authentication plugin if the user does not exist. (If the statement written to the binary log must specify a particular authentication plugin for a user, include it in the original statement.)

If the server adds the default authentication plugin for any users in the statement written to the binary log, it writes a warning to the error log naming those users.

If the original statement specifies the **FAILED\_LOGIN\_ATTEMPTS** or **PASSWORD\_LOCK\_TIME** option, the statement written to the binary log includes the option.

#### 13.7.1.2 CREATE ROLE Statement

CREATE ROLE [IF NOT EXISTS] ***role*** [, ***role*** ] ...

[**CREATE ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-role) creates one or more roles, which are named collections of privileges. To use this statement, you must have the global [**CREATE ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-role) or [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege. When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**CREATE ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-role) additionally requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

A role when created is locked, has no password, and is assigned the default authentication plugin. (These role attributes can be changed later with the [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statement, by users who have the global [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege.)

[**CREATE ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-role) either succeeds for all named roles or rolls back and has no effect if any error occurs. By default, an error occurs if you try to create a role that already exists. If the **IF NOT EXISTS** clause is given, the statement produces a warning for each named role that already exists, rather than an error.

The statement is written to the binary log if it succeeds, but not if it fails; in that case, rollback occurs and no changes are made. A statement written to the binary log includes all named roles. If the **IF NOT EXISTS** clause is given, this includes even roles that already exist and were not created.

Each role name uses the format described in [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names). For example:

CREATE ROLE 'admin', 'developer';

CREATE ROLE 'webapp'@'localhost';

The host name part of the role name, if omitted, defaults to **'%'**.

For role usage examples, see [Section 6.2.10, “Using Roles”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#roles).

#### 13.7.1.3 CREATE USER Statement

CREATE USER [IF NOT EXISTS]

***user*** [***auth\_option***] [, ***user*** [***auth\_option***]] ...

DEFAULT ROLE ***role*** [, ***role*** ] ...

[REQUIRE {NONE | ***tls\_option*** [[AND] ***tls\_option***] ...}]

[WITH ***resource\_option*** [***resource\_option***] ...]

[***password\_option*** | ***lock\_option***] ...

[COMMENT '***comment\_string***' | ATTRIBUTE '***json\_object***']

***user***:

(see [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names))

***auth\_option***: {

IDENTIFIED BY '***auth\_string***'

| IDENTIFIED BY RANDOM PASSWORD

| IDENTIFIED WITH ***auth\_plugin***

| IDENTIFIED WITH ***auth\_plugin*** BY '***auth\_string***'

| IDENTIFIED WITH ***auth\_plugin*** BY RANDOM PASSWORD

| IDENTIFIED WITH ***auth\_plugin*** AS '***auth\_string***'

}

***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***

}

***password\_option***: {

PASSWORD EXPIRE [DEFAULT | NEVER | INTERVAL ***N*** DAY]

| PASSWORD HISTORY {DEFAULT | ***N***}

| PASSWORD REUSE INTERVAL {DEFAULT | ***N*** DAY}

| PASSWORD REQUIRE CURRENT [DEFAULT | OPTIONAL]

| FAILED\_LOGIN\_ATTEMPTS ***N***

| PASSWORD\_LOCK\_TIME {***N*** | UNBOUNDED}

}

***lock\_option***: {

ACCOUNT LOCK

| ACCOUNT UNLOCK

}

The [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) statement creates new MySQL accounts. It enables authentication, role, SSL/TLS, resource-limit, and password-management properties to be established for new accounts. It also controls whether accounts are initially locked or unlocked.

To use [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user), you must have the global [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege, or the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege for the **mysql** system schema. When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) additionally requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

As of MySQL 8.0.22, [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) fails with an error if any account to be created is named as the **DEFINER** attribute for any stored object. (That is, the statement fails if creating an account would cause the account to adopt a currently orphaned stored object.) To perform the operation anyway, you must have the [**SET\_USER\_ID**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_set-user-id) privilege; in this case, the statement succeeds with a warning rather than failing with an error. Without [**SET\_USER\_ID**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_set-user-id), to perform the user-creation operation, drop the orphan objects, create the account and grant its privileges, and then re-create the dropped objects. For additional information, including how to identify which objects name a given account as the **DEFINER** attribute, see [Orphan Stored Objects](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security-orphan-objects).

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) either succeeds for all named users or rolls back and has no effect if any error occurs. By default, an error occurs if you try to create a user that already exists. If the **IF NOT EXISTS** clause is given, the statement produces a warning for each named user that already exists, rather than an error.

**Important**

Under some circumstances, [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) may be recorded in server logs or on the client side in a history file such as ~/.mysql\_history, which means that cleartext passwords may be read by anyone having read access to that information. For information about the conditions under which this occurs for the server logs and how to control it, see [Section 6.1.2.3, “Passwords and Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-logging). For similar information about client-side logging, see [Section 4.5.1.3, “mysql Client Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-logging).

There are several aspects to the [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) statement, described under the following topics:

[CREATE USER Overview](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user-overview)

[CREATE USER Authentication Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user-authentication)

[CREATE USER Role Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user-role)

[CREATE USER SSL/TLS Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user-tls)

[CREATE USER Resource-Limit Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user-resource-limits)

[CREATE USER Password-Management Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user-password-management)

[CREATE USER Account-Locking Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user-account-locking)

[CREATE USER Binary Logging](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user-binary-logging)

##### CREATE USER Overview

For each account, [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) creates a new row in the **mysql.user** system table. The account row reflects the properties specified in the statement. Unspecified properties are set to their default values:

Authentication: The authentication plugin defined by the [**default\_authentication\_plugin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_authentication_plugin) system variable, and empty credentials

Default role: **NONE**

SSL/TLS: **NONE**

Resource limits: Unlimited

Password management: **PASSWORD EXPIRE DEFAULT PASSWORD HISTORY DEFAULT PASSWORD REUSE INTERVAL DEFAULT PASSWORD REQUIRE CURRENT DEFAULT**; failed-login tracking and temporary account locking are disabled

Account locking: **ACCOUNT UNLOCK**

An account when first created has no privileges and a default role of **NONE**. To assign privileges or roles, use the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement.

Each account name uses the format described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names). For example:

CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY '***password***';

The host name part of the account name, if omitted, defaults to **'%'**.

Each ***user*** value naming an account may be followed by an optional ***auth\_option*** value that indicates how the account authenticates. These values enable account authentication plugins and credentials (for example, a password) to be specified. Each ***auth\_option*** value applies only to the account named immediately preceding it.

Following the ***user*** specifications, the statement may include options for SSL/TLS, resource-limit, password-management, and locking properties. All such options are global to the statement and apply to all accounts named in the statement.

Example: Create an account that uses the default authentication plugin and the given password. Mark the password expired so that the user must choose a new one at the first connection to the server:

CREATE USER 'jeffrey'@'localhost'

IDENTIFIED BY '***new\_password***' PASSWORD EXPIRE;

Example: Create an account that uses the **caching\_sha2\_password** authentication plugin and the given password. Require that a new password be chosen every 180 days, and enable failed-login tracking, such that three consecutive incorrect passwords cause temporary account locking for two days:

CREATE USER 'jeffrey'@'localhost'

IDENTIFIED WITH caching\_sha2\_password BY '***new\_password***'

PASSWORD EXPIRE INTERVAL 180 DAY

FAILED\_LOGIN\_ATTEMPTS 3 PASSWORD\_LOCK\_TIME 2;

Example: Create multiple accounts, specifying some per-account properties and some global properties:

CREATE USER

'jeffrey'@'localhost' IDENTIFIED WITH mysql\_native\_password

BY '***new\_password1***',

'jeanne'@'localhost' IDENTIFIED WITH caching\_sha2\_password

BY '***new\_password2***'

REQUIRE X509 WITH MAX\_QUERIES\_PER\_HOUR 60

PASSWORD HISTORY 5

ACCOUNT LOCK;

Each ***auth\_option*** value (**IDENTIFIED WITH ... BY** in this case) applies only to the account named immediately preceding it, so each account uses the immediately following authentication plugin and password.

The remaining properties apply globally to all accounts named in the statement, so for both accounts:

Connections must be made using a valid X.509 certificate.

Up to 60 queries per hour are permitted.

Password changes cannot reuse any of the five most recent passwords.

The account is locked initially, so effectively it is a placeholder and cannot be used until an administrator unlocks it.

Beginning with MySQL 8.0.21, you can optionally create a user with a user comment or a user attribute, as described here:

***User comment***

To set a user comment, add **COMMENT '*user\_comment*'** to the **CREATE USER** statement, where ***user\_comment*** is the text of the user comment.

Example (omitting any other options):

CREATE USER 'jon'@'localhost' COMMENT 'Some information about Jon';

***User attribute***

A user attribute is a JSON object made up of one or more key-value pairs, and is set by including **ATTRIBUTE '*json\_object*'** as part of of **CREATE USER**. ***json\_object*** must be a valid JSON object.

Example (omitting any other options):

CREATE USER 'jim'@'localhost'

ATTRIBUTE '{"fname": "James", "lname": "Scott", "phone": "123-456-7890"}';

User comments and user attributes are stored together in the **ATTRIBUTE** column of the [**INFORMATION\_SCHEMA.USER\_ATTRIBUTES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-user-attributes-table) table. This query displays the row in this table inserted by the statement just shown for creating the user **jin@localhost**:

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER = 'jim' AND HOST = 'localhost'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

USER: jim

HOST: localhost

ATTRIBUTE: {"fname": "James", "lname": "Scott", "phone": "123-456-7890"}

1 row in set (0.00 sec)

The **COMMENT** option in actuality provides a shortcut for setting a user attribute whose only element has **comment** as its key and whose value is the argument supplied for the option. You can see this by executing the statement **CREATE USER 'jon'@'localhost' COMMENT 'Some information about Jon'**, and observing the row which it inserts into the [**USER\_ATTRIBUTES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-user-attributes-table) table:

mysql> **CREATE USER 'jon'@'localhost' COMMENT 'Some information about Jon';**

Query OK, 0 rows affected (0.06 sec)

mysql> **SELECT \* FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER = 'jon' AND HOST = 'localhost';**

+------+-----------+-------------------------------------------+

| USER | HOST | ATTRIBUTE |

+------+-----------+-------------------------------------------+

| jon | localhost | {"comment": "Some information about Jon"} |

+------+-----------+-------------------------------------------+

1 row in set (0.00 sec)

You cannot use **COMMENT** and **ATTRIBUTE** together in the same **CREATE USER** statement; attempting to do so causes a syntax error. To set a user comment concurrently with setting a user attribute, use **ATTRIBUTE** and include in its argument a value with a **comment** key, like this:

mysql> CREATE USER 'bill'@'localhost'

-> ATTRIBUTE '{"fname":"William", "lname":"Schmidt",

-> "comment":"Website developer"}';

Query OK, 0 rows affected (0.16 sec)

Since the content of the **ATTRIBUTE** row is a JSON object, you can employ any appropriate MySQL JSON functions or operators to manipulate it, as shown here:

mysql> **SELECT**

-> **USER AS User,**

-> **HOST AS Host,**

-> **CONCAT(ATTRIBUTE->>"$.fname"," ",ATTRIBUTE->>"$.lname") AS 'Full Name',**

-> **ATTRIBUTE->>"$.comment" AS Comment**

-> **FROM INFORMATION\_SCHEMA.USER\_ATTRIBUTES**

-> **WHERE USER='bill' AND HOST='localhost';**

+------+-----------+-----------------+-------------------+

| User | Host | Full Name | Comment |

+------+-----------+-----------------+-------------------+

| bill | localhost | William Schmidt | Website developer |

+------+-----------+-----------------+-------------------+

1 row in set (0.00 sec)

To set or to make changes in the user comment or user attribute for an existing user, you can use a **COMMENT** or **ATTRIBUTE** option with an [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statement.

Because the user comment and user attribute are stored together internally in a single [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json) column, this sets an upper limit on their maximum combined size; see [JSON Storage Requirements](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-types-storage-reqs-json), for more information.

See also the description of the Information Schema [**USER\_ATTRIBUTES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-user-attributes-table) table for more information and examples.

##### CREATE USER Authentication Options

An account name may be followed by an ***auth\_option*** authentication option that specifies the account authentication plugin, credentials, or both.

**Note**

Clauses for random password generation apply only to accounts that use an authentication plugin that stores credentials internally to MySQL. For accounts that use a plugin that performs authentication against a credentials system that is external to MySQL, password management must be handled externally against that system as well. For more information about internal credentials storage, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

***auth\_plugin*** names an authentication plugin. The plugin name can be a quoted string literal or an unquoted name. Plugin names are stored in the **plugin** column of the **mysql.user** system table.

For ***auth\_option*** syntax that does not specify an authentication plugin, the default plugin is indicated by the value of the [**default\_authentication\_plugin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_authentication_plugin) system variable. For descriptions of each plugin, see [Section 6.4.1, “Authentication Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#authentication-plugins).

Credentials that are stored internally are stored in the **mysql.user** system table. An **'*auth\_string*'** value or **RANDOM PASSWORD** specifies account credentials, either as a cleartext (unencrypted) string or hashed in the format expected by the authentication plugin associated with the account, respectively:

For syntax that uses **BY '*auth\_string*'**, the string is cleartext and is passed to the authentication plugin for possible hashing. The result returned by the plugin is stored in the **mysql.user** table. A plugin may use the value as specified, in which case no hashing occurs.

For syntax that uses **BY RANDOM PASSWORD**, MySQL generates a random password and as cleartext and passes it to the authentication plugin for possible hashing. The result returned by the plugin is stored in the **mysql.user** table. A plugin may use the value as specified, in which case no hashing occurs.

Randomly generated passwords are available as of MySQL 8.0.18 and have the characteristics described in [Random Password Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#random-password-generation).

For syntax that uses **AS '*auth\_string*'**, the string is assumed to be already in the format the authentication plugin requires, and is stored as is in the **mysql.user** table. If a plugin requires a hashed value, the value must be already hashed in a format appropriate for the plugin; otherwise, the value cannot be used by the plugin and correct authentication of client connections does not occur.

As of MySQL 8.0.17, a hashed string can be either a string literal or a hexadecimal value. The latter corresponds to the type of value displayed by [**SHOW CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-user) for password hashes containing unprintable characters when the [**print\_identified\_with\_as\_hex**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_print_identified_with_as_hex) system variable is enabled.

If an authentication plugin performs no hashing of the authentication string, the **BY '*auth\_string*'** and **AS '*auth\_string*'** clauses have the same effect: The authentication string is stored as is in the **mysql.user** system table.

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) permits these ***auth\_option*** syntaxes:

**IDENTIFIED BY '*auth\_string*'**

Sets the account authentication plugin to the default plugin, passes the cleartext **'*auth\_string*'** value to the plugin for possible hashing, and stores the result in the account row in the **mysql.user** system table.

**IDENTIFIED BY RANDOM PASSWORD**

Sets the account authentication plugin to the default plugin, generates a random password, passes the cleartext password value to the plugin for possible hashing, and stores the result in the account row in the **mysql.user** system table. The statement also returns the cleartext password in a result set to make it available to the user or application executing the statement. For details about the result set and characteristics of randomly generated passwords, see [Random Password Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#random-password-generation).

**IDENTIFIED WITH *auth\_plugin***

Sets the account authentication plugin to ***auth\_plugin***, clears the credentials to the empty string, and stores the result in the account row in the **mysql.user** system table.

**IDENTIFIED WITH *auth\_plugin* BY '*auth\_string*'**

Sets the account authentication plugin to ***auth\_plugin***, passes the cleartext **'*auth\_string*'** value to the plugin for possible hashing, and stores the result in the account row in the **mysql.user** system table.

**IDENTIFIED WITH *auth\_plugin* BY RANDOM PASSWORD**

Sets the account authentication plugin to ***auth\_plugin***, generates a random password, passes the cleartext password value to the plugin for possible hashing, and stores the result in the account row in the **mysql.user** system table. The statement also returns the cleartext password in a result set to make it available to the user or application executing the statement. For details about the result set and characteristics of randomly generated passwords, see [Random Password Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#random-password-generation).

**IDENTIFIED WITH *auth\_plugin* AS '*auth\_string*'**

Sets the account authentication plugin to ***auth\_plugin*** and stores the **'*auth\_string*'** value as is in the **mysql.user** account row. If the plugin requires a hashed string, the string is assumed to be already hashed in the format the plugin requires.

Example: Specify the password as cleartext; the default plugin is used:

CREATE USER 'jeffrey'@'localhost'

IDENTIFIED BY '***password***';

Example: Specify the authentication plugin, along with a cleartext password value:

CREATE USER 'jeffrey'@'localhost'

IDENTIFIED WITH mysql\_native\_password BY '***password***';

In each case, the password value stored in the account row is the cleartext value **'*password*'** after it has been hashed by the authentication plugin associated with the account.

For additional information about setting passwords and authentication plugins, see [Section 6.2.14, “Assigning Account Passwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#assigning-passwords), and [Section 6.2.17, “Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#pluggable-authentication).

##### CREATE USER Role Options

The **DEFAULT ROLE** clause defines which roles become active when the user connects to the server and authenticates, or when the user executes the [**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) statement during a session.

Each role name uses the format described in [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names). For example:

CREATE USER 'joe'@'10.0.0.1' DEFAULT ROLE administrator, developer;

The host name part of the role name, if omitted, defaults to **'%'**.

The **DEFAULT ROLE** clause permits a list of one or more comma-separated role names. These roles need not exist at the time [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) is executed.

##### CREATE USER SSL/TLS Options

MySQL can check X.509 certificate attributes in addition to the usual authentication that is based on the user name and credentials. For background information on the use of SSL/TLS with MySQL, see [Section 6.3, “Using Encrypted Connections”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connections).

To specify SSL/TLS-related options for a MySQL account, use a **REQUIRE** clause that specifies one or more ***tls\_option*** values.

Order of **REQUIRE** options does not matter, but no option can be specified twice. The **AND** keyword is optional between **REQUIRE** options.

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) permits these ***tls\_option*** values:

**NONE**

Indicates that all accounts named by the statement have no SSL or X.509 requirements. Unencrypted connections are permitted if the user name and password are valid. Encrypted connections can be used, at the client's option, if the client has the proper certificate and key files.

CREATE USER 'jeffrey'@'localhost' REQUIRE NONE;

Clients attempt to establish a secure connection by default. For clients that have **REQUIRE NONE**, the connection attempt falls back to an unencrypted connection if a secure connection cannot be established. To require an encrypted connection, a client need specify only the [--ssl-mode=REQUIRED](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-mode) option; the connection attempt fails if a secure connection cannot be established.

**NONE** is the default if no SSL-related **REQUIRE** options are specified.

**SSL**

Tells the server to permit only encrypted connections for all accounts named by the statement.

CREATE USER 'jeffrey'@'localhost' REQUIRE SSL;

Clients attempt to establish a secure connection by default. For accounts that have **REQUIRE SSL**, the connection attempt fails if a secure connection cannot be established.

**X509**

For all accounts named by the statement, requires that clients present a valid certificate, but the exact certificate, issuer, and subject do not matter. The only requirement is that it should be possible to verify its signature with one of the CA certificates. Use of X.509 certificates always implies encryption, so the **SSL** option is unnecessary in this case.

CREATE USER 'jeffrey'@'localhost' REQUIRE X509;

For accounts with **REQUIRE X509**, clients must specify the [--ssl-key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-key) and [--ssl-cert](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-cert) options to connect. (It is recommended but not required that [--ssl-ca](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-ca) also be specified so that the public certificate provided by the server can be verified.) This is true for **ISSUER** and **SUBJECT** as well because those **REQUIRE** options imply the requirements of **X509**.

**ISSUER '*issuer*'**

For all accounts named by the statement, requires that clients present a valid X.509 certificate issued by CA **'*issuer*'**. If a client presents a certificate that is valid but has a different issuer, the server rejects the connection. Use of X.509 certificates always implies encryption, so the **SSL** option is unnecessary in this case.

CREATE USER 'jeffrey'@'localhost'

REQUIRE ISSUER '/C=SE/ST=Stockholm/L=Stockholm/

O=MySQL/CN=CA/emailAddress=ca@example.com';

Because **ISSUER** implies the requirements of **X509**, clients must specify the [--ssl-key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-key) and [--ssl-cert](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-cert) options to connect. (It is recommended but not required that [--ssl-ca](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-ca) also be specified so that the public certificate provided by the server can be verified.)

**SUBJECT '*subject*'**

For all accounts named by the statement, requires that clients present a valid X.509 certificate containing the subject ***subject***. If a client presents a certificate that is valid but has a different subject, the server rejects the connection. Use of X.509 certificates always implies encryption, so the **SSL** option is unnecessary in this case.

CREATE USER 'jeffrey'@'localhost'

REQUIRE SUBJECT '/C=SE/ST=Stockholm/L=Stockholm/

O=MySQL demo client certificate/

CN=client/emailAddress=client@example.com';

MySQL does a simple string comparison of the **'*subject*'** value to the value in the certificate, so lettercase and component ordering must be given exactly as present in the certificate.

Because **SUBJECT** implies the requirements of **X509**, clients must specify the [--ssl-key](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-key) and [--ssl-cert](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-cert) options to connect. (It is recommended but not required that [--ssl-ca](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_general_ssl-ca) also be specified so that the public certificate provided by the server can be verified.)

**CIPHER '*cipher*'**

For all accounts named by the statement, requires a specific cipher method for encrypting connections. This option is needed to ensure that ciphers and key lengths of sufficient strength are used. Encryption can be weak if old algorithms using short encryption keys are used.

CREATE USER 'jeffrey'@'localhost'

REQUIRE CIPHER 'EDH-RSA-DES-CBC3-SHA';

The **SUBJECT**, **ISSUER**, and **CIPHER** options can be combined in the **REQUIRE** clause:

CREATE USER 'jeffrey'@'localhost'

REQUIRE SUBJECT '/C=SE/ST=Stockholm/L=Stockholm/

O=MySQL demo client certificate/

CN=client/emailAddress=client@example.com'

AND ISSUER '/C=SE/ST=Stockholm/L=Stockholm/

O=MySQL/CN=CA/emailAddress=ca@example.com'

AND CIPHER 'EDH-RSA-DES-CBC3-SHA';

##### CREATE USER Resource-Limit Options

It is possible to place limits on use of server resources by an account, as discussed in [Section 6.2.20, “Setting Account Resource Limits”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#user-resources). To do so, use a **WITH** clause that specifies one or more ***resource\_option*** values.

Order of **WITH** options does not matter, except that if a given resource limit is specified multiple times, the last instance takes precedence.

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) permits these ***resource\_option*** values:

**MAX\_QUERIES\_PER\_HOUR *count***, **MAX\_UPDATES\_PER\_HOUR *count***, **MAX\_CONNECTIONS\_PER\_HOUR *count***

For all accounts named by the statement, these options restrict how many queries, updates, and connections to the server are permitted to each account during any given one-hour period. If ***count*** is **0** (the default), this means that there is no limitation for the account.

**MAX\_USER\_CONNECTIONS *count***

For all accounts named by the statement, restricts the maximum number of simultaneous connections to the server by each account. A nonzero ***count*** specifies the limit for the account explicitly. If ***count*** is **0** (the default), the server determines the number of simultaneous connections for the account from the global value of the [**max\_user\_connections**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_user_connections) system variable. If [**max\_user\_connections**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_user_connections) is also zero, there is no limit for the account.

Example:

CREATE USER 'jeffrey'@'localhost'

WITH MAX\_QUERIES\_PER\_HOUR 500 MAX\_UPDATES\_PER\_HOUR 100;

##### CREATE USER Password-Management Options

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) supports several ***password\_option*** values for password management:

Password expiration options: You can expire an account password manually and establish its password expiration policy. Policy options do not expire the password. Instead, they determine how the server applies automatic expiration to the account based on password age, which is assessed from the date and time of the most recent account password change.

Password reuse options: You can restrict password reuse based on number of password changes, time elapsed, or both.

Password verification-required options: You can indicate whether attempts to change an account password must specify the current password, as verification that the user attempting to make the change actually knows the current password.

Incorrect-password failed-login tracking options: You can cause the server to track failed login attempts and temporarily lock accounts for which too many consecutive incorrect passwords are given. The required number of failures and the lock time are configurable.

This section describes the syntax for password-management options. For information about establishing policy for password management, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

If multiple password-management options of a given type are specified, the last one takes precedence. For example, **PASSWORD EXPIRE DEFAULT PASSWORD EXPIRE NEVER** is the same as **PASSWORD EXPIRE NEVER**.

**Note**

Except for the options that pertain to failed-login tracking, password-management options apply only to accounts that use an authentication plugin that stores credentials internally to MySQL. For accounts that use a plugin that performs authentication against a credentials system that is external to MySQL, password management must be handled externally against that system as well. For more information about internal credentials storage, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

A client has an expired password if the account password was expired manually or the password age is considered greater than its permitted lifetime per the automatic expiration policy. In this case, the server either disconnects the client or restricts the operations permitted to it (see [Section 6.2.16, “Server Handling of Expired Passwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#expired-password-handling)). Operations performed by a restricted client result in an error until the user establishes a new account password.

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) permits these ***password\_option*** values for controlling password expiration:

**PASSWORD EXPIRE**

Immediately marks the password expired for all accounts named by the statement.

CREATE USER 'jeffrey'@'localhost' PASSWORD EXPIRE;

**PASSWORD EXPIRE DEFAULT**

Sets all accounts named by the statement so that the global expiration policy applies, as specified by the [**default\_password\_lifetime**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_default_password_lifetime) system variable.

CREATE USER 'jeffrey'@'localhost' PASSWORD EXPIRE DEFAULT;

**PASSWORD EXPIRE NEVER**

This expiration option overrides the global policy for all accounts named by the statement. For each, it disables password expiration so that the password never expires.

CREATE USER 'jeffrey'@'localhost' PASSWORD EXPIRE NEVER;

**PASSWORD EXPIRE INTERVAL *N* DAY**

This expiration option overrides the global policy for all accounts named by the statement. For each, it sets the password lifetime to ***N*** days. The following statement requires the password to be changed every 180 days:

CREATE USER 'jeffrey'@'localhost' PASSWORD EXPIRE INTERVAL 180 DAY;

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) permits these ***password\_option*** values for controlling reuse of previous passwords based on required minimum number of password changes:

**PASSWORD HISTORY DEFAULT**

Sets all accounts named by the statement so that the global policy about password history length applies, to prohibit reuse of passwords before the number of changes specified by the [**password\_history**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_password_history) system variable.

CREATE USER 'jeffrey'@'localhost' PASSWORD HISTORY DEFAULT;

**PASSWORD HISTORY *N***

This history-length option overrides the global policy for all accounts named by the statement. For each, it sets the password history length to ***N*** passwords, to prohibit reusing any of the ***N*** most recently chosen passwords. The following statement prohibits reuse of any of the previous 6 passwords:

CREATE USER 'jeffrey'@'localhost' PASSWORD HISTORY 6;

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) permits these ***password\_option*** values for controlling reuse of previous passwords based on time elapsed:

**PASSWORD REUSE INTERVAL DEFAULT**

Sets all statements named by the account so that the global policy about time elapsed applies, to prohibit reuse of passwords newer than the number of days specified by the [**password\_reuse\_interval**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_password_reuse_interval) system variable.

CREATE USER 'jeffrey'@'localhost' PASSWORD REUSE INTERVAL DEFAULT;

**PASSWORD REUSE INTERVAL *N* DAY**

This time-elapsed option overrides the global policy for all accounts named by the statement. For each, it sets the password reuse interval to ***N*** days, to prohibit reuse of passwords newer than that many days. The following statement prohibits password reuse for 360 days:

CREATE USER 'jeffrey'@'localhost' PASSWORD REUSE INTERVAL 360 DAY;

[**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) permits these ***password\_option*** values for controlling whether attempts to change an account password must specify the current password, as verification that the user attempting to make the change actually knows the current password:

**PASSWORD REQUIRE CURRENT**

This verification option overrides the global policy for all accounts named by the statement. For each, it requires that password changes specify the current password.

CREATE USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT;

**PASSWORD REQUIRE CURRENT OPTIONAL**

This verification option overrides the global policy for all accounts named by the statement. For each, it does not require that password changes specify the current password. (The current password may but need not be given.)

CREATE USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT OPTIONAL;

**PASSWORD REQUIRE CURRENT DEFAULT**

Sets all statements named by the account so that the global policy about password verification applies, as specified by the [**password\_require\_current**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_password_require_current) system variable.

CREATE USER 'jeffrey'@'localhost' PASSWORD REQUIRE CURRENT DEFAULT;

As of MySQL 8.0.19, [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) permits these ***password\_option*** values for controlling failed-login tracking:

**FAILED\_LOGIN\_ATTEMPTS *N***

Whether to track account login attempts that specify an incorrect password. ***N*** must be a number from 0 to 32767. A value of 0 disables failed-login tracking. Values greater than 0 indicate how many consecutive password failures cause temporary account locking (if **PASSWORD\_LOCK\_TIME** is also nonzero).

**PASSWORD\_LOCK\_TIME {*N* | UNBOUNDED}**

How long to lock the account after too many consecutive login attempts provide an incorrect password. ***N*** must be a number from 0 to 32767, or **UNBOUNDED**. A value of 0 disables temporary account locking. Values greater than 0 indicate how long to lock the account in days. A value of **UNBOUNDED** causes the account locking duration to be unbounded; once locked, the account remains in a locked state until unlocked. For information about the conditions under which unlocking occurs, see [Failed-Login Tracking and Temporary Account Locking](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#failed-login-tracking).

For failed-login tracking and temporary locking to occur, an account's **FAILED\_LOGIN\_ATTEMPTS** and **PASSWORD\_LOCK\_TIME** options both must be nonzero. The following statement creates an account that remains locked for two days after four consecutive password failures:

CREATE USER 'jeffrey'@'localhost'

FAILED\_LOGIN\_ATTEMPTS 4 PASSWORD\_LOCK\_TIME 2;

##### CREATE USER Account-Locking Options

MySQL supports account locking and unlocking using the **ACCOUNT LOCK** and **ACCOUNT UNLOCK** options, which specify the locking state for an account. For additional discussion, see [Section 6.2.19, “Account Locking”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-locking).

If multiple account-locking options are specified, the last one takes precedence.

##### CREATE USER Binary Logging

**[CREATE USER](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "create-user" \o "13.7.1.3 CREATE USER Statement)** is written to the binary log if it succeeds, but not if it fails; in that case, rollback occurs and no changes are made. A statement written to the binary log includes all named users. If the **IF NOT EXISTS** clause is given, this includes even users that already exist and were not created.

The statement written to the binary log specifies an authentication plugin for each user, determined as follows:

The plugin named in the original statement, if one was specified.

Otherwise, the default authentication plugin. In particular, if a user **u1** already exists and uses a nondefault authentication plugin, the statement written to the binary log for **CREATE USER IF NOT EXISTS u1** names the default authentication plugin. (If the statement written to the binary log must specify a nondefault authentication plugin for a user, include it in the original statement.)

If the server adds the default authentication plugin for any nonexisting users in the statement written to the binary log, it writes a warning to the error log naming those users.

If the original statement specifies the **FAILED\_LOGIN\_ATTEMPTS** or **PASSWORD\_LOCK\_TIME** option, the statement written to the binary log includes the option.

#### 13.7.1.4 DROP ROLE Statement

DROP ROLE [IF EXISTS] ***role*** [, ***role*** ] ...

[**DROP ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-role) removes one or more roles (named collections of privileges). To use this statement, you must have the global [**DROP ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop-role) or [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege. When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**DROP ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-role) additionally requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

As of MySQL 8.0.16, users who have the [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege can use this statement to drop accounts that are locked or unlocked. Users who have the [**DROP ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop-role) privilege can use this statement only to drop accounts that are locked (unlocked accounts are presumably user accounts used to log in to the server and not just as roles).

Roles named in the [**mandatory\_roles**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_mandatory_roles) system variable value cannot be dropped.

[**DROP ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-role) either succeeds for all named roles or rolls back and has no effect if any error occurs. By default, an error occurs if you try to drop a role that does not exist. If the **IF EXISTS** clause is given, the statement produces a warning for each named role that does not exist, rather than an error.

The statement is written to the binary log if it succeeds, but not if it fails; in that case, rollback occurs and no changes are made. A statement written to the binary log includes all named roles. If the **IF EXISTS** clause is given, this includes even roles that do not exist and were not dropped.

Each role name uses the format described in [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names). For example:

DROP ROLE 'admin', 'developer';

DROP ROLE 'webapp'@'localhost';

The host name part of the role name, if omitted, defaults to **'%'**.

A dropped role is automatically revoked from any user account (or role) to which the role was granted. Within any current session for such an account, its adjusted privileges apply beginning with the next statement executed.

For role usage examples, see [Section 6.2.10, “Using Roles”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#roles).

#### 13.7.1.5 DROP USER Statement

DROP USER [IF EXISTS] ***user*** [, ***user***] ...

The [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user) statement removes one or more MySQL accounts and their privileges. It removes privilege rows for the account from all grant tables.

Roles named in the [**mandatory\_roles**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_mandatory_roles) system variable value cannot be dropped.

To use [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user), you must have the global [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege, or the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privilege for the **mysql** system schema. When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user) additionally requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

As of MySQL 8.0.22, [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user) fails with an error if any account to be dropped is named as the **DEFINER** attribute for any stored object. (That is, the statement fails if dropping an account would cause a stored object to become orphaned.) To perform the operation anyway, you must have the [**SET\_USER\_ID**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_set-user-id) privilege; in this case, the statement succeeds with a warning rather than failing with an error. For additional information, including how to identify which objects name a given account as the **DEFINER** attribute, see [Orphan Stored Objects](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security-orphan-objects).

[**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user) either succeeds for all named users or rolls back and has no effect if any error occurs. By default, an error occurs if you try to drop a user that does not exist. If the **IF EXISTS** clause is given, the statement produces a warning for each named user that does not exist, rather than an error.

The statement is written to the binary log if it succeeds, but not if it fails; in that case, rollback occurs and no changes are made. A statement written to the binary log includes all named users. If the **IF EXISTS** clause is given, this includes even users that do not exist and were not dropped.

Each account name uses the format described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names). For example:

DROP USER 'jeffrey'@'localhost';

The host name part of the account name, if omitted, defaults to **'%'**.

**Important**

[**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user) does not automatically close any open user sessions. Rather, in the event that a user with an open session is dropped, the statement does not take effect until that user's session is closed. Once the session is closed, the user is dropped, and that user's next attempt to log in fails. This is by design.

[**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user) does not automatically drop or invalidate databases or objects within them that the old user created. This includes stored programs or views for which the **DEFINER** attribute names the dropped user. Attempts to access such objects may produce an error if they execute in definer security context. (For information about security context, see [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security).)

#### 13.7.1.6 GRANT Statement

GRANT

***priv\_type*** [(***column\_list***)]

[, ***priv\_type*** [(***column\_list***)]] ...

ON [***object\_type***] ***priv\_level***

TO ***user\_or\_role*** [, ***user\_or\_role***] ...

[WITH GRANT OPTION]

[AS ***user***

[WITH ROLE

DEFAULT

| NONE

| ALL

| ALL EXCEPT ***role*** [, ***role*** ] ...

| ***role*** [, ***role*** ] ...

]

]

}

GRANT PROXY ON ***user\_or\_role***

TO ***user\_or\_role*** [, ***user\_or\_role***] ...

[WITH GRANT OPTION]

GRANT ***role*** [, ***role***] ...

TO ***user\_or\_role*** [, ***user\_or\_role***] ...

[WITH ADMIN OPTION]

***object\_type***: {

TABLE

| FUNCTION

| PROCEDURE

}

***priv\_level***: {

\*

| \*.\*

| ***db\_name***.\*

| ***db\_name.tbl\_name***

| ***tbl\_name***

| ***db\_name***.***routine\_name***

}

***user\_or\_role***: {

***user*** (see [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names))

| ***role*** (see [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names))

}

The [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement assigns privileges and roles to MySQL user accounts and roles. There are several aspects to the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement, described under the following topics:

[GRANT General Overview](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-overview)

[Object Quoting Guidelines](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-quoting)

[Account Names](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-accounts)

[Privileges Supported by MySQL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-privileges)

[Global Privileges](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-global-privileges)

[Database Privileges](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-database-privileges)

[Table Privileges](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-table-privileges)

[Column Privileges](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-column-privileges)

[Stored Routine Privileges](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-routine-privileges)

[Proxy User Privileges](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-proxy-privileges)

[Granting Roles](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-roles)

[The **AS** Clause and Privilege Restrictions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-as)

[Other Account Characteristics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-other-characteristics)

[MySQL and Standard SQL Versions of GRANT](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant-mysql-vs-standard-sql)

##### GRANT General Overview

The [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement enables system administrators to grant privileges and roles, which can be granted to user accounts and roles. These syntax restrictions apply:

[**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) cannot mix granting both privileges and roles in the same statement. A given [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement must grant either privileges or roles.

The **ON** clause distinguishes whether the statement grants privileges or roles:

With **ON**, the statement grants privileges.

Without **ON**, the statement grants roles.

It is permitted to assign both privileges and roles to an account, but you must use separate [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statements, each with syntax appropriate to what is to be granted.

For more information about roles, see [Section 6.2.10, “Using Roles”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#roles).

To grant a privilege with [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant), you must have the [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) privilege, and you must have the privileges that you are granting. (Alternatively, if you have the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the grant tables in the **mysql** system schema, you can grant any account any privilege.) When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) additionally requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

[**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) either succeeds for all named users and roles or rolls back and has no effect if any error occurs. The statement is written to the binary log only if it succeeds for all named users and roles.

The [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) statement is related to [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) and enables administrators to remove account privileges. See [Section 13.7.1.8, “REVOKE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke).

Each account name uses the format described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names). Each role name uses the format described in [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names). For example:

GRANT ALL ON db1.\* TO 'jeffrey'@'localhost';

GRANT 'role1', 'role2' TO 'user1'@'localhost', 'user2'@'localhost';

GRANT SELECT ON world.\* TO 'role3';

The host name part of the account or role name, if omitted, defaults to **'%'**.

Normally, a database administrator first uses [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) to create an account and define its nonprivilege characteristics such as its password, whether it uses secure connections, and limits on access to server resources, then uses [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) to define its privileges. [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) may be used to change the nonprivilege characteristics of existing accounts. For example:

CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY '***password***';

GRANT ALL ON db1.\* TO 'jeffrey'@'localhost';

GRANT SELECT ON db2.invoice TO 'jeffrey'@'localhost';

ALTER USER 'jeffrey'@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 90;

From the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) program, [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) responds with **Query OK, 0 rows affected** when executed successfully. To determine what privileges result from the operation, use [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants). See [Section 13.7.7.21, “SHOW GRANTS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants).

**Important**

Under some circumstances, [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) may be recorded in server logs or on the client side in a history file such as ~/.mysql\_history, which means that cleartext passwords may be read by anyone having read access to that information. For information about the conditions under which this occurs for the server logs and how to control it, see [Section 6.1.2.3, “Passwords and Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-logging). For similar information about client-side logging, see [Section 4.5.1.3, “mysql Client Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-logging).

[**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) supports host names up to 255 characters long (60 characters prior to MySQL 8.0.17). User names can be up to 32 characters. Database, table, column, and routine names can be up to 64 characters.

**Warning**

Do not attempt to change the permissible length for user names by altering the ***mysql.user*** system table. Doing so results in unpredictable behavior which may even make it impossible for users to log in to the MySQL server. Never alter the structure of tables in the **mysql** system schema in any manner except by means of the procedure described in [Section 2.11, “Upgrading MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading).

##### Object Quoting Guidelines

Several objects within [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statements are subject to quoting, although quoting is optional in many cases: Account, role, database, table, column, and routine names. For example, if a ***user\_name*** or ***host\_name*** value in an account name is legal as an unquoted identifier, you need not quote it. However, quotation marks are necessary to specify a ***user\_name*** string containing special characters (such as **-**), or a ***host\_name*** string containing special characters or wildcard characters such as **%** (for example, **'test-user'@'%.com'**). Quote the user name and host name separately.

To specify quoted values:

Quote database, table, column, and routine names as identifiers.

Quote user names and host names as identifiers or as strings.

Quote passwords as strings.

For string-quoting and identifier-quoting guidelines, see [Section 9.1.1, “String Literals”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#string-literals), and [Section 9.2, “Schema Object Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#identifiers).

The **\_** and **%** wildcards are permitted when specifying database names in [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statements that grant privileges at the database level (**GRANT ... ON *db\_name*.\***). This means, for example, that to use a **\_** character as part of a database name, specify it as **\\_** in the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement, to prevent the user from being able to access additional databases matching the wildcard pattern (for example, **GRANT ... ON `foo\\_bar`.\* TO ...**).

When a database name not is used to grant privileges at the database level, but as a qualifier for granting privileges to some other object such as a table or routine (for example, **GRANT ... ON *db\_name*.*tbl\_name***), wildcard characters are treated as normal characters.

##### Account Names

A ***user*** value in a [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement indicates a MySQL account to which the statement applies. To accommodate granting rights to users from arbitrary hosts, MySQL supports specifying the ***user*** value in the form **'*user\_name*'@'*host\_name*'**.

You can specify wildcards in the host name. For example, **'*user\_name*'@'%.example.com'** applies to ***user\_name*** for any host in the **example.com** domain, and **'*user\_name*'@'198.51.100.%'** applies to ***user\_name*** for any host in the **198.51.100** class C subnet.

The simple form **'*user\_name*'** is a synonym for **'*user\_name*'@'%'**.

MySQL does not support wildcards in user names. To refer to an anonymous user, specify an account with an empty user name with the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement:

GRANT ALL ON test.\* TO ''@'localhost' ...;

In this case, any user who connects from the local host with the correct password for the anonymous user is permitted access, with the privileges associated with the anonymous-user account.

For additional information about user name and host name values in account names, see [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names).

**Warning**

If you permit local anonymous users to connect to the MySQL server, you should also grant privileges to all local users as **'*user\_name*'@'localhost'**. Otherwise, the anonymous user account for **localhost** in the **mysql.user** system table is used when named users try to log in to the MySQL server from the local machine. For details, see [Section 6.2.6, “Access Control, Stage 1: Connection Verification”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#connection-access).

To determine whether this issue applies to you, execute the following query, which lists any anonymous users:

SELECT Host, User FROM mysql.user WHERE User='';

To avoid the problem just described, delete the local anonymous user account using this statement:

DROP USER ''@'localhost';

##### Privileges Supported by MySQL

The following tables summarize the permissible static and dynamic ***priv\_type*** privilege types that can be specified for the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) and [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) statements, and the levels at which each privilege can be granted. For additional information about each privilege, see [Section 6.2.2, “Privileges Provided by MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#privileges-provided). For information about the differences between static and dynamic privileges, see [Static Versus Dynamic Privileges](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#static-dynamic-privileges).

**Table 13.11 Permissible Static Privileges for GRANT and REVOKE**

| **Privilege** | **Meaning and Grantable Levels** |
| --- | --- |
| [**ALL [PRIVILEGES]**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_all) | Grant all privileges at specified access level except [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) and [**PROXY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_proxy). |
| [**ALTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter) | Enable use of [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table). Levels: Global, database, table. |
| [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine) | Enable stored routines to be altered or dropped. Levels: Global, database, routine. |
| [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create) | Enable database and table creation. Levels: Global, database, table. |
| [**CREATE ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-role) | Enable role creation. Level: Global. |
| [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine) | Enable stored routine creation. Levels: Global, database. |
| [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-tablespace) | Enable tablespaces and log file groups to be created, altered, or dropped. Level: Global. |
| [**CREATE TEMPORARY TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-temporary-tables) | Enable use of [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table). Levels: Global, database. |
| [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) | Enable use of [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user), [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user), [**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user), and [**REVOKE ALL PRIVILEGES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke). Level: Global. |
| [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-view) | Enable views to be created or altered. Levels: Global, database, table. |
| [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) | Enable use of [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete). Level: Global, database, table. |
| [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) | Enable databases, tables, and views to be dropped. Levels: Global, database, table. |
| [**DROP ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop-role) | Enable roles to be dropped. Level: Global. |
| [**EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_event) | Enable use of events for the Event Scheduler. Levels: Global, database. |
| [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute) | Enable the user to execute stored routines. Levels: Global, database, routine. |
| [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file) | Enable the user to cause the server to read or write files. Level: Global. |
| [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) | Enable privileges to be granted to or removed from other accounts. Levels: Global, database, table, routine, proxy. |
| [**INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_index) | Enable indexes to be created or dropped. Levels: Global, database, table. |
| [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) | Enable use of [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert). Levels: Global, database, table, column. |
| [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_lock-tables) | Enable use of [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) on tables for which you have the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) privilege. Levels: Global, database. |
| [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process) | Enable the user to see all processes with [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist). Level: Global. |
| [**PROXY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_proxy) | Enable user proxying. Level: From user to user. |
| [**REFERENCES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_references) | Enable foreign key creation. Levels: Global, database, table, column. |
| [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) | Enable use of [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) operations. Level: Global. |
| [**REPLICATION CLIENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-client) | Enable the user to ask where source or replica servers are. Level: Global. |
| [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) | Enable replicas to read binary log events from the source. Level: Global. |
| [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) | Enable use of [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). Levels: Global, database, table, column. |
| [**SHOW DATABASES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-databases) | Enable [**SHOW DATABASES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-databases) to show all databases. Level: Global. |
| [**SHOW VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-view) | Enable use of [**SHOW CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-view). Levels: Global, database, table. |
| [**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_shutdown) | Enable use of [**mysqladmin shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin). Level: Global. |
| [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) | Enable use of other administrative operations such as [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to), [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to), [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill), [**PURGE BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs), [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable), and [**mysqladmin debug**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) command. Level: Global. |
| [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_trigger) | Enable trigger operations. Levels: Global, database, table. |
| [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) | Enable use of [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update). Levels: Global, database, table, column. |
| [**USAGE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_usage) | Synonym for “no privileges” |

**Table 13.12 Permissible Dynamic Privileges for GRANT and REVOKE**

| **Privilege** | **Meaning and Grantable Levels** |
| --- | --- |
| [**APPLICATION\_PASSWORD\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_application-password-admin) | Enable dual password administration. Level: Global. |
| [**AUDIT\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_audit-admin) | Enable audit log configuration. Level: Global. |
| [**BACKUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_backup-admin) | Enable backup administration. Level: Global. |
| [**BINLOG\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_binlog-admin) | Enable binary log control. Level: Global. |
| [**BINLOG\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_binlog-encryption-admin) | Enable activation and deactivation of binary log encryption. Level: Global. |
| [**CLONE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_clone-admin) | Enable clone administration. Level: Global. |
| [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) | Enable connection limit/restriction control. Level: Global. |
| [**ENCRYPTION\_KEY\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_encryption-key-admin) | Enable **InnoDB** key rotation. Level: Global. |
| [**FIREWALL\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_firewall-admin) | Enable firewall rule administration, any user. Level: Global. |
| [**FIREWALL\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_firewall-user) | Enable firewall rule administration, self. Level: Global. |
| [**FLUSH\_OPTIMIZER\_COSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-optimizer-costs) | Enable optimizer cost reloading. Level: Global. |
| [**FLUSH\_STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-status) | Enable status indicator flushing. Level: Global. |
| [**FLUSH\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-tables) | Enable table flushing. Level: Global. |
| [**FLUSH\_USER\_RESOURCES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-user-resources) | Enable user-resource flushing. Level: Global. |
| [**GROUP\_REPLICATION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_group-replication-admin) | Enable Group Replication control. Level: Global. |
| [**INNODB\_REDO\_LOG\_ENABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_innodb-redo-log-enable) | Enable or disable redo logging. Level: Global. |
| [**INNODB\_REDO\_LOG\_ARCHIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_innodb-redo-log-archive) | Enable redo log archiving administration. Level: Global. |
| [**NDB\_STORED\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_ndb-stored-user) | Enable sharing of user or role between SQL nodes (NDB Cluster). Level: Global. |
| [**PERSIST\_RO\_VARIABLES\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_persist-ro-variables-admin) | Enable persisting read-only system variables. Level: Global. |
| [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) | Act as the **PRIVILEGE\_CHECKS\_USER** for a replication channel. Level: Global. |
| [**REPLICATION\_SLAVE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave-admin) | Enable regular replication control. Level: Global. |
| [**RESOURCE\_GROUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_resource-group-admin) | Enable resource group administration. Level: Global. |
| [**RESOURCE\_GROUP\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_resource-group-user) | Enable resource group administration. Level: Global. |
| [**ROLE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_role-admin) | Enable roles to be granted or revoked, use of **WITH ADMIN OPTION**. Level: Global. |
| [**SESSION\_VARIABLES\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_session-variables-admin) | Enable setting restricted session system variables. Level: Global. |
| [**SET\_USER\_ID**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_set-user-id) | Enable setting non-self **DEFINER** values. Level: Global. |
| [**SHOW\_ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-routine) | Enable access to stored routine definitions. Level: Global. |
| [**SYSTEM\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-user) | Designate account as system account. Level: Global. |
| [**SYSTEM\_VARIABLES\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-variables-admin) | Enable modifying or persisting global system variables. Level: Global. |
| [**TABLE\_ENCRYPTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_table-encryption-admin) | Enable overriding default encryption settings. Level: Global. |
| [**VERSION\_TOKEN\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_version-token-admin) | Enable use of Version Tokens UDFs. Level: Global. |
| [**XA\_RECOVER\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_xa-recover-admin) | Enable [**XA RECOVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#xa-statements) execution. Level: Global. |

A trigger is associated with a table. To create or drop a trigger, you must have the [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_trigger) privilege for the table, not the trigger.

In [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statements, the [**ALL [PRIVILEGES]**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_all) or [**PROXY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_proxy) privilege must be named by itself and cannot be specified along with other privileges. [**ALL [PRIVILEGES]**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_all) stands for all privileges available for the level at which privileges are to be granted except for the [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) and [**PROXY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_proxy) privileges.

MySQL account information is stored in the tables of the **mysql** system schema. For additional details, consult [Section 6.2, “Access Control and Account Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#access-control), which discusses the **mysql** system schema and the access control system extensively.

If the grant tables hold privilege rows that contain mixed-case database or table names and the [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) system variable is set to a nonzero value, [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) cannot be used to revoke these privileges. It is necessary in such cases to manipulate the grant tables directly. ([**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) does not create such rows when [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) is set, but such rows might have been created prior to setting that variable. The [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) setting can only be configured at server startup.)

Privileges can be granted at several levels, depending on the syntax used for the **ON** clause. For [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke), the same **ON** syntax specifies which privileges to remove.

For the global, database, table, and routine levels, [**GRANT ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) assigns only the privileges that exist at the level you are granting. For example, **GRANT ALL ON *db\_name*.\*** is a database-level statement, so it does not grant any global-only privileges such as [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file). Granting [**ALL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_all) does not assign the [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) or [**PROXY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_proxy) privilege.

The ***object\_type*** clause, if present, should be specified as **TABLE**, **FUNCTION**, or **PROCEDURE** when the following object is a table, a stored function, or a stored procedure.

The privileges that a user holds for a database, table, column, or routine are formed additively as the logical [**OR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_or) of the account privileges at each of the privilege levels, including the global level. It is not possible to deny a privilege granted at a higher level by absence of that privilege at a lower level. For example, this statement grants the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privileges globally:

GRANT SELECT, INSERT ON \*.\* TO u1;

The globally granted privileges apply to all databases, tables, and columns, even though not granted at any of those lower levels.

As of MySQL 8.0.16, it is possible to explicitly deny a privilege granted at the global level by revoking it for particular databases, if the [**partial\_revokes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_partial_revokes) system variable is enabled:

GRANT SELECT, INSERT, UPDATE ON \*.\* TO u1;

REVOKE INSERT, UPDATE ON db1.\* FROM u1;

The result of the preceding statements is that [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) applies globally to all tables, whereas [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) apply globally except to tables in **db1**. Account access to **db1** is read only.

Details of the privilege-checking procedure are presented in [Section 6.2.7, “Access Control, Stage 2: Request Verification”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#request-access).

If you are using table, column, or routine privileges for even one user, the server examines table, column, and routine privileges for all users and this slows down MySQL a bit. Similarly, if you limit the number of queries, updates, or connections for any users, the server must monitor these values.

MySQL enables you to grant privileges on databases or tables that do not exist. For tables, the privileges to be granted must include the [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create) privilege. This behavior is by design, and is intended to enable the database administrator to prepare user accounts and privileges for databases or tables that are to be created at a later time.

**Important**

MySQL does not automatically revoke any privileges when you drop a database or table. However, if you drop a routine, any routine-level privileges granted for that routine are revoked.

##### Global Privileges

Global privileges are administrative or apply to all databases on a given server. To assign global privileges, use **ON \*.\*** syntax:

GRANT ALL ON \*.\* TO 'someuser'@'somehost';

GRANT SELECT, INSERT ON \*.\* TO 'someuser'@'somehost';

The [**CREATE TABLESPACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-tablespace), [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user), [**FILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_file), [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process), [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload), [**REPLICATION CLIENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-client), [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave), [**SHOW DATABASES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-databases), [**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_shutdown), and [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) static privileges are administrative and can only be granted globally.

Dynamic privileges are all global and can only be granted globally.

Other privileges can be granted globally or at more specific levels.

The effect of [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) granted at the global level differs for static and dynamic privileges:

[**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) granted for any static global privilege applies to all static global privileges.

[**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) granted for any dynamic privilege applies only to that dynamic privilege.

**GRANT ALL** at the global level grants all static global privileges and all currently registered dynamic privileges. A dynamic privilege registered subsequent to execution of the **GRANT** statement is not granted retroactively to any account.

MySQL stores global privileges in the **mysql.user** system table.

##### Database Privileges

Database privileges apply to all objects in a given database. To assign database-level privileges, use **ON *db\_name*.\*** syntax:

GRANT ALL ON mydb.\* TO 'someuser'@'somehost';

GRANT SELECT, INSERT ON mydb.\* TO 'someuser'@'somehost';

If you use **ON \*** syntax (rather than **ON \*.\***), privileges are assigned at the database level for the default database. An error occurs if there is no default database.

The [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create), [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop), [**EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_event), [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option), [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_lock-tables), and [**REFERENCES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_references) privileges can be specified at the database level. Table or routine privileges also can be specified at the database level, in which case they apply to all tables or routines in the database.

MySQL stores database privileges in the **mysql.db** system table.

##### Table Privileges

Table privileges apply to all columns in a given table. To assign table-level privileges, use **ON *db\_name.tbl\_name*** syntax:

GRANT ALL ON mydb.mytbl TO 'someuser'@'somehost';

GRANT SELECT, INSERT ON mydb.mytbl TO 'someuser'@'somehost';

If you specify ***tbl\_name*** rather than ***db\_name.tbl\_name***, the statement applies to ***tbl\_name*** in the default database. An error occurs if there is no default database.

The permissible ***priv\_type*** values at the table level are [**ALTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter), [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-view), [**CREATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete), [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop), [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option), [**INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_index), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert), [**REFERENCES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_references), [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select), [**SHOW VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-view), [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_trigger), and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update).

Table-level privileges apply to base tables and views. They do not apply to tables created with [**CREATE TEMPORARY TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-temporary-table), even if the table names match. For information about **TEMPORARY** table privileges, see [Section 13.1.20.2, “CREATE TEMPORARY TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-temporary-table).

MySQL stores table privileges in the **mysql.tables\_priv** system table.

##### Column Privileges

Column privileges apply to single columns in a given table. Each privilege to be granted at the column level must be followed by the column or columns, enclosed within parentheses.

GRANT SELECT (col1), INSERT (col1, col2) ON mydb.mytbl TO 'someuser'@'somehost';

The permissible ***priv\_type*** values for a column (that is, when you use a ***column\_list*** clause) are [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert), [**REFERENCES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_references), [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select), and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update).

MySQL stores column privileges in the **mysql.columns\_priv** system table.

##### Stored Routine Privileges

The [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine), [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine), [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute), and [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) privileges apply to stored routines (procedures and functions). They can be granted at the global and database levels. Except for [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine), these privileges can be granted at the routine level for individual routines.

GRANT CREATE ROUTINE ON mydb.\* TO 'someuser'@'somehost';

GRANT EXECUTE ON PROCEDURE mydb.myproc TO 'someuser'@'somehost';

The permissible ***priv\_type*** values at the routine level are [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine), [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute), and [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option). [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine) is not a routine-level privilege because you must have the privilege at the global or database level to create a routine in the first place.

MySQL stores routine-level privileges in the **mysql.procs\_priv** system table.

##### Proxy User Privileges

The [**PROXY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_proxy) privilege enables one user to be a proxy for another. The proxy user impersonates or takes the identity of the proxied user; that is, it assumes the privileges of the proxied user.

GRANT PROXY ON 'localuser'@'localhost' TO 'externaluser'@'somehost';

When [**PROXY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_proxy) is granted, it must be the only privilege named in the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement, and the only permitted **WITH** option is **WITH GRANT OPTION**.

Proxying requires that the proxy user authenticate through a plugin that returns the name of the proxied user to the server when the proxy user connects, and that the proxy user have the **PROXY** privilege for the proxied user. For details and examples, see [Section 6.2.18, “Proxy Users”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#proxy-users).

MySQL stores proxy privileges in the **mysql.proxies\_priv** system table.

##### Granting Roles

**GRANT** syntax without an **ON** clause grants roles rather than individual privileges. A role is a named collection of privileges; see [Section 6.2.10, “Using Roles”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#roles). For example:

GRANT 'role1', 'role2' TO 'user1'@'localhost', 'user2'@'localhost';

Each role to be granted must exist, as well as each user account or role to which it is to be granted. As of MySQL 8.0.16, roles cannot be granted to anonymous users.

Granting a role does not automatically cause the role to be active. For information about role activation and inactivation, see [Activating Roles](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#roles-activating).

These privileges are required to grant roles:

If you have the [**ROLE\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_role-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege), you can grant or revoke any role to users or roles.

If you were granted a role with a [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement that includes the **WITH ADMIN OPTION** clause, you become able to grant that role to other users or roles, or revoke it from other users or roles, as long as the role is active at such time as you subsequently grant or revoke it. This includes the ability to use **WITH ADMIN OPTION** itself.

To grant a role that has the [**SYSTEM\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-user) privilege, you must have the [**SYSTEM\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-user) privilege.

It is possible to create circular references with [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant). For example:

CREATE USER 'u1', 'u2';

CREATE ROLE 'r1', 'r2';

GRANT 'u1' TO 'u1'; -- simple loop: u1 => u1

GRANT 'r1' TO 'r1'; -- simple loop: r1 => r1

GRANT 'r2' TO 'u2';

GRANT 'u2' TO 'r2'; -- mixed user/role loop: u2 => r2 => u2

Circular grant references are permitted but add no new privileges or roles to the grantee because a user or role already has its privileges and roles.

##### The AS Clause and Privilege Restrictions

As of MySQL 8.0.16, [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) has an **AS *user* [WITH ROLE]** clause that specifies additional information about the privilege context to use for statement execution. This syntax is visible at the SQL level, although its primary purpose is to enable uniform replication across all nodes of grantor privilege restrictions imposed by partial revokes, by causing those restrictions to appear in the binary log. For information about partial revokes, see [Section 6.2.12, “Privilege Restriction Using Partial Revokes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#partial-revokes).

When the **AS *user*** clause is specified, statement execution takes into account any privilege restrictions associated with the named user, including all roles specified by **WITH ROLE**, if present. The result is that the privileges actually granted by the statement may be reduced relative to those specified.

These conditions apply to the **AS *user*** clause:

**AS** has an effect only when the named ***user*** has privilege restrictions (which implies that the [**partial\_revokes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_partial_revokes) system variable is enabled).

If **WITH ROLE** is given, all roles named must be granted to the named ***user***.

The named ***user*** should be a MySQL account specified as **'*user\_name*'@'*host\_name*'**, [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user), or [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user). The current user may be named together with **WITH ROLE** for the case that the executing user wants [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) to execute with a set of roles applied that may differ from the roles active within the current session.

**AS** cannot be used to gain privileges not possessed by the user who executes the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement. The executing user must have at least the privileges to be granted, but the **AS** clause can only restrict the privileges granted, not escalate them.

With respect to the privileges to be granted, **AS** cannot specify a user/role combination that has more privileges (fewer restrictions) than the user who executes the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement. The **AS** user/role combination is permitted to have more privileges than the executing user, but only if the statement does not grant those additional privileges.

**AS** is supported only for granting global privileges (**ON \*.\***).

**AS** is not supported for [**PROXY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_proxy) grants.

The following example illustrates the effect of the **AS** clause. Create a user **u1** that has some global privileges, as well as restrictions on those privileges:

CREATE USER u1;

GRANT SELECT, INSERT, UPDATE, DELETE ON \*.\* TO u1;

REVOKE INSERT, UPDATE ON schema1.\* FROM u1;

REVOKE SELECT ON schema2.\* FROM u1;

Also create a role **r1** that lifts some of the privilege restrictions and grant the role to **u1**:

CREATE ROLE r1;

GRANT INSERT ON schema1.\* TO r1;

GRANT SELECT ON schema2.\* TO r1;

GRANT r1 TO u1;

Now, using an account that has no privilege restrictions of its own, grant to multiple users the same set of global privileges, but each with different restrictions imposed by the **AS** clause, and check which privileges are actually granted.

The [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement here has no **AS** clause, so the privileges granted are exactly those specified:

mysql> **CREATE USER u2;**

mysql> **GRANT SELECT, INSERT, UPDATE ON \*.\* TO u2;**

mysql> **SHOW GRANTS FOR u2;**

+-------------------------------------------------+

| Grants for u2@% |

+-------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE ON \*.\* TO `u2`@`%` |

+-------------------------------------------------+

The [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement here has an **AS** clause, so the privileges granted are those specified but with the restrictions from **u1** applied:

mysql> **CREATE USER u3;**

mysql> **GRANT SELECT, INSERT, UPDATE ON \*.\* TO u3 AS u1;**

mysql> **SHOW GRANTS FOR u3;**

+----------------------------------------------------+

| Grants for u3@% |

+----------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE ON \*.\* TO `u3`@`%` |

| REVOKE INSERT, UPDATE ON `schema1`.\* FROM `u3`@`%` |

| REVOKE SELECT ON `schema2`.\* FROM `u3`@`%` |

+----------------------------------------------------+

As mentioned previously, the **AS** clause can only add privilege restrictions; it cannot escalate privileges. Thus, although **u1** has the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privilege, that is not included in the privileges granted because the statement does not specify granting [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete).

The **AS** clause for the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement here makes the role **r1** active for **u1**. That role lifts some of the restrictions on **u1**. Consequently, the privileges granted have some restrictions, but not so many as for the previous [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement:

mysql> **CREATE USER u4;**

mysql> **GRANT SELECT, INSERT, UPDATE ON \*.\* TO u4 AS u1 WITH ROLE r1;**

mysql> **SHOW GRANTS FOR u4;**

+-------------------------------------------------+

| Grants for u4@% |

+-------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE ON \*.\* TO `u4`@`%` |

| REVOKE UPDATE ON `schema1`.\* FROM `u4`@`%` |

+-------------------------------------------------+

If a [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement includes an **AS *user*** clause, privilege restrictions on the user who executes the statement are ignored (rather than applied as they would be in the absence of an **AS** clause).

##### Other Account Characteristics

The optional **WITH** clause is used to enable a user to grant privileges to other users. The **WITH GRANT OPTION** clause gives the user the ability to give to other users any privileges the user has at the specified privilege level.

To grant the [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) privilege to an account without otherwise changing its privileges, do this:

GRANT USAGE ON \*.\* TO 'someuser'@'somehost' WITH GRANT OPTION;

Be careful to whom you give the [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) privilege because two users with different privileges may be able to combine privileges!

You cannot grant another user a privilege which you yourself do not have; the [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) privilege enables you to assign only those privileges which you yourself possess.

Be aware that when you grant a user the [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) privilege at a particular privilege level, any privileges the user possesses (or may be given in the future) at that level can also be granted by that user to other users. Suppose that you grant a user the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege on a database. If you then grant the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege on the database and specify **WITH GRANT OPTION**, that user can give to other users not only the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege, but also [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert). If you then grant the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege to the user on the database, the user can grant [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert), [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select), and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update).

For a nonadministrative user, you should not grant the [**ALTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter) privilege globally or for the **mysql** system schema. If you do that, the user can try to subvert the privilege system by renaming tables!

For additional information about security risks associated with particular privileges, see [Section 6.2.2, “Privileges Provided by MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#privileges-provided).

##### MySQL and Standard SQL Versions of GRANT

The biggest differences between the MySQL and standard SQL versions of [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) are:

MySQL associates privileges with the combination of a host name and user name and not with only a user name.

Standard SQL does not have global or database-level privileges, nor does it support all the privilege types that MySQL supports.

MySQL does not support the standard SQL **UNDER** privilege.

Standard SQL privileges are structured in a hierarchical manner. If you remove a user, all privileges the user has been granted are revoked. This is also true in MySQL if you use [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user). See [Section 13.7.1.5, “DROP USER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user).

In standard SQL, when you drop a table, all privileges for the table are revoked. In standard SQL, when you revoke a privilege, all privileges that were granted based on that privilege are also revoked. In MySQL, privileges can be dropped with [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user) or [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) statements.

In MySQL, it is possible to have the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege for only some of the columns in a table. In this case, you can still execute [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) statements on the table, provided that you insert values only for those columns for which you have the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege. The omitted columns are set to their implicit default values if strict SQL mode is not enabled. In strict mode, the statement is rejected if any of the omitted columns have no default value. (Standard SQL requires you to have the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege on all columns.) For information about strict SQL mode and implicit default values, see [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode), and [Section 11.6, “Data Type Default Values”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#data-type-defaults).

#### 13.7.1.7 RENAME USER Statement

RENAME USER ***old\_user*** TO ***new\_user***

[, ***old\_user*** TO ***new\_user***] ...

The [**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user) statement renames existing MySQL accounts. An error occurs for old accounts that do not exist or new accounts that already exist.

To use [**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user), you must have the global [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege, or the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the **mysql** system schema. When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user) additionally requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

As of MySQL 8.0.22, [**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user) fails with an error if any account to be renamed is named as the **DEFINER** attribute for any stored object. (That is, the statement fails if renaming an account would cause a stored object to become orphaned.) To perform the operation anyway, you must have the [**SET\_USER\_ID**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_set-user-id) privilege; in this case, the statement succeeds with a warning rather than failing with an error. For additional information, including how to identify which objects name a given account as the **DEFINER** attribute, see [Orphan Stored Objects](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security-orphan-objects).

Each account name uses the format described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names). For example:

RENAME USER 'jeffrey'@'localhost' TO 'jeff'@'127.0.0.1';

The host name part of the account name, if omitted, defaults to **'%'**.

[**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user) causes the privileges held by the old user to be those held by the new user. However, [**RENAME USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-user) does not automatically drop or invalidate databases or objects within them that the old user created. This includes stored programs or views for which the **DEFINER** attribute names the old user. Attempts to access such objects may produce an error if they execute in definer security context. (For information about security context, see [Section 25.6, “Stored Object Access Control”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-objects-security).)

The privilege changes take effect as indicated in [Section 6.2.13, “When Privilege Changes Take Effect”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#privilege-changes).

#### 13.7.1.8 REVOKE Statement

REVOKE

***priv\_type*** [(***column\_list***)]

[, ***priv\_type*** [(***column\_list***)]] ...

ON [***object\_type***] ***priv\_level***

FROM ***user\_or\_role*** [, ***user\_or\_role***] ...

REVOKE ALL [PRIVILEGES], GRANT OPTION

FROM ***user\_or\_role*** [, ***user\_or\_role***] ...

REVOKE PROXY ON ***user\_or\_role***

FROM ***user\_or\_role*** [, ***user\_or\_role***] ...

REVOKE ***role*** [, ***role*** ] ...

FROM ***user\_or\_role*** [, ***user\_or\_role*** ] ...

***user\_or\_role***: {

***user*** (see [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names))

| ***role*** (see [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names).

}

The [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) statement enables system administrators to revoke privileges and roles, which can be revoked from user accounts and roles.

For details on the levels at which privileges exist, the permissible ***priv\_type***, ***priv\_level***, and ***object\_type*** values, and the syntax for specifying users and passwords, see [Section 13.7.1.6, “GRANT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant).

For information about roles, see [Section 6.2.10, “Using Roles”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#roles).

When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) or privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege), in addition to any other required privileges described in the following discussion.

[**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) either succeeds for all named users and roles or rolls back and has no effect if any error occurs. The statement is written to the binary log only if it succeeds for all named users and roles.

Each account name uses the format described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names). Each role name uses the format described in [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names). For example:

REVOKE INSERT ON \*.\* FROM 'jeffrey'@'localhost';

REVOKE 'role1', 'role2' FROM 'user1'@'localhost', 'user2'@'localhost';

REVOKE SELECT ON world.\* FROM 'role3';

The host name part of the account or role name, if omitted, defaults to **'%'**.

To use the first [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) syntax, you must have the [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_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 users or roles:

REVOKE ALL PRIVILEGES, GRANT OPTION

FROM ***user\_or\_role*** [, ***user\_or\_role***] ...

**REVOKE ALL PRIVILEGES, GRANT OPTION** does not revoke any roles.

To use this [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) syntax, you must have the global [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege, or the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the **mysql** system schema.

The syntax for which the [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) keyword is followed by one or more role names takes a **FROM** clause indicating one or more users or roles from which to revoke the roles.

Roles named in the [**mandatory\_roles**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_mandatory_roles) system variable value cannot be revoked.

A revoked role immediately affects any user account from which it was revoked, such that within any current session for the account, its privileges are adjusted for the next statement executed.

Revoking a role revokes the role itself, not the privileges that it represents. Suppose that an account is granted a role that includes a given privilege, and is also granted the privilege explicitly or another role that includes the privilege. In this case, the account still possesses that privilege if the first role is revoked. For example, if an account is granted two roles that each include [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select), the account still can select after either role is revoked.

**REVOKE ALL ON \*.\*** (at the global level) revokes all granted static global privileges and all granted dynamic privileges.

User accounts and roles from which privileges and roles are to be revoked must exist, but the privileges and roles to be revoked need not be currently granted to them.

A revoked privilege that is granted but not known to the server is revoked with a warning. This situtation can occur for dynamic privileges. For example, a dynamic privilege can be granted while the component that registers it is installed, but if that component is subsequently uninstalled, the privilege becomes unregistered, although accounts that possess the privilege still possess it and it can be revoked from them.

[**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) removes privileges, but does not remove rows from the **mysql.user** system table. To remove a user account entirely, use [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user). See [Section 13.7.1.5, “DROP USER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user).

If the grant tables hold privilege rows that contain mixed-case database or table names and the [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) system variable is set to a nonzero value, [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) cannot be used to revoke these privileges. It is necessary in such cases to manipulate the grant tables directly. ([**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) does not create such rows when [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) is set, but such rows might have been created prior to setting the variable. The [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) setting can only be configured when initializing the server.)

When successfully executed from the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) program, [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke) responds with **Query OK, 0 rows affected**. To determine what privileges remain after the operation, use [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants). See [Section 13.7.7.21, “SHOW GRANTS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants).

#### 13.7.1.9 SET DEFAULT ROLE Statement

SET DEFAULT ROLE

{NONE | ALL | ***role*** [, ***role*** ] ...}

TO ***user*** [, ***user*** ] ...

For each ***user*** named immediately after the **TO** keyword, this statement defines which roles become active when the user connects to the server and authenticates, or when the user executes the [**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) statement during a session.

[**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) is alternative syntax for [**ALTER USER ... DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) (see [Section 13.7.1.1, “ALTER USER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user)). However, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) can set the default for only a single user, whereas [**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) can set the default for multiple users. On the other hand, you can specify **CURRENT\_USER** as the user name for the [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) statement, whereas you cannot for [**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role).

[**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) requires these privileges:

Setting the default roles for another user requires the global [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege, or the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the **mysql.default\_roles** system table.

Setting the default roles for yourself requires no special privileges, as long as the roles you want as the default have been granted to you.

Each role name uses the format described in [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names). For example:

SET DEFAULT ROLE 'admin', 'developer' TO 'joe'@'10.0.0.1';

The host name part of the role name, if omitted, defaults to **'%'**.

The clause following the **DEFAULT ROLE** keywords permits these values:

**NONE**: Set the default to **NONE** (no roles).

**ALL**: Set the default to all roles granted to the account.

***role* [, *role* ] ...**: Set the default to the named roles, which must exist and be granted to the account at the time [**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) is executed.

**Note**

[**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) and [**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) are different statements:

[**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) defines which account roles to activate by default within account sessions.

[**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) sets the active roles within the current session to the current account default roles.

For role usage examples, see [Section 6.2.10, “Using Roles”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#roles).

#### 13.7.1.10 SET PASSWORD Statement

SET PASSWORD [FOR ***user***] ***auth\_option***

[REPLACE '***current\_auth\_string***']

[RETAIN CURRENT PASSWORD]

***auth\_option***: {

= '***auth\_string***'

| TO RANDOM

}

The [**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) statement assigns a password to a MySQL user account. The password may be either explicitly specified in the statement or randomly generated by MySQL. The statement may also include a password-verification clause that specifies the account current password to be replaced, and a clause that manages whether an account has a secondary password. **'*auth\_string*'** and **'*current\_auth\_string*'** each represent a cleartext (unencrypted) password.

**Note**

Rather than using [**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) to assign passwords, [**ALTER USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-user) is the preferred statement for account alterations, including assigning passwords. For example:

ALTER USER ***user*** IDENTIFIED BY '***auth\_string***';

**Note**

Clauses for random password generation, password verification, and secondary passwords apply only to accounts that use an authentication plugin that stores credentials internally to MySQL. For accounts that use a plugin that performs authentication against a credentials system that is external to MySQL, password management must be handled externally against that system as well. For more information about internal credentials storage, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

The **REPLACE '*current\_auth\_string*'** clause performs password verification and is available as of MySQL 8.0.13. If given:

**REPLACE** specifies the account current password to be replaced, as a cleartext (unencrypted) string.

The clause must be given if password changes for the account are required to specify the current password, as verification that the user attempting to make the change actually knows the current password.

The clause is optional if password changes for the account may but need not specify the current password.

The statement fails if the clause is given but does not match the current password, even if the clause is optional.

**REPLACE** can be specified only when changing the account password for the current user.

For more information about password verification by specifying the current password, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

The **RETAIN CURRENT PASSWORD** clause implements dual-password capability and is available as of MySQL 8.0.14. If given:

**RETAIN CURRENT PASSWORD** retains an account current password as its secondary password, replacing any existing secondary password. The new password becomes the primary password, but clients can use the account to connect to the server using either the primary or secondary password. (Exception: If the new password specified by the [**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) statement is empty, the secondary password becomes empty as well, even if **RETAIN CURRENT PASSWORD** is given.)

If you specify **RETAIN CURRENT PASSWORD** for an account that has an empty primary password, the statement fails.

If an account has a secondary password and you change its primary password without specifying **RETAIN CURRENT PASSWORD**, the secondary password remains unchanged.

For more information about use of dual passwords, see [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

[**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) permits these ***auth\_option*** syntaxes:

**= '*auth\_string*'**

Assigns the account the given literal password.

**TO RANDOM**

Assigns the account a password randomly generated by MySQL. The statement also returns the cleartext password in a result set to make it available to the user or application executing the statement.

For details about the result set and characteristics of randomly generated passwords, see [Random Password Generation](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#random-password-generation).

Random password generation is available as of MySQL 8.0.18.

**Important**

Under some circumstances, [**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) may be recorded in server logs or on the client side in a history file such as ~/.mysql\_history, which means that cleartext passwords may be read by anyone having read access to that information. For information about the conditions under which this occurs for the server logs and how to control it, see [Section 6.1.2.3, “Passwords and Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-logging). For similar information about client-side logging, see [Section 4.5.1.3, “mysql Client Logging”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-logging).

[**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) can be used with or without a **FOR** clause that explicitly names a user account:

With a **FOR *user*** clause, the statement sets the password for the named account, which must exist:

SET PASSWORD FOR 'jeffrey'@'localhost' = '***auth\_string***';

With no **FOR *user*** clause, the statement sets the password for the current user:

SET PASSWORD = '***auth\_string***';

Any client who connects to the server using a nonanonymous account can change the password for that account. (In particular, you can change your own password.) To see which account the server authenticated you as, invoke the [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) function:

SELECT CURRENT\_USER();

If a **FOR *user*** clause is given, the account name uses the format described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names). For example:

SET PASSWORD FOR 'bob'@'%.example.org' = '***auth\_string***';

The host name part of the account name, if omitted, defaults to **'%'**.

[**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) interprets the string as a cleartext string, passes it to the authentication plugin associated with the account, and stores the result returned by the plugin in the account row in the **mysql.user** system table. (The plugin is given the opportunity to hash the value into the encryption format it expects. The plugin may use the value as specified, in which case no hashing occurs.)

Setting the password for a named account (with a **FOR** clause) requires the [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_update) privilege for the **mysql** system schema. Setting the password for yourself (for a nonanonymous account with no **FOR** clause) requires no special privileges.

Statements that modify secondary passwords require these privileges:

The [**APPLICATION\_PASSWORD\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_application-password-admin) privilege is required to use the **RETAIN CURRENT PASSWORD** clause for [**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) statements that apply to your own account. The privilege is required to manipulate your own secondary password because most users require only one password.

If an account is to be permitted to manipulate secondary passwords for all accounts, it should be granted the [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-user) privilege rather than [**APPLICATION\_PASSWORD\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_application-password-admin).

When the [**read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_read_only) system variable is enabled, [**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) requires the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege), in addition to any other required privileges.

For additional information about setting passwords and authentication plugins, see [Section 6.2.14, “Assigning Account Passwords”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#assigning-passwords), and [Section 6.2.17, “Pluggable Authentication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#pluggable-authentication).

#### 13.7.1.11 SET ROLE Statement

SET ROLE {

DEFAULT

| NONE

| ALL

| ALL EXCEPT ***role*** [, ***role*** ] ...

| ***role*** [, ***role*** ] ...

}

[**SET ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) modifies the current user's effective privileges within the current session by specifying which of its granted roles are active. Granted roles include those granted explicitly to the user and those named in the [**mandatory\_roles**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_mandatory_roles) system variable value.

Examples:

SET ROLE DEFAULT;

SET ROLE 'role1', 'role2';

SET ROLE ALL;

SET ROLE ALL EXCEPT 'role1', 'role2';

Each role name uses the format described in [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names). The host name part of the role name, if omitted, defaults to **'%'**.

Privileges that the user has been granted directly (rather than through roles) remain unaffected by changes to the active roles.

The statement permits these role specifiers:

**DEFAULT**: Activate the account default roles. Default roles are those specified with [**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role).

When a user connects to the server and authenticates successfully, the server determines which roles to activate as the default roles. If the [**activate\_all\_roles\_on\_login**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_activate_all_roles_on_login) system variable is enabled, the server activates all granted roles. Otherwise, the server executes [**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) implicitly. The server activates only default roles that can be activated. The server writes warnings to its error log for default roles that cannot be activated, but the client receives no warnings.

If a user executes [**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) during a session, an error occurs if any default role cannot be activated (for example, if it does not exist or is not granted to the user). In this case, the current active roles are not changed.

**NONE**: Set the active roles to **NONE** (no active roles).

**ALL**: Activate all roles granted to the account.

**ALL EXCEPT *role* [, *role* ] ...**: Activate all roles granted to the account except those named. The named roles need not exist or be granted to the account.

***role* [, *role* ] ...**: Activate the named roles, which must be granted to the account.

**Note**

[**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) and [**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) are different statements:

[**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) defines which account roles to activate by default within account sessions.

[**SET ROLE DEFAULT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) sets the active roles within the current session to the current account default roles.

For role usage examples, see [Section 6.2.10, “Using Roles”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#roles).

### 13.7.2 Resource Group Management Statements

[13.7.2.1 ALTER RESOURCE GROUP Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-resource-group)

[13.7.2.2 CREATE RESOURCE GROUP Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-resource-group)

[13.7.2.3 DROP RESOURCE GROUP Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-resource-group)

[13.7.2.4 SET RESOURCE GROUP Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-resource-group)

MySQL supports creation and management of resource groups, and permits assigning threads running within the server to particular groups so that threads execute according to the resources available to the group. This section describes the SQL statements available for resource group management. For general discussion of the resource group capability, see [Section 5.1.16, “Resource Groups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#resource-groups).

#### 13.7.2.1 ALTER RESOURCE GROUP Statement

ALTER RESOURCE GROUP ***group\_name***

[VCPU [=] ***vcpu\_spec*** [, ***vcpu\_spec***] ...]

[THREAD\_PRIORITY [=] ***N***]

[ENABLE|DISABLE [FORCE]]

***vcpu\_spec***: {***N*** | ***M*** - ***N***}

[**ALTER RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-resource-group) is used for resource group management (see [Section 5.1.16, “Resource Groups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#resource-groups)). This statement alters modifiable attributes of an existing resource group. It requires the [**RESOURCE\_GROUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_resource-group-admin) privilege.

***group\_name*** identifies which resource group to alter. If the group does not exist, an error occurs.

The attributes for CPU affinity, priority, and whether the group is enabled can be modified with [**ALTER RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-resource-group). These attributes are specified the same way as described for [**CREATE RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-resource-group) (see [Section 13.7.2.2, “CREATE RESOURCE GROUP Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-resource-group)). Only the attributes specified are altered. Unspecified attributes retain their current values.

The **FORCE** modifier is used with **DISABLE**. It determines statement behavior if the resource group has any threads assigned to it:

If **FORCE** is not given, existing threads in the group continue to run until they terminate, but new threads cannot be assigned to the group.

If **FORCE** is given, existing threads in the group are moved to their respective default group (system threads to **SYS\_default**, user threads to **USR\_default**).

The name and type attributes are set at group creation time and cannot be modified thereafter with [**ALTER RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-resource-group).

Examples:

Alter a group CPU affinity:

ALTER RESOURCE GROUP rg1 VCPU = 0-63;

Alter a group thread priority:

ALTER RESOURCE GROUP rg2 THREAD\_PRIORITY = 5;

Disable a group, moving any threads assigned to it to the default groups:

ALTER RESOURCE GROUP rg3 DISABLE FORCE;

Resource group management is local to the server on which it occurs. [**ALTER RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-resource-group) statements are not written to the binary log and are not replicated.

#### 13.7.2.2 CREATE RESOURCE GROUP Statement

CREATE RESOURCE GROUP ***group\_name***

TYPE = {SYSTEM|USER}

[VCPU [=] ***vcpu\_spec*** [, ***vcpu\_spec***] ...]

[THREAD\_PRIORITY [=] ***N***]

[ENABLE|DISABLE]

***vcpu\_spec***: {***N*** | ***M*** - ***N***}

[**CREATE RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-resource-group) is used for resource group management (see [Section 5.1.16, “Resource Groups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#resource-groups)). This statement creates a new resource group and assigns its initial attribute values. It requires the [**RESOURCE\_GROUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_resource-group-admin) privilege.

***group\_name*** identifies which resource group to create. If the group already exists, an error occurs.

The **TYPE** attribute is required. It should be **SYSTEM** for a system resource group, **USER** for a user resource group. The group type affects permitted **THREAD\_PRIORITY** values, as described later.

The **VCPU** attribute indicates the CPU affinity; that is, the set of virtual CPUs the group can use:

If **VCPU** is not given, the resource group has no CPU affinity and can use all available CPUs.

If **VCPU** is given, the attribute value is a list of comma-separated CPU numbers or ranges:

Each number must be an integer in the range from 0 to the number of CPUs − 1. For example, on a system with 64 CPUs, the number can range from 0 to 63.

A range is given in the form ***M*** − ***N***, where ***M*** is less than or equal to ***N*** and both numbers are in the CPU range.

If a CPU number is an integer outside the permitted range or is not an integer, an error occurs.

Example **VCPU** specifiers (these are all equivalent):

VCPU = 0,1,2,3,9,10

VCPU = 0-3,9-10

VCPU = 9,10,0-3

VCPU = 0,10,1,9,3,2

The **THREAD\_PRIORITY** attribute indicates the priority for threads assigned to the group:

If **THREAD\_PRIORITY** is not given, the default priority is 0.

If **THREAD\_PRIORITY** is given, the attribute value must be in the range from -20 (highest priority) to 19 (lowest priority). The priority for system resource groups must be in the range from -20 to 0. The priority for user resource groups must be in the range from 0 to 19. Use of different ranges for system and user groups ensures that user threads never have a higher priority than system threads.

**ENABLE** and **DISABLE** specify that the resource group is initially enabled or disabled. If neither is specified, the group is enabled by default. A disabled group cannot have threads assigned to it.

Examples:

Create an enabled user group that has a single CPU and the lowest priority:

CREATE RESOURCE GROUP rg1

TYPE = USER

VCPU = 0

THREAD\_PRIORITY = 19;

Create a disabled system group that has no CPU affinity (can use all CPUs) and the highest priority:

CREATE RESOURCE GROUP rg2

TYPE = SYSTEM

THREAD\_PRIORITY = -20

DISABLE;

Resource group management is local to the server on which it occurs. [**CREATE RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-resource-group) statements are not written to the binary log and are not replicated.

#### 13.7.2.3 DROP RESOURCE GROUP Statement

DROP RESOURCE GROUP ***group\_name*** [FORCE]

[**DROP RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-resource-group) is used for resource group management (see [Section 5.1.16, “Resource Groups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#resource-groups)). This statement drops a resource group. It requires the [**RESOURCE\_GROUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_resource-group-admin) privilege.

***group\_name*** identifies which resource group to drop. If the group does not exist, an error occurs.

The **FORCE** modifier determines statement behavior if the resource group has any threads assigned to it:

If **FORCE** is not given and any threads are assigned to the group, an error occurs.

If **FORCE** is given, existing threads in the group are moved to their respective default group (system threads to **SYS\_default**, user threads to **USR\_default**).

Examples:

Drop a group, failing if the group contains any threads:

DROP RESOURCE GROUP rg1;

Drop a group and move existing threads to the default groups:

DROP RESOURCE GROUP rg2 FORCE;

Resource group management is local to the server on which it occurs. [**DROP RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-resource-group) statements are not written to the binary log and are not replicated.

#### 13.7.2.4 SET RESOURCE GROUP Statement

SET RESOURCE GROUP ***group\_name***

[FOR ***thread\_id*** [, ***thread\_id***] ...]

[**SET RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-resource-group) is used for resource group management (see [Section 5.1.16, “Resource Groups”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#resource-groups)). This statement assigns threads to a resource group. It requires the [**RESOURCE\_GROUP\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_resource-group-admin) or [**RESOURCE\_GROUP\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_resource-group-user) privilege.

***group\_name*** identifies which resource group to be assigned. Any ***thread\_id*** values indicate threads to assign to the group. Thread IDs can be determined from the Performance Schema [**threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-threads-table) table. If the resource group or any named thread ID does not exist, an error occurs.

With no **FOR** clause, the statement assigns the current thread for the session to the resource group.

With a **FOR** clause that names thread IDs, the statement assigns those threads to the resource group.

For attempts to assign a system thread to a user resource group or a user thread to a system resource group, a warning occurs.

Examples:

Assign the current session thread to a group:

SET RESOURCE GROUP rg1;

Assign the named threads to a group:

SET RESOURCE GROUP rg2 FOR 14, 78, 4;

Resource group management is local to the server on which it occurs. [**SET RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-resource-group) statements are not written to the binary log and are not replicated.

An alternative to [**SET RESOURCE GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-resource-group) is the [**RESOURCE\_GROUP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints-resource-group) optimizer hint, which assigns individual statements to a resource group. See [Section 8.9.3, “Optimizer Hints”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-hints).

### 13.7.3 Table Maintenance Statements

[13.7.3.1 ANALYZE TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table)

[13.7.3.2 CHECK TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table)

[13.7.3.3 CHECKSUM TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#checksum-table)

[13.7.3.4 OPTIMIZE TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table)

[13.7.3.5 REPAIR TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table)

#### 13.7.3.1 ANALYZE TABLE Statement

ANALYZE [NO\_WRITE\_TO\_BINLOG | LOCAL]

TABLE ***tbl\_name*** [, ***tbl\_name***] ...

ANALYZE [NO\_WRITE\_TO\_BINLOG | LOCAL]

TABLE ***tbl\_name***

UPDATE HISTOGRAM ON ***col\_name*** [, ***col\_name***] ...

[WITH ***N*** BUCKETS]

ANALYZE [NO\_WRITE\_TO\_BINLOG | LOCAL]

TABLE ***tbl\_name***

DROP HISTOGRAM ON ***col\_name*** [, ***col\_name***] ...

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) generates table statistics:

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) without either **HISTOGRAM** clause performs a key distribution analysis and stores the distribution for the named table or tables. For **MyISAM** tables, [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) for key distribution analysis is equivalent to using [**myisamchk --analyze**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) with the **UPDATE HISTOGRAM** clause generates histogram statistics for the named table columns and stores them in the data dictionary. Only one table name is permitted for this syntax.

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) with the **DROP HISTOGRAM** clause removes histogram statistics for the named table columns from the data dictionary. Only one table name is permitted for this syntax.

This statement requires [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privileges for the table.

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) works with **InnoDB**, **NDB**, and **MyISAM** tables. It does not work with views.

If the [**innodb\_read\_only**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_read_only) system variable is enabled, [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) may fail because it cannot update statistics tables in the data dictionary, which use **InnoDB**. For [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) operations that update the key distribution, failure may occur even if the operation updates the table itself (for example, if it is a **MyISAM** table). To obtain the updated distribution statistics, set [**information\_schema\_stats\_expiry=0**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_information_schema_stats_expiry).

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) is supported for partitioned tables, and you can use **ALTER TABLE ... ANALYZE PARTITION** to analyze one or more partitions; for more information, see [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), and [Section 24.3.4, “Maintenance of Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-maintenance).

During the analysis, the table is locked with a read lock for **InnoDB** and **MyISAM**.

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) removes the table from the table definition cache, which requires a flush lock. If there are long running statements or transactions still using the table, subsequent statements and transactions must wait for those operations to finish before the flush lock is released. Because [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) itself typically finishes quickly, it may not be apparent that delayed transactions or statements involving the same table are due to the remaining flush lock.

By default, the server writes [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) statements to the binary log so that they replicate to replicas. To suppress logging, specify the optional **NO\_WRITE\_TO\_BINLOG** keyword or its alias **LOCAL**.

[ANALYZE TABLE Output](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table-output)

[Key Distribution Analysis](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table-key-distribution-analysis)

[Histogram Statistics Analysis](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table-histogram-statistics-analysis)

[Other Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table-other-considerations)

##### ANALYZE TABLE Output

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) returns a result set with the columns shown in the following table.

| **Column** | **Value** |
| --- | --- |
| **Table** | The table name |
| **Op** | **analyze** or **histogram** |
| **Msg\_type** | **status**, **error**, **info**, **note**, or **warning** |
| **Msg\_text** | An informational message |

##### Key Distribution Analysis

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) without either **HISTOGRAM** clause performs a key distribution analysis and stores the distribution for the table or tables. Any existing histogram statistics remain unaffected.

If the table has not changed since the last key distribution analysis, the table is not analyzed again.

MySQL uses the stored key distribution to decide the order in which tables should be joined for joins 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.

To check the stored key distribution cardinality, use the [**SHOW INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index) statement or the **INFORMATION\_SCHEMA** [**STATISTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-statistics-table) table. See [Section 13.7.7.22, “SHOW INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index), and [Section 26.3.34, “The INFORMATION\_SCHEMA STATISTICS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-statistics-table).

For **InnoDB** tables, [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) determines index cardinality by performing random dives on each of the index trees and updating index cardinality estimates accordingly. Because these are only estimates, repeated runs of [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) could produce different numbers. This makes [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) fast on **InnoDB** tables but not 100% accurate because it does not take all rows into account.

You can make the [statistics](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_statistics) collected by [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) more precise and more stable by enabling [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent), as explained in [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats). When [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is enabled, it is important to run [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) after major changes to index column data, as statistics are not recalculated periodically (such as after a server restart).

If [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is enabled, you can change the number of random dives by modifying the [**innodb\_stats\_persistent\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent_sample_pages) system variable. If [**innodb\_stats\_persistent**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_persistent) is disabled, modify [**innodb\_stats\_transient\_sample\_pages**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_stats_transient_sample_pages) instead.

For more information about key distribution analysis in **InnoDB**, see [Section 15.8.10.1, “Configuring Persistent Optimizer Statistics Parameters”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-persistent-stats), and [Section 15.8.10.3, “Estimating ANALYZE TABLE Complexity for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-analyze-table-complexity).

MySQL uses index cardinality estimates in join optimization. If a join is not optimized in the right way, try running [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table). In the few cases that [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) does not produce values good enough for your particular tables, you can use **FORCE INDEX** with your queries to force the use of a particular index, or set the [**max\_seeks\_for\_key**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_seeks_for_key) system variable to ensure that MySQL prefers index lookups over table scans. See [Section B.3.5, “Optimizer-Related Issues”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#optimizer-issues).

##### Histogram Statistics Analysis

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) with the **HISTOGRAM** clauses enables management of histogram statistics for table column values. For information about histogram statistics, see [Section 8.9.6, “Optimizer Statistics”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#optimizer-statistics).

These histogram operations are available:

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) with an **UPDATE HISTOGRAM** clause generates histogram statistics for the named table columns and stores them in the data dictionary. Only one table name is permitted for this syntax.

The optional **WITH *N* BUCKETS** clauses specifies the number of buckets for the histogram. The value of ***N*** must be an integer in the range from 1 to 1024. If this clause is omitted, the number of buckets is 100.

[**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) with a **DROP HISTOGRAM** clause removes histogram statistics for the named table columns from the data dictionary. Only one table name is permitted for this syntax.

Stored histogram management statements affect only the named columns. Consider these statements:

ANALYZE TABLE t UPDATE HISTOGRAM ON c1, c2, c3 WITH 10 BUCKETS;

ANALYZE TABLE t UPDATE HISTOGRAM ON c1, c3 WITH 10 BUCKETS;

ANALYZE TABLE t DROP HISTOGRAM ON c2;

The first statement updates the histograms for columns **c1**, **c2**, and **c3**, replacing any existing histograms for those columns. The second statement updates the histograms for **c1** and **c3**, leaving the **c2** histogram unaffected. The third statement removes the histogram for **c2**, leaving those for **c1** and **c3** unaffected.

Histogram generation is not supported for encrypted tables (to avoid exposing data in the statistics) or **TEMPORARY** tables.

Histogram generation applies to columns of all data types except geometry types (spatial data) and [**JSON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#json).

Histograms can be generated for stored and virtual generated columns.

Histograms cannot be generated for columns that are covered by single-column unique indexes.

Histogram management statements attempt to perform as much of the requested operation as possible, and report diagnostic messages for the remainder. For example, if an **UPDATE HISTOGRAM** statement names multiple columns, but some of them do not exist or have an unsupported data type, histograms are generated for the other columns, and messages are produced for the invalid columns.

Histograms are affected by these DDL statements:

[**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) removes histograms for columns in the dropped table.

[**DROP DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-database) removes histograms for any table in the dropped database because the statement drops all tables in the database.

[**RENAME TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#rename-table) does not remove histograms. Instead, it renames histograms for the renamed table to be associated with the new table name.

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements that remove or modify a column remove histograms for that column.

[**ALTER TABLE ... CONVERT TO CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) removes histograms for character columns because they are affected by the change of character set. Histograms for noncharacter columns remain unaffected.

The [**histogram\_generation\_max\_mem\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_histogram_generation_max_mem_size) system variable controls the maximum amount of memory available for histogram generation. The global and session values may be set at runtime.

Changing the global [**histogram\_generation\_max\_mem\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_histogram_generation_max_mem_size) value requires privileges sufficient to set global system variables. Changing the session [**histogram\_generation\_max\_mem\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_histogram_generation_max_mem_size) value requires privileges sufficient to set restricted session system variables. See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

If the estimated amount of data to be read into memory for histogram generation exceeds the limit defined by [**histogram\_generation\_max\_mem\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_histogram_generation_max_mem_size), MySQL samples the data rather than reading all of it into memory. Sampling is evenly distributed over the entire table. MySQL uses **SYSTEM** sampling, which is a page-level sampling method.

The **sampling-rate** value in the **HISTOGRAM** column of [**INFORMATION\_SCHEMA.COLUMN\_STATISTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-column-statistics-table) table can be queried to determine the fraction of data that was sampled to create the histogram. The **sampling-rate** is a number between 0.0 and 1.0. A value of 1 means that all of the data was read (no sampling).

The following example demonstrates sampling. To ensure that the amount of data exceeds the [**histogram\_generation\_max\_mem\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_histogram_generation_max_mem_size) limit for the purpose of the example, the limit is set to a low value (2000000 bytes) prior to generating histogram statistics for the **birth\_date** column of the **employees** table.

mysql> **SET histogram\_generation\_max\_mem\_size = 2000000;**

mysql> **USE employees;**

mysql> **ANALYZE TABLE employees UPDATE HISTOGRAM ON birth\_date WITH 16 BUCKETS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: employees.employees

Op: histogram

Msg\_type: status

Msg\_text: Histogram statistics created for column 'birth\_date'.

mysql> **SELECT HISTOGRAM->>'$."sampling-rate"'**

**FROM INFORMATION\_SCHEMA.COLUMN\_STATISTICS**

**WHERE TABLE\_NAME = "employees"**

**AND COLUMN\_NAME = "birth\_date";**

+---------------------------------+

| HISTOGRAM->>'$."sampling-rate"' |

+---------------------------------+

| 0.0491431208869665 |

+---------------------------------+

A **sampling-rate** value of 0.0491431208869665 means that approximately 4.9% of the data from the **birth\_date** column was read into memory for generating histogram statistics.

As of MySQL 8.0.19, the **InnoDB** storage engine provides its own sampling implementation for data stored in **InnoDB** tables. The default sampling implementation used by MySQL when storage engines do not provide their own requires a full table scan, which is costly for large tables. The **InnoDB** sampling implementation improves sampling performance by avoiding full table scans.

The **sampled\_pages\_read** and **sampled\_pages\_skipped** [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) counters can be used to monitor sampling of **InnoDB** data pages. (For general [**INNODB\_METRICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table) counter usage information, see [Section 26.4.23, “The INFORMATION\_SCHEMA INNODB\_METRICS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-metrics-table).)

The following example demonstrates sampling counter usage, which requires enabling the counters prior to generating histogram statistics.

mysql> **SET GLOBAL innodb\_monitor\_enable = 'sampled%';**

mysql> **USE employees;**

mysql> **ANALYZE TABLE employees UPDATE HISTOGRAM ON birth\_date WITH 16 BUCKETS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: employees.employees

Op: histogram

Msg\_type: status

Msg\_text: Histogram statistics created for column 'birth\_date'.

mysql> **USE INFORMATION\_SCHEMA;**

mysql> **SELECT NAME, COUNT FROM INNODB\_METRICS WHERE NAME LIKE 'sampled%'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: sampled\_pages\_read

COUNT: 43

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NAME: sampled\_pages\_skipped

COUNT: 843

This formula approximates a sampling rate based on the sampling counter data:

sampling rate = sampled\_page\_read/(sampled\_pages\_read + sampled\_pages\_skipped)

A sampling rate based on sampling counter data is roughly the same as the **sampling-rate** value in the **HISTOGRAM** column of [**INFORMATION\_SCHEMA.COLUMN\_STATISTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-column-statistics-table) table.

For information about memory allocations performed for histogram generation, monitor the Performance Schema **memory/sql/histograms** instrument. See [Section 27.12.20.10, “Memory Summary Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-memory-summary-tables).

##### Other Considerations

**ANALYZE TABLE** clears table statistics from the [**INFORMATION\_SCHEMA.INNODB\_TABLESTATS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablestats-table) table and sets the **STATS\_INITIALIZED** column to **Uninitialized**. Statistics are collected again the next time the table is accessed.

#### 13.7.3.2 CHECK TABLE Statement

CHECK TABLE ***tbl\_name*** [, ***tbl\_name***] ... [***option***] ...

***option***: {

FOR UPGRADE

| QUICK

| FAST

| MEDIUM

| EXTENDED

| CHANGED

}

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) checks a table or tables for errors. [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) can also check views for problems, such as tables that are referenced in the view definition that no longer exist.

To check a table, you must have some privilege for it.

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) works for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine), [**ARCHIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#archive-storage-engine), and [**CSV**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#csv-storage-engine) tables.

Before running [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) on **InnoDB** tables, see [CHECK TABLE Usage Notes for InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table-innodb).

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) is supported for partitioned tables, and you can use **ALTER TABLE ... CHECK PARTITION** to check one or more partitions; for more information, see [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), and [Section 24.3.4, “Maintenance of Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-maintenance).

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) ignores virtual generated columns that are not indexed.

[CHECK TABLE Output](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table-output)

[Checking Version Compatibility](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table-version-compatibility)

[Checking Data Consistency](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table-data-consistency)

[CHECK TABLE Usage Notes for InnoDB Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table-innodb)

[CHECK TABLE Usage Notes for MyISAM Tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table-myisam)

##### CHECK TABLE Output

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) returns a result set with the columns shown in the following table.

| **Column** | **Value** |
| --- | --- |
| **Table** | The table name |
| **Op** | Always **check** |
| **Msg\_type** | **status**, **error**, **info**, **note**, or **warning** |
| **Msg\_text** | An informational message |

The statement might produce many rows of information for each checked table. The last row has a **Msg\_type** value of **status** and the **Msg\_text** normally should be **OK**. **Table is already up to date** means that the storage engine for the table indicated that there was no need to check the table.

##### Checking Version Compatibility

The **FOR UPGRADE** option checks whether the named tables are compatible with the current version of MySQL. With **FOR UPGRADE**, the server checks each table to determine whether there have been any incompatible changes in any of the table's data types or indexes since the table was created. If not, the check succeeds. Otherwise, if there is a possible incompatibility, the server runs a full check on the table (which might take some time).

Incompatibilities might occur because the storage format for a data type has changed or because its sort order has changed. Our aim is to avoid these changes, but occasionally they are necessary to correct problems that would be worse than an incompatibility between releases.

**FOR UPGRADE** discovers these incompatibilities:

The indexing order for end-space in [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns for **InnoDB** and **MyISAM** tables changed between MySQL 4.1 and 5.0.

The storage method of the new [**DECIMAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#fixed-point-types) data type changed between MySQL 5.0.3 and 5.0.5.

Changes are sometimes made to character sets or collations that require table indexes to be rebuilt. For details about such changes, see [Section 2.11.4, “Changes in MySQL 8.0”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#upgrading-from-previous-series). For information about rebuilding tables, see [Section 2.11.13, “Rebuilding or Repairing Tables or Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\installing.html#rebuilding-tables).

MySQL 8.0 does not support the 2-digit [**YEAR(2)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year) data type permitted in older versions of MySQL. For tables containing [**YEAR(2)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year) columns, [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) recommends [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), which converts 2-digit [**YEAR(2)**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year) columns to 4-digit [**YEAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#year) columns.

Trigger creation time is maintained.

A table is reported as needing a rebuild if it contains old temporal columns in pre-5.6.4 format ([**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time), [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), and [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) columns without support for fractional seconds precision) and the [**avoid\_temporal\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_avoid_temporal_upgrade) system variable is disabled. This helps the MySQL upgrade procedure detect and upgrade tables containing old temporal columns. If [**avoid\_temporal\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_avoid_temporal_upgrade) is enabled, **FOR UPGRADE** ignores the old temporal columns present in the table; consequently, the upgrade procedure does not upgrade them.

To check for tables that contain such temporal columns and need a rebuild, disable [**avoid\_temporal\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_avoid_temporal_upgrade) before executing [**CHECK TABLE ... FOR UPGRADE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table).

Warnings are issued for tables that use nonnative partitioning because nonnative partitioning is removed in MySQL 8.0. See [Chapter 24, *Partitioning*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html).

##### Checking Data Consistency

The following table shows the other check options that can be given. These options are passed to the storage engine, which may use or ignore them.

| **Type** | **Meaning** |
| --- | --- |
| **QUICK** | Do not scan the rows to check for incorrect links. Applies to **InnoDB** and **MyISAM** tables and views. |
| **FAST** | Check only tables that have not been closed properly. Ignored for **InnoDB**; applies only to **MyISAM** tables and views. |
| **CHANGED** | Check only tables that have been changed since the last check or that have not been closed properly. Ignored for **InnoDB**; applies only to **MyISAM** tables and views. |
| **MEDIUM** | Scan rows to verify that deleted links are valid. This also calculates a key checksum for the rows and verifies this with a calculated checksum for the keys. Ignored for **InnoDB**; applies only to **MyISAM** tables and views. |
| **EXTENDED** | Do a full key lookup for all keys for each row. This ensures that the table is 100% consistent, but takes a long time. Ignored for **InnoDB**; applies only to **MyISAM** tables and views. |

You can combine check options, as in the following example that does a quick check on the table to determine whether it was closed properly:

CHECK TABLE test\_table FAST QUICK;

**Note**

If [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) finds no problems with a table that is marked as “corrupted” or “not closed properly”, [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) may remove the mark.

If a table is corrupted, the problem is most likely in the indexes and not in the data part. All of the preceding check types check the indexes thoroughly and should thus find most errors.

To check a table that you assume is okay, use no check options or the **QUICK** option. The latter should be used when you are in a hurry and can take the very small risk that **QUICK** does not find an error in the data file. (In most cases, under normal usage, MySQL should find any error in the data file. If this happens, the table is marked as “corrupted” and cannot be used until it is repaired.)

**FAST** and **CHANGED** are mostly intended to be used from a script (for example, to be executed from **cron**) to check tables periodically. In most cases, **FAST** is to be preferred over **CHANGED**. (The only case when it is not preferred is when you suspect that you have found a bug in the **MyISAM** code.)

**EXTENDED** is to be used only after you have run a normal check but still get errors from a table when MySQL tries to update a row or find a row by key. This is very unlikely if a normal check has succeeded.

Use of [**CHECK TABLE ... EXTENDED**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) might influence execution plans generated by the query optimizer.

Some problems reported by [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) cannot be corrected automatically:

**Found row where the auto\_increment column has the value 0**.

This means that you have a row in the table where the **AUTO\_INCREMENT** index column contains the value 0. (It is possible to create a row where the **AUTO\_INCREMENT** column is 0 by explicitly setting the column to 0 with an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement.)

This is not an error in itself, but could cause trouble if you decide to dump the table and restore it or do an [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) on the table. In this case, the **AUTO\_INCREMENT** column changes value according to the rules of **AUTO\_INCREMENT** columns, which could cause problems such as a duplicate-key error.

To get rid of the warning, execute an [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statement to set the column to some value other than 0.

##### CHECK TABLE Usage Notes for InnoDB Tables

The following notes apply to [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables:

If [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) encounters a corrupt page, the server exits to prevent error propagation (Bug #10132). If the corruption occurs in a secondary index but table data is readable, running [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) can still cause a server exit.

If [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) encounters a corrupted **DB\_TRX\_ID** or **DB\_ROLL\_PTR** field in a clustered index, [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) can cause **InnoDB** to access an invalid undo log record, resulting in an [MVCC](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_mvcc)-related server exit.

If [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) encounters errors in **InnoDB** tables or indexes, it reports an error, and usually marks the index and sometimes marks the table as corrupted, preventing further use of the index or table. Such errors include an incorrect number of entries in a secondary index or incorrect links.

If [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) finds an incorrect number of entries in a secondary index, it reports an error but does not cause a server exit or prevent access to the file.

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) surveys the index page structure, then surveys each key entry. It does not validate the key pointer to a clustered record or follow the path for [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) pointers.

When an **InnoDB** table is stored in its own [.ibd file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file), the first 3 [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) of the **.ibd** file contain header information rather than table or index data. The [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) statement does not detect inconsistencies that affect only the header data. To verify the entire contents of an **InnoDB** **.ibd** file, use the [**innochecksum**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#innochecksum) command.

When running [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) on large **InnoDB** tables, other threads may be blocked during [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) execution. To avoid timeouts, the semaphore wait threshold (600 seconds) is extended by 2 hours (7200 seconds) for [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) operations. If **InnoDB** detects semaphore waits of 240 seconds or more, it starts printing **InnoDB** monitor output to the error log. If a lock request extends beyond the semaphore wait threshold, **InnoDB** aborts the process. To avoid the possibility of a semaphore wait timeout entirely, run [**CHECK TABLE QUICK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) instead of [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table).

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) functionality for **InnoDB** **SPATIAL** indexes includes an R-tree validity check and a check to ensure that the R-tree row count matches the clustered index.

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) supports secondary indexes on virtual generated columns, which are supported by **InnoDB**.

As of MySQL 8.0.14, **InnoDB** supports parallel clustered index reads, which can improve [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) performance. **InnoDB** reads the clustered index twice during a [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) operation. The second read can be performed in parallel. The [**innodb\_parallel\_read\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_parallel_read_threads) session variable must be set to a value greater than 1 for parallel clustered index reads to occur. The default value is 4. The actual number of threads used to perform a parallel clustered index read is determined by the [**innodb\_parallel\_read\_threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_parallel_read_threads) setting or the number of index subtrees to scan, whichever is smaller.

##### CHECK TABLE Usage Notes for MyISAM Tables

The following notes apply to [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables:

[**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) updates key statistics for **MyISAM** tables.

If [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) output does not return **OK** or **Table is already up to date**, you should normally run a repair of the table. See [Section 7.6, “MyISAM Table Maintenance and Crash Recovery”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\backup-and-recovery.html#myisam-table-maintenance).

If none of the [**CHECK TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table) options **QUICK**, **MEDIUM**, or **EXTENDED** are specified, the default check type for dynamic-format **MyISAM** tables is **MEDIUM**. This has the same result as running [**myisamchk --medium-check *tbl\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) on the table. The default check type also is **MEDIUM** for static-format **MyISAM** tables, unless **CHANGED** or **FAST** is specified. In that case, the default is **QUICK**. The row scan is skipped for **CHANGED** and **FAST** because the rows are very seldom corrupted.

#### 13.7.3.3 CHECKSUM TABLE Statement

CHECKSUM TABLE ***tbl\_name*** [, ***tbl\_name***] ... [QUICK | EXTENDED]

[**CHECKSUM TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#checksum-table) reports a [checksum](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_checksum) for the contents of a table. You can use this statement to verify that the contents are the same before and after a backup, rollback, or other operation that is intended to put the data back to a known state.

This statement requires the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for the table.

This statement is not supported for views. If you run [**CHECKSUM TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#checksum-table) against a view, the **Checksum** value is always **NULL**, and a warning is returned.

For a nonexistent table, [**CHECKSUM TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#checksum-table) returns **NULL** and generates a warning.

During the checksum operation, the table is locked with a read lock for **InnoDB** and **MyISAM**.

##### Performance Considerations

By default, the entire table is read row by row and the checksum is calculated. For large tables, this could take a long time, thus you would only perform this operation occasionally. This row-by-row calculation is what you get with the **EXTENDED** clause, with **InnoDB** and all other storage engines other than **MyISAM**, and with **MyISAM** tables not created with the **CHECKSUM=1** clause.

For **MyISAM** tables created with the **CHECKSUM=1** clause, [**CHECKSUM TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#checksum-table) or [**CHECKSUM TABLE ... QUICK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#checksum-table) returns the “live” table checksum that can be returned very fast. If the table does not meet all these conditions, the **QUICK** method returns **NULL**. The **QUICK** method is not supported with **InnoDB** tables. See [Section 13.1.20, “CREATE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) for the syntax of the **CHECKSUM** clause.

The checksum value depends on the table row format. If the row format changes, the checksum also changes. For example, the storage format for temporal types such as [**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time), [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), and [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) changed in MySQL 5.6 prior to MySQL 5.6.5, so if a 5.5 table is upgraded to MySQL 5.6, the checksum value may change.

**Important**

If the checksums for two tables are different, then it is almost certain that the tables are different in some way. However, because the hashing function used by [**CHECKSUM TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#checksum-table) is not guaranteed to be collision-free, there is a slight chance that two tables which are not identical can produce the same checksum.

#### 13.7.3.4 OPTIMIZE TABLE Statement

OPTIMIZE [NO\_WRITE\_TO\_BINLOG | LOCAL]

TABLE ***tbl\_name*** [, ***tbl\_name***] ...

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) reorganizes the physical storage of table data and associated index data, to reduce storage space and improve I/O efficiency when accessing the table. The exact changes made to each table depend on the [storage engine](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_storage_engine) used by that table.

Use [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) in these cases, depending on the type of table:

After doing substantial insert, update, or delete operations on an **InnoDB** table that has its own [.ibd file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) because it was created with the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) option enabled. The table and indexes are reorganized, and disk space can be reclaimed for use by the operating system.

After doing substantial insert, update, or delete operations on columns that are part of a **FULLTEXT** index in an **InnoDB** table. Set the configuration option [**innodb\_optimize\_fulltext\_only=1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_optimize_fulltext_only) first. To keep the index maintenance period to a reasonable time, set the [**innodb\_ft\_num\_word\_optimize**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_ft_num_word_optimize) option to specify how many words to update in the search index, and run a sequence of **OPTIMIZE TABLE** statements until the search index is fully updated.

After deleting a large part of a **MyISAM** or **ARCHIVE** table, or making many changes to a **MyISAM** or **ARCHIVE**table with variable-length rows (tables that have [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob), or [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns). Deleted rows are maintained in a linked list and subsequent [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert) operations reuse old row positions. You can use [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) to reclaim the unused space and to defragment the data file. After extensive changes to a table, this statement may also improve performance of statements that use the table, sometimes significantly.

This statement requires [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privileges for the table.

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) works for [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine), and [**ARCHIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#archive-storage-engine) tables. [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is also supported for dynamic columns of in-memory [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables. It does not work for fixed-width columns of in-memory tables, nor does it work for Disk Data tables. The performance of **OPTIMIZE** on NDB Cluster tables can be tuned using [--ndb-optimization-delay](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#option_mysqld_ndb-optimization-delay), which controls the length of time to wait between processing batches of rows by [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table). For more information, see [Section 23.1.7.11, “Previous NDB Cluster Issues Resolved in NDB Cluster 8.0”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-limitations-resolved).

For NDB Cluster tables, [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) can be interrupted by (for example) killing the SQL thread performing the **OPTIMIZE** operation.

By default, [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) does not work for tables created using any other storage engine and returns a result indicating this lack of support. You can make [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) work for other storage engines by starting [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) with the [--skip-new](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-new) option. In this case, [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is just mapped to [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

This statement does not work with views.

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is supported for partitioned tables. For information about using this statement with partitioned tables and table partitions, see [Section 24.3.4, “Maintenance of Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-maintenance).

By default, the server writes [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) statements to the binary log so that they replicate to replicas. To suppress logging, specify the optional **NO\_WRITE\_TO\_BINLOG** keyword or its alias **LOCAL**.

[OPTIMIZE TABLE Output](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table-output)

[InnoDB Details](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table-innodb-details)

[MyISAM Details](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table-myisam-details)

[Other Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table-other-considerations)

##### OPTIMIZE TABLE Output

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) returns a result set with the columns shown in the following table.

| **Column** | **Value** |
| --- | --- |
| **Table** | The table name |
| **Op** | Always **optimize** |
| **Msg\_type** | **status**, **error**, **info**, **note**, or **warning** |
| **Msg\_text** | An informational message |

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) table catches and throws any errors that occur while copying table statistics from the old file to the newly created file. For example. if the user ID of the owner of the .MYD or .MYI file is different from the user ID of the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process, [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) generates a "cannot change ownership of the file" error unless [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is started by the **root** user.

##### InnoDB Details

For **InnoDB** tables, [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is mapped to [**ALTER TABLE ... FORCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), which rebuilds the table to update index statistics and free unused space in the clustered index. This is displayed in the output of [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) when you run it on an **InnoDB** table, as shown here:

mysql> OPTIMIZE TABLE foo;

+----------+----------+----------+-------------------------------------------------------------------+

| Table | Op | Msg\_type | Msg\_text |

+----------+----------+----------+-------------------------------------------------------------------+

| test.foo | optimize | note | Table does not support optimize, doing recreate + analyze instead |

| test.foo | optimize | status | OK |

+----------+----------+----------+-------------------------------------------------------------------+

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) uses [online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl) for regular and partitioned **InnoDB** tables, which reduces downtime for concurrent DML operations. The table rebuild triggered by [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is completed in place. An exclusive table lock is only taken briefly during the prepare phase and the commit phase of the operation. During the prepare phase, metadata is updated and an intermediate table is created. During the commit phase, table metadata changes are committed.

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) rebuilds the table using the table copy method under the following conditions:

When the [**old\_alter\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_old_alter_table) system variable is enabled.

When the server is started with the [--skip-new](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-new) option.

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) using [online DDL](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-online-ddl) is not supported for **InnoDB** tables that contain **FULLTEXT** indexes. The table copy method is used instead.

**InnoDB** stores data using a page-allocation method and does not suffer from fragmentation in the same way that legacy storage engines (such as **MyISAM**) do. When considering whether or not to run optimize, consider the workload of transactions that your server is expected to process:

Some level of fragmentation is expected. **InnoDB** only fills [pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_page) 93% full, to leave room for updates without having to split pages.

Delete operations might leave gaps that leave pages less filled than desired, which could make it worthwhile to optimize the table.

Updates to rows usually rewrite the data within the same page, depending on the data type and row format, when sufficient space is available. See [Section 15.9.1.5, “How Compression Works for InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-compression-internals) and [Section 15.10, “InnoDB Row Formats”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-row-format).

High-concurrency workloads might leave gaps in indexes over time, as **InnoDB** retains multiple versions of the same data due through its [MVCC](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_mvcc) mechanism. See [Section 15.3, “InnoDB Multi-Versioning”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-multi-versioning).

##### MyISAM Details

For **MyISAM** tables, [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) works as follows:

If the table has deleted or split rows, repair the table.

If the index pages are not sorted, sort them.

If the table's statistics are not up to date (and the repair could not be accomplished by sorting the index), update them.

##### Other Considerations

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is performed online for regular and partitioned **InnoDB** tables. Otherwise, MySQL [locks the table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_table_lock) during the time [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) is running.

[**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) does not sort R-tree indexes, such as spatial indexes on **POINT** columns. (Bug #23578)

#### 13.7.3.5 REPAIR TABLE Statement

REPAIR [NO\_WRITE\_TO\_BINLOG | LOCAL]

TABLE ***tbl\_name*** [, ***tbl\_name***] ...

[QUICK] [EXTENDED] [USE\_FRM]

[**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) repairs a possibly corrupted table, for certain storage engines only.

This statement requires [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) and [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privileges for the table.

Although normally you should never have to run [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table), if disaster strikes, this statement is very likely to get back all your data from a **MyISAM** table. If your tables become corrupted often, try to find the reason for it, to eliminate the need to use [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table). See [Section B.3.3.3, “What to Do If MySQL Keeps Crashing”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\error-handling.html#crashing), and [Section 16.2.4, “MyISAM Table Problems”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-table-problems).

[**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) checks the table to see whether an upgrade is required. If so, it performs the upgrade, following the same rules as [**CHECK TABLE ... FOR UPGRADE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table). See [Section 13.7.3.2, “CHECK TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#check-table), for more information.

**Important**

Make a backup of a table before performing a table repair operation; under some circumstances the operation might cause data loss. Possible causes include but are not limited to file system errors. See [Chapter 7, *Backup and Recovery*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\backup-and-recovery.html).

If the server exits during a [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) operation, it is essential after restarting it that you immediately execute another [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) statement for the table before performing any other operations on it. In the worst case, you might have a new clean index file without information about the data file, and then the next operation you perform could overwrite the data file. This is an unlikely but possible scenario that underscores the value of making a backup first.

In the event that a table on the source becomes corrupted and you run [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) on it, any resulting changes to the original table are not propagated to replicas.

[REPAIR TABLE Storage Engine and Partitioning Support](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "repair-table-support" \o "REPAIR TABLE Storage Engine and Partitioning Support)

[REPAIR TABLE Options](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table-options)

[REPAIR TABLE Output](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table-output)

[Table Repair Considerations](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table-table-repair-considerations)

##### REPAIR TABLE Storage Engine and Partitioning Support

**[REPAIR TABLE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "repair-table" \o "13.7.3.5 REPAIR TABLE Statement)** works for [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine), [**ARCHIVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#archive-storage-engine), and [**CSV**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#csv-storage-engine) tables. For [**MyISAM**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html#myisam-storage-engine) tables, it has the same effect as [**myisamchk --recover *tbl\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) by default. This statement does not work with views.

[**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) is supported for partitioned tables. However, the **USE\_FRM** option cannot be used with this statement on a partitioned table.

You can use **ALTER TABLE ... REPAIR PARTITION** to repair one or more partitions; for more information, see [Section 13.1.9, “ALTER TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), and [Section 24.3.4, “Maintenance of Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-maintenance).

##### REPAIR TABLE Options

**NO\_WRITE\_TO\_BINLOG** or **LOCAL**

By default, the server writes [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) statements to the binary log so that they replicate to replicas. To suppress logging, specify the optional **NO\_WRITE\_TO\_BINLOG** keyword or its alias **LOCAL**.

**QUICK**

If you use the **QUICK** option, [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) tries to repair only the index file, and not the data file. This type of repair is like that done by [**myisamchk --recover --quick**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

**EXTENDED**

If you use the **EXTENDED** option, MySQL creates the index row by row instead of creating one index at a time with sorting. This type of repair is like that done by [**myisamchk --safe-recover**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

**USE\_FRM**

The **USE\_FRM** option is available for use if the .MYI index file is missing or if its header is corrupted. This option tells MySQL not to trust the information in the .MYI file header and to re-create it using information from the data dictionary. This kind of repair cannot be done with [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

**Caution**

Use the **USE\_FRM** option only if you cannot use regular **REPAIR** modes. Telling the server to ignore the .MYI file makes important table metadata stored in the .MYI unavailable to the repair process, which can have deleterious consequences:

The current **AUTO\_INCREMENT** value is lost.

The link to deleted records in the table is lost, which means that free space for deleted records remains unoccupied thereafter.

The .MYI header indicates whether the table is compressed. If the server ignores this information, it cannot tell that a table is compressed and repair can cause change or loss of table contents. This means that **USE\_FRM** should not be used with compressed tables. That should not be necessary, anyway: Compressed tables are read only, so they should not become corrupt.

If you use **USE\_FRM** for a table that was created by a different version of the MySQL server than the one you are currently running, [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) does not attempt to repair the table. In this case, the result set returned by [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) contains a line with a **Msg\_type** value of **error** and a **Msg\_text** value of **Failed repairing incompatible .FRM file**.

If **USE\_FRM** is used, [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) does not check the table to see whether an upgrade is required.

##### REPAIR TABLE Output

**[REPAIR TABLE](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "repair-table" \o "13.7.3.5 REPAIR TABLE Statement)** returns a result set with the columns shown in the following table.

| **Column** | **Value** |
| --- | --- |
| **Table** | The table name |
| **Op** | Always **repair** |
| **Msg\_type** | **status**, **error**, **info**, **note**, or **warning** |
| **Msg\_text** | An informational message |

The [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) statement might produce many rows of information for each repaired table. The last row has a **Msg\_type** value of **status** and **Msg\_test** normally should be **OK**. For a **MyISAM** table, if you do not get **OK**, you should try repairing it with [**myisamchk --safe-recover**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk). ([**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) does not implement all the options of [**myisamchk**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk). With [**myisamchk --safe-recover**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk), you can also use options that [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) does not support, such as [--max-record-length](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#option_myisamchk_max-record-length).)

[**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) table catches and throws any errors that occur while copying table statistics from the old corrupted file to the newly created file. For example. if the user ID of the owner of the .MYD or .MYI file is different from the user ID of the [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) process, [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) generates a "cannot change ownership of the file" error unless [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is started by the **root** user.

##### Table Repair Considerations

[**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) upgrades a table if it contains old temporal columns in pre-5.6.4 format ([**TIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#time), [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime), and [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) columns without support for fractional seconds precision) and the [**avoid\_temporal\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_avoid_temporal_upgrade) system variable is disabled. If [**avoid\_temporal\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_avoid_temporal_upgrade) is enabled, [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) ignores the old temporal columns present in the table and does not upgrade them.

To upgrade tables that contain such temporal columns, disable [**avoid\_temporal\_upgrade**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_avoid_temporal_upgrade) before executing [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table).

You may be able to increase [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) performance by setting certain system variables. See [Section 8.6.3, “Optimizing REPAIR TABLE Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#repair-table-optimization).

### 13.7.4 Component, Plugin, and User-Defined Function Statements

[13.7.4.1 CREATE FUNCTION Statement for User-Defined Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf)

[13.7.4.2 DROP FUNCTION Statement for User-Defined Functions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf)

[13.7.4.3 INSTALL COMPONENT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-component)

[13.7.4.4 INSTALL PLUGIN Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin)

[13.7.4.5 UNINSTALL COMPONENT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-component)

[13.7.4.6 UNINSTALL PLUGIN Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin)

#### 13.7.4.1 CREATE FUNCTION Statement for User-Defined Functions

CREATE [AGGREGATE] FUNCTION ***function\_name***

RETURNS {STRING|INTEGER|REAL|DECIMAL}

SONAME ***shared\_library\_name***

This statement loads the user-defined function (UDF) named ***function\_name***. (**CREATE FUNCTION** is also used to created stored functions; see [Section 13.1.17, “CREATE PROCEDURE and CREATE FUNCTION Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-procedure).)

A user-defined function is a way to extend MySQL with a new function that works like a native (built-in) MySQL function such as [**ABS()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_abs) or [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat). See [Adding a User-Defined Function](https://dev.mysql.com/doc/extending-mysql/8.0/en/adding-udf.html).

***function\_name*** is the name that should be used in SQL statements to invoke the function. The **RETURNS** clause indicates the type of the function's return value. **DECIMAL** is a legal value after **RETURNS**, but currently **DECIMAL** functions return string values and should be written like **STRING** functions.

The **AGGREGATE** keyword, if given, signifies that the UDF is an aggregate (group) function. An aggregate UDF works exactly like a native MySQL aggregate function such as [**SUM()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_sum) or [**COUNT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_count).

***shared\_library\_name*** is the base name of the shared library file containing 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) system variable. For more information, see [Section 5.7.1, “Installing and Uninstalling User-Defined Functions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#udf-loading).

[**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf) requires the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege for the **mysql** system schema because it adds a row to the **mysql.func** system table to register the function.

[**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf) also adds the function to the Performance Schema [**user\_defined\_functions**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-defined-functions-table) table that provides runtime information about installed UDFs. See [Section 27.12.21.8, “The user\_defined\_functions Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-defined-functions-table).

**Note**

Like the **mysql.func** system table, the Performance Schema [**user\_defined\_functions**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-defined-functions-table) table lists UDFs installed using [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf). Unlike the **mysql.func** table, the [**user\_defined\_functions**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-defined-functions-table) table also lists UDFs installed automatically by server components or plugins. This difference makes [**user\_defined\_functions**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-defined-functions-table) preferable to **mysql.func** for checking which UDFs are installed.

During the normal startup sequence, the server loads UDFs registered in the **mysql.func** table. If the server is started with the [--skip-grant-tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-grant-tables) option, UDFs registered in the table are not loaded and are unavailable.

**Note**

To upgrade the shared library associated with a UDF, issue a [**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf) statement, upgrade the shared library, and then issue a [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf) statement. If you upgrade the shared library first and then use [**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf), the server may unexpectedly shut down.

#### 13.7.4.2 DROP FUNCTION Statement for User-Defined Functions

DROP FUNCTION [IF EXISTS] ***function\_name***

This statement drops the user-defined function (UDF) named ***function\_name***. (**DROP FUNCTION** is also used to drop stored functions; see [Section 13.1.29, “DROP PROCEDURE and DROP FUNCTION Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-procedure).)

[**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf) is the complement of [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf). It requires the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privilege for the **mysql** system schema because it removes the row from the **mysql.func** system table that registers the function.

[**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf) also removes the function from the Performance Schema [**user\_defined\_functions**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-defined-functions-table) table that provides runtime information about installed UDFs. See [Section 27.12.21.8, “The user\_defined\_functions Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-defined-functions-table).

During the normal startup sequence, the server loads UDFs registered in the **mysql.func** table. Because [**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf) removes the **mysql.func** row for the dropped function, the server does not load the function during subsequent restarts.

[**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf) cannot be used to drop a UDF that is installed automatically by components or plugins rather than by using [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf). Such a UDF is also dropped automatically, when the component or plugin that installed it is uninstalled.

**Note**

To upgrade the shared library associated with a UDF, issue a [**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf) statement, upgrade the shared library, and then issue a [**CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-function-udf) statement. If you upgrade the shared library first and then use [**DROP FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-function-udf), the server may unexpectedly shut down.

#### 13.7.4.3 INSTALL COMPONENT Statement

INSTALL COMPONENT ***component\_name*** [, ***component\_name*** ] ...

This statement installs one or more components, which become active immediately. A component provides services that are available to the server and other components; see [Section 5.5, “MySQL Components”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#components). [**INSTALL COMPONENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-component) requires the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege for the **mysql.component** system table because it adds a row to that table to register the component.

Example:

INSTALL COMPONENT 'file://component1', 'file://component2';

A component is named using a URN that begins with **file://** and indicates the base name of the library file that implements the component, located in the directory named by the [**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) system variable. Component names do not include any platform-dependent file name suffix such as .so or .dll. (These naming details are subject to change because component name interpretation is itself performed by a service and the component infrastructure makes it possible to replace the default service implementation with alternative implementations.)

If any error occurs, the statement fails and has no effect. For example, this happens if a component name is erroneous, a named component does not exist or is already installed, or component initialization fails.

A loader service handles component loading, which includes adding installed components to the **mysql.component** system table that serves as a registry. For subsequent server restarts, any components listed in **mysql.component** are loaded by the loader service during the startup sequence. This occurs even if the server is started with the [--skip-grant-tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-grant-tables) option.

If a component depends on services not present in the registry and you attempt to install the component without also installing the component or components that provide the services on which it depends, an error occurs:

ERROR 3527 (HY000): Cannot satisfy dependency for service 'component\_a'

required by component 'component\_b'.

To avoid this problem, either install all components in the same statement, or install the dependent component after installing any components on which it depends.

**Note**

For keyring components, do not use [**INSTALL COMPONENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-component). Instead, configure keyring component loading using a manifest file. See [Section 6.4.4.2, “Keyring Component Installation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring-component-installation).

#### 13.7.4.4 INSTALL PLUGIN Statement

INSTALL PLUGIN ***plugin\_name*** SONAME '***shared\_library\_name***'

This statement installs a server plugin. It requires the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_insert) privilege for the **mysql.plugin** system table because it adds a row to that table to register the plugin.

***plugin\_name*** is the name of the plugin as defined in the plugin descriptor structure contained in the library file (see [Plugin Data Structures](https://dev.mysql.com/doc/extending-mysql/8.0/en/plugin-data-structures.html)). 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.

***shared\_library\_name*** is the name of the shared library that contains the plugin code. The name includes the file name extension (for example, libmyplugin.so, libmyplugin.dll, or libmyplugin.dylib).

The shared library must be located in the plugin directory (the directory named by the [**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) system variable). The library must be in the plugin directory itself, not in a subdirectory. By default, [**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) is the plugin directory under the directory named by the **pkglibdir** configuration variable, but it can be changed by setting the value of [**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) is a relative path name, it is taken to be relative to the MySQL base directory (the value of the [**basedir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_basedir) system variable).

[**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) 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. When the server shuts down, it executes the deinitialization function for each plugin that is loaded so that the plugin has a chance to perform any final cleanup.

[**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) also registers the plugin by adding a line that indicates the plugin name and library file name to the **mysql.plugin** system table. During the normal startup sequence, the server loads and initializes plugins registered in **mysql.plugin**. This means that a plugin is installed with [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) only once, not every time the server starts. If the server is started with the [--skip-grant-tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-grant-tables) option, plugins registered in the **mysql.plugin** table are not loaded and are unavailable.

A plugin library can contain multiple plugins. For each of them to be installed, use a separate [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) statement. Each statement names a different plugin, but all of them specify the same library name.

[**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) 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.

For options that control individual plugin loading at server startup, see [Section 5.6.1, “Installing and Uninstalling Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#plugin-loading). If you need to load plugins for a single server startup when the [--skip-grant-tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-grant-tables) option is given (which tells the server not to read system tables), use the [--plugin-load](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_plugin-load) option. See [Section 5.1.7, “Server Command Options”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-options).

To remove a plugin, use the [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) statement.

For additional information about plugin loading, see [Section 5.6.1, “Installing and Uninstalling Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#plugin-loading).

To see what plugins are installed, use the [**SHOW PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins) statement or query the **INFORMATION\_SCHEMA** the [**PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-plugins-table) table.

If you recompile a plugin library and need to reinstall it, you can use either of the following methods:

Use [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) to uninstall all plugins in the library, install the new plugin library file in the plugin directory, and then use [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) to install all plugins in the library. This procedure has the advantage that it can be used without stopping the server. However, if the plugin library contains many plugins, you must issue many [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) and [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) statements.

Stop the server, install the new plugin library file in the plugin directory, and restart the server.

#### 13.7.4.5 UNINSTALL COMPONENT Statement

UNINSTALL COMPONENT ***component\_name*** [, ***component\_name*** ] ...

This statement deactivates and uninstalls one or more components. A component provides services that are available to the server and other components; see [Section 5.5, “MySQL Components”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#components). [**UNINSTALL COMPONENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-component) is the complement of [**INSTALL COMPONENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-component). It requires the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privilege for the **mysql.component** system table because it removes the row from that table that registers the component.

Example:

UNINSTALL COMPONENT 'file://component1', 'file://component2';

For information about component naming, see [Section 13.7.4.3, “INSTALL COMPONENT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-component).

If any error occurs, the statement fails and has no effect. For example, this happens if a component name is erroneous, a named component is not installed, or cannot be uninstalled because other installed components depend on it.

A loader service handles component unloading, which includes removing uninstalled components from the **mysql.component** system table that serves as a registry. As a result, unloaded components are not loaded during the startup sequence for subsequent server restarts.

**Note**

This statement has no effect for keyring components, which are loaded using a manifest file and cannot be uninstalled. See [Section 6.4.4.2, “Keyring Component Installation”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#keyring-component-installation).

#### 13.7.4.6 UNINSTALL PLUGIN Statement

UNINSTALL PLUGIN ***plugin\_name***

This statement removes an installed server plugin. [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) is the complement of [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin). It requires the [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_delete) privilege for the **mysql.plugin** system table because it removes the row from that table that registers the plugin.

***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** system table, so that subsequent server restarts do not load and initialize the plugin. [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) does not remove the plugin's shared library file.

You cannot uninstall a plugin if any table that uses it is open.

Plugin removal has implications for the use of associated tables. For example, if a full-text parser plugin is associated with a **FULLTEXT** index on the table, uninstalling the plugin makes the table unusable. Any attempt to access the table results in an error. The table cannot even be opened, so you cannot drop an index for which the plugin is used. This means that uninstalling a plugin is something to do with care unless you do not care about the table contents. If you are uninstalling a plugin with no intention of reinstalling it later and you care about the table contents, you should dump the table with [**mysqldump**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqldump) and remove the **WITH PARSER** clause from the dumped [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement so that you can reload the table later. If you do not care about the table, [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table) can be used even if any plugins associated with the table are missing.

For additional information about plugin loading, see [Section 5.6.1, “Installing and Uninstalling Plugins”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#plugin-loading).

### 13.7.5 CLONE Statement

CLONE ***clone\_action***

***clone\_action***: {

LOCAL DATA DIRECTORY [=] '***clone\_dir***';

| INSTANCE FROM '***user***'@'***host***':***port***

IDENTIFIED BY '***password***'

[DATA DIRECTORY [=] '***clone\_dir***']

[REQUIRE [NO] SSL]

}

The [**CLONE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#clone) statement is used to clone data locally or from a remote MySQL server instance. To use [**CLONE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#clone) syntax, the clone plugin must be installed. See [Section 5.6.7, “The Clone Plugin”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin).

[**CLONE LOCAL DATA DIRECTORY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#clone) syntax clones data from the local MySQL data directory to a directory on the same server or node where the MySQL server instance runs. The **'clone\_dir'** directory is the full path of the local directory that data is cloned to. An absolute path is required. The specified directory must not exist, but the specified path must be an existent path. The MySQL server requires the necessary write access to create the specified directory. For more information, see [Section 5.6.7.2, “Cloning Data Locally”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-local).

[**CLONE INSTANCE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#clone) syntax clones data from a remote MySQL server instance (the donor) and transfers it to the MySQL instance where the cloning operation was initiated (the recipient).

***user*** is the clone user on the donor MySQL server instance.

***host*** is the [**hostname**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_hostname) address of the donor MySQL server instance. Internet Protocol version 6 (IPv6) address format is not supported. An alias to the IPv6 address can be used instead. An IPv4 address can be used as is.

***port*** is the [**port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_port) number of the donor MySQL server instance. (The X Protocol port specified by [**mysqlx\_port**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\document-store.html#sysvar_mysqlx_port) is not supported. Connecting to the donor MySQL server instance through MySQL Router is also not supported.)

**IDENTIFIED BY '*password*'** specifies the password of the clone user on the donor MySQL server instance.

**DATA DIRECTORY [=] '*clone\_dir*'** is an optional clause used to specify a directory on the recipient for the data you are cloning. Use this option if you do not want to remove existing data in the recipient data directory. An absolute path is required, and the directory must not exist. The MySQL server must have the necessary write access to create the directory.

When the optional **DATA DIRECTORY [=] '*clone\_dir*'** clause is not used, a cloning operation removes existing data in the recipient data directory, replaces it with the cloned data, and automatically restarts the server afterward.

**[REQUIRE [NO] SSL]** explicitly specifies whether an encrypted connection is to be used or not when transferring cloned data over the network. An error is returned if the explicit specification cannot be satisfied. If an SSL clause is not specified, clone attempts to establish an encrypted connection by default, falling back to an insecure connection if the secure connection attempt fails. A secure connection is required when cloning encrypted data regardless of whether this clause is specified. For more information, see [Configuring an Encrypted Connection for Cloning](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-remote-ssl).

For additional information about cloning data from a remote MySQL server instance, see [Section 5.6.7.3, “Cloning Remote Data”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#clone-plugin-remote).

### 13.7.6 SET Statements

[13.7.6.1 SET Syntax for Variable Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable)

[13.7.6.2 SET CHARACTER SET Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set)

[13.7.6.3 SET NAMES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names)

The [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-statement) statement has several forms. Descriptions for those forms that are not associated with a specific server capability appear in subsections of this section:

[**SET *var\_name* = *value***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) enables you to assign values to variables that affect the operation of the server or clients. See [Section 13.7.6.1, “SET Syntax for Variable Assignment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable).

[**SET CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set) and [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) assign values to character set and collation variables associated with the current connection to the server. See [Section 13.7.6.2, “SET CHARACTER SET Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set), and [Section 13.7.6.3, “SET NAMES Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names).

Descriptions for the other forms appear elsewhere, grouped with other statements related to the capability they help implement:

[**SET DEFAULT ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role) and [**SET ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) set the default role and current role for user accounts. See [Section 13.7.1.9, “SET DEFAULT ROLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-default-role), and [Section 13.7.1.11, “SET ROLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role).

[**SET PASSWORD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password) assigns account passwords. See [Section 13.7.1.10, “SET PASSWORD Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-password).

**SET RESOURCE GROUP** assigns threads to a resource group. See [Section 13.7.2.4, “SET RESOURCE GROUP Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-resource-group).

[**SET TRANSACTION ISOLATION LEVEL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction) sets the isolation level for transaction processing. See [Section 13.3.7, “SET TRANSACTION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-transaction).

#### 13.7.6.1 SET Syntax for Variable Assignment

SET ***variable*** = ***expr*** [, ***variable*** = ***expr***] ...

***variable***: {

***user\_var\_name***

| ***param\_name***

| ***local\_var\_name***

| {GLOBAL | @@GLOBAL.} ***system\_var\_name***

| {PERSIST | @@PERSIST.} ***system\_var\_name***

| {PERSIST\_ONLY | @@PERSIST\_ONLY.} ***system\_var\_name***

| [SESSION | @@SESSION. | @@] ***system\_var\_name***

}

[**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) syntax for variable assignment enables you to assign values to different types of variables that affect the operation of the server or clients:

User-defined variables. See [Section 9.4, “User-Defined Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#user-variables).

Stored procedure and function parameters, and stored program local variables. See [Section 13.6.4, “Variables in Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stored-program-variables).

System variables. See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables). System variables also can be set at server startup, as described in [Section 5.1.9, “Using System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#using-system-variables).

A [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement that assigns variable values is not written to the binary log, so in replication scenarios it affects only the host on which you execute it. To affect all replication hosts, execute the statement on each host.

The following sections describe [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) syntax for setting variables. They use the [**=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-equal) assignment operator, but the [**:=**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_assign-value) assignment operator is also permitted for this purpose.

[User-Defined Variable Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable-user-variables)

[Parameter and Local Variable Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable-parameters-local-variables)

[System Variable Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable-system-variables)

[SET Error Handling](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable-error-handling)

[Multiple Variable Assignment](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable-multiple-assignments)

[System Variable References in Expressions](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#variable-references-in-expressions)

##### User-Defined Variable Assignment

User-defined variables are created locally within a session and exist only within the context of that session; see [Section 9.4, “User-Defined Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\language-structure.html#user-variables).

A user-defined variable is written as **@*var\_name*** and is assigned an expression value as follows:

SET @***var\_name*** = ***expr***;

Examples:

SET @name = 43;

SET @total\_tax = (SELECT SUM(tax) FROM taxable\_transactions);

As demonstrated by those statements, ***expr*** can range from simple (a literal value) to more complex (the value returned by a scalar subquery).

The Performance Schema [**user\_variables\_by\_thread**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-variable-tables) table contains information about user-defined variables. See [Section 27.12.10, “Performance Schema User-Defined Variable Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-user-variable-tables).

##### Parameter and Local Variable Assignment

[**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) applies to parameters and local variables in the context of the stored object within which they are defined. The following procedure uses the **increment** procedure parameter and **counter** local variable:

CREATE PROCEDURE p(increment INT)

BEGIN

DECLARE counter INT DEFAULT 0;

WHILE counter < 10 DO

-- ... do work ...

SET counter = counter + increment;

END WHILE;

END;

##### System Variable Assignment

The MySQL server maintains system variables that configure its operation. A system variable can have a global value that affects server operation as a whole, a session value that affects the current session, or both. Many system variables are dynamic and can be changed at runtime using the [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement to affect operation of the current server instance. [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) can also be used to persist certain system variables to the mysqld-auto.cnf file in the data directory, to affect server operation for subsequent startups.

If you change a session system variable, the value remains in effect within your session until you change the variable to a different value or the session ends. The change has no effect on other sessions.

If you change a global system variable, the value is remembered and used to initialize the session value for new sessions until you change the variable to a different value or the server exits. The change is visible to any client that accesses the global value. However, the change affects the corresponding session value only for clients that connect after the change. The global variable change does not affect the session value for any current client sessions (not even the session within which the global value change occurs).

To make a global system variable setting permanent so that it applies across server restarts, you can persist it to the mysqld-auto.cnf file in the data directory. It is also possible to make persistent configuration changes by manually modifying a my.cnf option file, but that is more cumbersome, and an error in a manually entered setting might not be discovered until much later. [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statements that persist system variables are more convenient and avoid the possibility of malformed settings because settings with syntax errors do not succeed and do not change server configuration. For more information about persisting system variables and the mysqld-auto.cnf file, see [Section 5.1.9.3, “Persisted System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#persisted-system-variables).

**Note**

Setting or persisting a global system variable value always requires special privileges. Setting a session system variable value normally requires no special privileges and can be done by any user, although there are exceptions. For more information, see [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

The following discussion describes the syntax options for setting and persisting system variables:

To assign a value to a global system variable, precede the variable name by the **GLOBAL** keyword or the **@@GLOBAL.** qualifier:

SET GLOBAL max\_connections = 1000;

SET @@GLOBAL.max\_connections = 1000;

To assign a value to a session system variable, precede the variable name by the **SESSION** or **LOCAL** keyword, by the **@@SESSION.**, **@@LOCAL.**, or **@@** qualifier, or by no keyword or no modifier at all:

SET SESSION sql\_mode = 'TRADITIONAL';

SET LOCAL sql\_mode = 'TRADITIONAL';

SET @@SESSION.sql\_mode = 'TRADITIONAL';

SET @@LOCAL.sql\_mode = 'TRADITIONAL';

SET @@sql\_mode = 'TRADITIONAL';

SET sql\_mode = 'TRADITIONAL';

A client can change its own session variables, but not those of any other client.

To persist a global system variable to the mysqld-auto.cnf option file in the data directory, precede the variable name by the **PERSIST** keyword or the **@@PERSIST.** qualifier:

SET PERSIST max\_connections = 1000;

SET @@PERSIST.max\_connections = 1000;

This [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) syntax enables you to make configuration changes at runtime that also persist across server restarts. Like [**SET GLOBAL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable), [**SET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) sets the global variable runtime value, but also writes the variable setting to the mysqld-auto.cnf file (replacing any existing variable setting if there is one).

To persist a global system variable to the mysqld-auto.cnf file without setting the global variable runtime value, precede the variable name by the **PERSIST\_ONLY** keyword or the **@@PERSIST\_ONLY.** qualifier:

SET PERSIST\_ONLY back\_log = 100;

SET @@PERSIST\_ONLY.back\_log = 100;

Like **PERSIST**, **PERSIST\_ONLY** writes the variable setting to mysqld-auto.cnf. However, unlike **PERSIST**, **PERSIST\_ONLY** does not modify the global variable runtime value. This makes **PERSIST\_ONLY** suitable for configuring read-only system variables that can be set only at server startup.

To set a global system variable value to the compiled-in MySQL default value or a session system variable to the current corresponding global value, set the variable to the value **DEFAULT**. For example, the following two statements are identical in setting the session value of [**max\_join\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_join_size) to the current global value:

SET @@SESSION.max\_join\_size = DEFAULT;

SET @@SESSION.max\_join\_size = @@GLOBAL.max\_join\_size;

Using [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) to persist a global system variable to a value of **DEFAULT** or to its literal default value assigns the variable its default value and adds a setting for the variable to mysqld-auto.cnf. To remove the variable from the file, use [**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist).

Some system variables cannot be persisted or are persist-restricted. See [Section 5.1.9.4, “Nonpersistible and Persist-Restricted System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#nonpersistible-system-variables).

A system variable implemented by a plugin can be persisted if the plugin is installed when the [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement is executed. Assignment of the persisted plugin variable takes effect for subsequent server restarts if the plugin is still installed. If the plugin is no longer installed, the plugin variable no longer exists when the server reads the mysqld-auto.cnf file. In this case, the server writes a warning to the error log and continues:

currently unknown variable '***var\_name***'

was read from the persisted config file

To display system variable names and values:

Use the [**SHOW VARIABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables) statement; see [Section 13.7.7.41, “SHOW VARIABLES Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables).

Several Performance Schema tables provide system variable information. See [Section 27.12.14, “Performance Schema System Variable Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-system-variable-tables).

The Performance Schema [**variables\_info**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-variables-info-table) table contains information showing when and by which user each system variable was most recently set. See [Section 27.12.14.2, “Performance Schema variables\_info Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-variables-info-table).

The Performance Schema [**persisted\_variables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-persisted-variables-table) table provides an SQL interface to the mysqld-auto.cnf file, enabling its contents to be inspected at runtime using [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements. See [Section 27.12.14.1, “Performance Schema persisted\_variables Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-persisted-variables-table).

##### SET Error Handling

If any variable assignment in a [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement fails, the entire statement fails and no variables are changed, nor is the mysqld-auto.cnf file changed.

[**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) produces an error under the circumstances described here. Most of the examples show [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statements that use keyword syntax (for example, **GLOBAL** or **SESSION**), but the principles are also true for statements that use the corresponding modifiers (for example, **@@GLOBAL.** or **@@SESSION.**).

Use of [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) (any variant) to set a read-only variable:

mysql> **SET GLOBAL version = 'abc';**

ERROR 1238 (HY000): Variable 'version' is a read only variable

Use of **GLOBAL**, **PERSIST**, or **PERSIST\_ONLY** to set a variable that has only a session value:

mysql> **SET GLOBAL sql\_log\_bin = ON;**

ERROR 1228 (HY000): Variable 'sql\_log\_bin' is a SESSION

variable and can't be used with SET GLOBAL

Use of **SESSION** to set a variable that has only a global value:

mysql> **SET SESSION max\_connections = 1000;**

ERROR 1229 (HY000): Variable 'max\_connections' is a

GLOBAL variable and should be set with SET GLOBAL

Omission of **GLOBAL**, **PERSIST**, or **PERSIST\_ONLY** to set a variable that has only a global value:

mysql> **SET max\_connections = 1000;**

ERROR 1229 (HY000): Variable 'max\_connections' is a

GLOBAL variable and should be set with SET GLOBAL

Use of **PERSIST** or **PERSIST\_ONLY** to set a variable that cannot be persisted:

mysql> **SET PERSIST port = 3307;**

ERROR 1238 (HY000): Variable 'port' is a read only variable

mysql> **SET PERSIST\_ONLY port = 3307;**

ERROR 1238 (HY000): Variable 'port' is a non persistent read only variable

The **@@GLOBAL.**, **@@PERSIST.**, **@@PERSIST\_ONLY.**, **@@SESSION.**, and **@@** modifiers apply only to system variables. An error occurs for attempts to apply them to user-defined variables, stored procedure or function parameters, or stored program local variables.

Not all system variables can be set to **DEFAULT**. In such cases, assigning **DEFAULT** results in an error.

An error occurs for attempts to assign **DEFAULT** to user-defined variables, stored procedure or function parameters, or stored program local variables.

##### Multiple Variable Assignment

A [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement can contain multiple variable assignments, separated by commas. This statement assigns values to a user-defined variable and a system variable:

SET @x = 1, SESSION sql\_mode = '';

If you set multiple system variables in a single statement, the most recent **GLOBAL**, **PERSIST**, **PERSIST\_ONLY**, or **SESSION** keyword in the statement is used for following assignments that have no keyword specified.

Examples of multiple-variable assignment:

SET GLOBAL sort\_buffer\_size = 1000000, SESSION sort\_buffer\_size = 1000000;

SET @@GLOBAL.sort\_buffer\_size = 1000000, @@LOCAL.sort\_buffer\_size = 1000000;

SET GLOBAL max\_connections = 1000, sort\_buffer\_size = 1000000;

The **@@GLOBAL.**, **@@PERSIST.**, **@@PERSIST\_ONLY.**, **@@SESSION.**, and **@@** modifiers apply only to the immediately following system variable, not any remaining system variables. This statement sets the [**sort\_buffer\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sort_buffer_size) global value to 50000 and the session value to 1000000:

SET @@GLOBAL.sort\_buffer\_size = 50000, sort\_buffer\_size = 1000000;

##### System Variable References in Expressions

To refer to the value of a system variable in expressions, use one of the **@@**-modifiers (except **@@PERSIST.** and **@@PERSIST\_ONLY.**, which are not permitted in expressions). For example, you can retrieve system variable values in a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement like this:

SELECT @@GLOBAL.sql\_mode, @@SESSION.sql\_mode, @@sql\_mode;

**Note**

A reference to a system variable in an expression as **@@*var\_name*** (with **@@** rather than **@@GLOBAL.** or **@@SESSION.**) returns the session value if it exists and the global value otherwise. This differs from **SET @@*var\_name* = *expr***, which always refers to the session value.

#### 13.7.6.2 SET CHARACTER SET Statement

SET {CHARACTER SET | CHARSET}

{'***charset\_name***' | DEFAULT}

This statement maps all strings sent between the server and the current client with the given mapping. **SET CHARACTER SET** sets three session system variables: [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) and [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) are set to the given character set, and [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) to the value of [**character\_set\_database**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_database). See [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection).

***charset\_name*** may be quoted or unquoted.

The default character set mapping can be restored by using the value **DEFAULT**. The default depends on the server configuration.

Some character sets cannot be used as the client character set. Attempting to use them with [**SET CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-character-set) produces an error. See [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset).

#### 13.7.6.3 SET NAMES Statement

SET NAMES {'***charset\_name***'

[COLLATE '***collation\_name***'] | DEFAULT}

This statement sets the three session system variables [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client), [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection), and [**character\_set\_results**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_results) to the given character set. Setting [**character\_set\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_connection) to **charset\_name** also sets [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) to the default collation for **charset\_name**. See [Section 10.4, “Connection Character Sets and Collations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection).

The optional **COLLATE** clause may be used to specify a collation explicitly. If given, the collation must one of the permitted collations for ***charset\_name***.

***charset\_name*** and ***collation\_name*** may be quoted or unquoted.

The default mapping can be restored by using a value of **DEFAULT**. The default depends on the server configuration.

Some character sets cannot be used as the client character set. Attempting to use them with [**SET NAMES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-names) produces an error. See [Impermissible Client Character Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\charset.html#charset-connection-impermissible-client-charset).

### 13.7.7 SHOW Statements

[13.7.7.1 SHOW BINARY LOGS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binary-logs)

[13.7.7.2 SHOW BINLOG EVENTS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events)

[13.7.7.3 SHOW CHARACTER SET Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set)

[13.7.7.4 SHOW COLLATION Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation)

[13.7.7.5 SHOW COLUMNS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns)

[13.7.7.6 SHOW CREATE DATABASE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-database)

[13.7.7.7 SHOW CREATE EVENT Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-event)

[13.7.7.8 SHOW CREATE FUNCTION Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-function)

[13.7.7.9 SHOW CREATE PROCEDURE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-procedure)

[13.7.7.10 SHOW CREATE TABLE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table)

[13.7.7.11 SHOW CREATE TRIGGER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-trigger)

[13.7.7.12 SHOW CREATE USER Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-user)

[13.7.7.13 SHOW CREATE VIEW Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-view)

[13.7.7.14 SHOW DATABASES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-databases)

[13.7.7.15 SHOW ENGINE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine)

[13.7.7.16 SHOW ENGINES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engines)

[13.7.7.17 SHOW ERRORS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors)

[13.7.7.18 SHOW EVENTS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events)

[13.7.7.19 SHOW FUNCTION CODE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-function-code)

[13.7.7.20 SHOW FUNCTION STATUS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-function-status)

[13.7.7.21 SHOW GRANTS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants)

[13.7.7.22 SHOW INDEX Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index)

[13.7.7.23 SHOW MASTER STATUS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status)

[13.7.7.24 SHOW OPEN TABLES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-open-tables)

[13.7.7.25 SHOW PLUGINS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins)

[13.7.7.26 SHOW PRIVILEGES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-privileges)

[13.7.7.27 SHOW PROCEDURE CODE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-procedure-code)

[13.7.7.28 SHOW PROCEDURE STATUS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-procedure-status)

[13.7.7.29 SHOW PROCESSLIST Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist)

[13.7.7.30 SHOW PROFILE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile)

[13.7.7.31 SHOW PROFILES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profiles)

[13.7.7.32 SHOW RELAYLOG EVENTS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events)

[13.7.7.33 SHOW REPLICAS | SHOW SLAVE HOSTS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas)

[13.7.7.34 SHOW SLAVE HOSTS | SHOW REPLICAS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-hosts)

[13.7.7.35 SHOW REPLICA | SLAVE STATUS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status)

[13.7.7.36 SHOW SLAVE | REPLICA STATUS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-status)

[13.7.7.37 SHOW STATUS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-status)

[13.7.7.38 SHOW TABLE STATUS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status)

[13.7.7.39 SHOW TABLES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-tables)

[13.7.7.40 SHOW TRIGGERS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-triggers)

[13.7.7.41 SHOW VARIABLES Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables)

[13.7.7.42 SHOW WARNINGS Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings)

**[SHOW](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "show" \o "13.7.7 SHOW Statements)** has many forms that provide information about databases, tables, columns, or status information about the server. This section describes those following:

SHOW {BINARY | MASTER} LOGS

SHOW BINLOG EVENTS [IN '***log\_name***'] [FROM ***pos***] [LIMIT [***offset***,] ***row\_count***]

SHOW CHARACTER SET [***like\_or\_where***]

SHOW COLLATION [***like\_or\_where***]

SHOW [FULL] COLUMNS FROM ***tbl\_name*** [FROM ***db\_name***] [***like\_or\_where***]

SHOW CREATE DATABASE ***db\_name***

SHOW CREATE EVENT ***event\_name***

SHOW CREATE FUNCTION ***func\_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 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 MASTER 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 RELAYLOG EVENTS [IN '***log\_name***'] [FROM ***pos***] [LIMIT [***offset***,] ***row\_count***]

SHOW {REPLICAS | SLAVE HOSTS}

SHOW {REPLICA | SLAVE} STATUS [FOR CHANNEL ***channel***]

SHOW [GLOBAL | SESSION] STATUS [***like\_or\_where***]

SHOW TABLE STATUS [FROM ***db\_name***] [***like\_or\_where***]

SHOW [FULL] 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statement includes a [**LIKE '*pattern*'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statements also accept a **WHERE** clause that provides more flexibility in specifying which rows to display. See [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

Many MySQL APIs (such as PHP) enable you to treat the result returned from a [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statement as you would a result set from a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select); see [Chapter 29, *Connectors and APIs*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\connectors-apis.html), or your API documentation for more information. In addition, you can work in SQL with results from queries on tables in the **INFORMATION\_SCHEMA** database, which you cannot easily do with results from [**SHOW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show) statements. See [Chapter 26, *INFORMATION\_SCHEMA Tables*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html).

#### 13.7.7.1 SHOW BINARY LOGS Statement

SHOW BINARY LOGS

SHOW MASTER LOGS

Lists the binary log files on the server. This statement is used as part of the procedure described in [Section 13.4.1.1, “PURGE BINARY LOGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#purge-binary-logs), that shows how to determine which logs can be purged. [**SHOW BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binary-logs) requires the [**REPLICATION CLIENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-client) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

Encrypted binary log files have a 512-byte file header that stores information required for encryption and decryption of the file. This is included in the file size displayed by [**SHOW BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binary-logs). The **Encrypted** column shows whether or not the binary log file is encrypted. Binary log encryption is active if [**binlog\_encryption=ON**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_binlog_encryption) is set for the server. Existing binary log files are not encrypted or decrypted if binary log encryption is activated or deactivated while the server is running.

mysql> **SHOW BINARY LOGS;**

+---------------+-----------+-----------+

| Log\_name | File\_size | Encrypted |

+---------------+-----------+-----------+

| binlog.000015 | 724935 | Yes |

| binlog.000016 | 733481 | Yes |

+---------------+-----------+-----------+

[**SHOW MASTER LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binary-logs) is equivalent to [**SHOW BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binary-logs).

#### 13.7.7.2 SHOW BINLOG EVENTS Statement

SHOW BINLOG EVENTS

[IN '***log\_name***']

[FROM ***pos***]

[LIMIT [***offset***,] ***row\_count***]

Shows the events in the binary log. If you do not specify **'*log\_name*'**, the first binary log is displayed. **SHOW BINLOG EVENTS** requires the [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) privilege.

The **LIMIT** clause has the same syntax as for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. See [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

**Note**

Issuing a [**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events) with no **LIMIT** clause could start a very time- and resource-consuming process because the server returns to the client the complete contents of the binary log (which includes all statements executed by the server that modify data). As an alternative to [**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events), use the [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) utility to save the binary log to a text file for later examination and analysis. See [Section 4.6.9, “mysqlbinlog — Utility for Processing Binary Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog).

[**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events) displays the following fields for each event in the binary log:

**Log\_name**

The name of the file that is being listed.

**Pos**

The position at which the event occurs.

**Event\_type**

An identifier that describes the event type.

**Server\_id**

The server ID of the server on which the event originated.

**End\_log\_pos**

The position at which the next event begins, which is equal to **Pos** plus the size of the event.

**Info**

More detailed information about the event type. The format of this information depends on the event type.

For compressed transaction payloads, the **Transaction\_payload\_event** is first printed as a single unit, then it is unpacked and each event inside it is printed.

Some events relating to the setting of user and system variables are not included in the output from [**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events). To get complete coverage of events within a binary log, use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog).

[**SHOW BINLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-binlog-events) does not work with relay log files. You can use [**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events) for this purpose.

#### 13.7.7.3 SHOW CHARACTER SET Statement

SHOW CHARACTER SET

[LIKE '***pattern***' | WHERE ***expr***]

The [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) statement shows all available character sets. The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which character set names to match. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show). For example:

mysql> **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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) output has these columns:

**Charset**

The character set name.

**Description**

A description of the character set.

**Default collation**

The default collation for the character set.

**Maxlen**

The maximum number of bytes required to store one character.

The **filename** character set is for internal use only; consequently, [**SHOW CHARACTER SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-character-set) does not display it.

Character set information is also available from the **INFORMATION\_SCHEMA** [**CHARACTER\_SETS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table) table. See [Section 26.3.4, “The INFORMATION\_SCHEMA CHARACTER\_SETS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-character-sets-table).

#### 13.7.7.4 SHOW COLLATION Statement

SHOW COLLATION

[LIKE '***pattern***' | WHERE ***expr***]

This statement lists collations supported by the server. By default, the output from [**SHOW COLLATION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) includes all available collations. The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which collation names to match. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show). For example:

mysql> **SHOW COLLATION WHERE Charset = '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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-collation) output has these columns:

**Collation**

The collation name.

**Charset**

The name of the character set with which the collation is associated.

**Id**

The collation ID.

**Default**

Whether the collation is the default for its character set.

**Compiled**

Whether the character set is compiled into the server.

**Sortlen**

This is related to the amount of memory required to sort strings expressed in the character set.

To see the default collation for each character set, use the following statement. **Default** is a reserved word, so to use it as an identifier, it must be quoted as such:

mysql> **SHOW COLLATION WHERE `Default` = 'Yes';**

+---------------------+----------+----+---------+----------+---------+

| Collation | Charset | Id | Default | Compiled | Sortlen |

+---------------------+----------+----+---------+----------+---------+

| big5\_chinese\_ci | big5 | 1 | Yes | Yes | 1 |

| dec8\_swedish\_ci | dec8 | 3 | Yes | Yes | 1 |

| cp850\_general\_ci | cp850 | 4 | Yes | Yes | 1 |

| hp8\_english\_ci | hp8 | 6 | Yes | Yes | 1 |

| koi8r\_general\_ci | koi8r | 7 | Yes | Yes | 1 |

| latin1\_swedish\_ci | latin1 | 8 | Yes | Yes | 1 |

...

Collation information is also available from the **INFORMATION\_SCHEMA** [**COLLATIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table) table. See [Section 26.3.6, “The INFORMATION\_SCHEMA COLLATIONS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-collations-table).

#### 13.7.7.5 SHOW COLUMNS Statement

SHOW [EXTENDED] [FULL] {COLUMNS | FIELDS}

{FROM | IN} ***tbl\_name***

[{FROM | IN} ***db\_name***]

[LIKE '***pattern***' | WHERE ***expr***]

[**SHOW COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns) displays information about the columns in a given table. It also works for views. [**SHOW COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns) displays information only for those columns for which you have some privilege.

mysql> **SHOW COLUMNS FROM City;**

+-------------+----------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------------+----------+------+-----+---------+----------------+

| ID | int(11) | NO | PRI | NULL | auto\_increment |

| Name | char(35) | NO | | | |

| CountryCode | char(3) | NO | MUL | | |

| District | char(20) | NO | | | |

| Population | int(11) | NO | | 0 | |

+-------------+----------+------+-----+---------+----------------+

An alternative to ***tbl\_name* FROM *db\_name*** syntax is ***db\_name.tbl\_name***. These two statements are equivalent:

SHOW COLUMNS FROM mytable FROM mydb;

SHOW COLUMNS FROM mydb.mytable;

The optional **EXTENDED** keyword causes the output to include information about hidden columns that MySQL uses internally and are not accessible by users.

The optional **FULL** keyword causes the output to include the column collation and comments, as well as the privileges you have for each column.

The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which column names to match. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

The data types may differ from what you expect them to be based on a [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement because MySQL sometimes changes data types when you create or alter a table. The conditions under which this occurs are described in [Section 13.1.20.7, “Silent Column Specification Changes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#silent-column-changes).

[**SHOW COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns) displays the following values for each table column:

**Field**

The name of the column.

**Type**

The column data type.

**Collation**

The collation for nonbinary string columns, or **NULL** for other columns. This value is displayed only if you use the **FULL** keyword.

**Null**

The column nullability. The value is **YES** if **NULL** values can be stored in the column, **NO** if not.

**Key**

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, nonunique 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** index. (A **UNIQUE** index permits multiple **NULL** values, but you can tell whether the column permits **NULL** by checking the **Null** field.)

If **Key** is **MUL**, the column is the first column of a nonunique index in which multiple occurrences of a given value are permitted within the column.

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.

**Default**

The default value for the column. This is **NULL** if the column has an explicit default of **NULL**, or if the column definition includes no **DEFAULT** clause.

**Extra**

Any additional information that is available about a given column. The value is nonempty in these cases:

**auto\_increment** for columns that have the **AUTO\_INCREMENT** attribute.

**on update CURRENT\_TIMESTAMP** for [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) or [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) columns that have the **ON UPDATE CURRENT\_TIMESTAMP** attribute.

**VIRTUAL GENERATED** or **VIRTUAL STORED** for generated columns.

**DEFAULT\_GENERATED** for columns that have an expression default value.

**Privileges**

The privileges you have for the column. This value is displayed only if you use the **FULL** keyword.

**Comment**

Any comment included in the column definition. This value is displayed only if you use the **FULL** keyword.

Table column information is also available from the **INFORMATION\_SCHEMA** [**COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-columns-table) table. See [Section 26.3.8, “The INFORMATION\_SCHEMA COLUMNS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-columns-table). The extended information about hidden columns is available only using **SHOW EXTENDED COLUMNS**; it cannot be obtained from the [**COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-columns-table) table.

You can list a table's columns with the [**mysqlshow *db\_name* *tbl\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlshow) command.

The [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) statement provides information similar to [**SHOW COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns). See [Section 13.8.1, “DESCRIBE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe).

The [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table), [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status), and [**SHOW INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index) statements also provide information about tables. See [Section 13.7.7, “SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show).

#### 13.7.7.6 SHOW CREATE DATABASE Statement

SHOW CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] ***db\_name***

Shows the [**CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-database) statement that creates the named database. If the **SHOW** statement includes an **IF NOT EXISTS** clause, the output too includes such a clause. [**SHOW CREATE SCHEMA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-database) is a synonym for [**SHOW CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-database).

mysql> **SHOW CREATE DATABASE test\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Database: test

Create Database: CREATE DATABASE `test` /\*!40100 DEFAULT CHARACTER SET utf8mb4

COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80014 DEFAULT ENCRYPTION='N' \*/

mysql> **SHOW CREATE SCHEMA test\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Database: test

Create Database: CREATE DATABASE `test` /\*!40100 DEFAULT CHARACTER SET utf8mb4

COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80014 DEFAULT ENCRYPTION='N' \*/

[**SHOW CREATE DATABASE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-database) quotes table and column names according to the value of the [**sql\_quote\_show\_create**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_quote_show_create) option. See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

#### 13.7.7.7 SHOW CREATE EVENT Statement

SHOW CREATE EVENT ***event\_name***

This statement displays the [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statement needed to re-create a given event. It requires the [**EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_event) privilege for the database from which the event is to be shown. For example (using the same event **e\_daily** defined and then altered in [Section 13.7.7.18, “SHOW EVENTS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events)):

mysql> **SHOW CREATE EVENT myschema.e\_daily\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Event: e\_daily

sql\_mode: ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,

NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,

ERROR\_FOR\_DIVISION\_BY\_ZERO,

NO\_ENGINE\_SUBSTITUTION

time\_zone: SYSTEM

Create Event: CREATE DEFINER=`jon`@`ghidora` 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: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

Database Collation: utf8mb4\_0900\_ai\_ci

**character\_set\_client** is the session value of the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable when the event was created. **collation\_connection** is the session value of the [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variable when the event was created. **Database Collation** is the collation of the database with which the event is associated.

The output reflects the current status of the event (**ENABLE**) rather than the status with which it was created.

#### 13.7.7.8 SHOW CREATE FUNCTION Statement

SHOW CREATE FUNCTION ***func\_name***

This statement is similar to [**SHOW CREATE PROCEDURE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-procedure) but for stored functions. See [Section 13.7.7.9, “SHOW CREATE PROCEDURE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-procedure).

#### 13.7.7.9 SHOW CREATE PROCEDURE Statement

SHOW CREATE PROCEDURE ***proc\_name***

This statement is a MySQL extension. It returns the exact string that can be used to re-create the named stored procedure. A similar statement, [**SHOW CREATE FUNCTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-function), displays information about stored functions (see [Section 13.7.7.8, “SHOW CREATE FUNCTION Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-function)).

To use either statement, you must be the user named as the routine **DEFINER**, have the [**SHOW\_ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-routine) privilege, have the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege at the global level, or have the [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine), [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine), or [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute) privilege granted at a scope that includes the routine. The value displayed for the **Create Procedure** or **Create Function** field is **NULL** if you have only [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine), [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine), or [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute).

mysql> **SHOW CREATE PROCEDURE test.citycount\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Procedure: citycount

sql\_mode: ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,

NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,

ERROR\_FOR\_DIVISION\_BY\_ZERO,

NO\_ENGINE\_SUBSTITUTION

Create Procedure: CREATE DEFINER=`me`@`localhost`

PROCEDURE `citycount`(IN country CHAR(3), OUT cities INT)

BEGIN

SELECT COUNT(\*) INTO cities FROM world.city

WHERE CountryCode = country;

END

character\_set\_client: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

Database Collation: utf8mb4\_0900\_ai\_ci

mysql> **SHOW CREATE FUNCTION test.hello\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function: hello

sql\_mode: ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,

NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,

ERROR\_FOR\_DIVISION\_BY\_ZERO,

NO\_ENGINE\_SUBSTITUTION

Create Function: CREATE DEFINER=`me`@`localhost`

FUNCTION `hello`(s CHAR(20))

RETURNS char(50) CHARSET utf8mb4

DETERMINISTIC

RETURN CONCAT('Hello, ',s,'!')

character\_set\_client: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

Database Collation: utf8mb4\_0900\_ai\_ci

**character\_set\_client** is the session value of the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable when the routine was created. **collation\_connection** is the session value of the [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variable when the routine was created. **Database Collation** is the collation of the database with which the routine is associated.

#### 13.7.7.10 SHOW CREATE TABLE Statement

SHOW CREATE TABLE ***tbl\_name***

Shows the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement that creates the named table. To use this statement, you must have some privilege for the table. This statement also works with views.

mysql> **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=utf8mb4

As of MySQL 8.0.16, MySQL implements **CHECK** constraints and [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) displays them. All **CHECK** constraints are displayed as table constraints. That is, a **CHECK** constraint originally specified as part of a column definition displays as a separate clause not part of the column definition. Example:

mysql> **CREATE TABLE t1 (**

**i1 INT CHECK (i1 <> 0), -- column constraint**

**i2 INT,**

**CHECK (i2 > i1), -- table constraint**

**CHECK (i2 <> 0) NOT ENFORCED -- table constraint, not enforced**

**);**

mysql> **SHOW CREATE TABLE t1\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t1

Create Table: CREATE TABLE `t1` (

`i1` int(11) DEFAULT NULL,

`i2` int(11) DEFAULT NULL,

CONSTRAINT `t1\_chk\_1` CHECK ((`i1` <> 0)),

CONSTRAINT `t1\_chk\_2` CHECK ((`i2` > `i1`)),

CONSTRAINT `t1\_chk\_3` CHECK ((`i2` <> 0)) /\*!80016 NOT ENFORCED \*/

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

[**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) quotes table and column names according to the value of the [**sql\_quote\_show\_create**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_quote_show_create) option. See [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

When altering the storage engine of a table, table options that are not applicable to the new storage engine are retained in the table definition to enable reverting the table with its previously defined options to the original storage engine, if necessary. For example, when changing the storage engine from InnoDB to MyISAM, InnoDB-specific options such as **ROW\_FORMAT=COMPACT** are retained.

mysql> **CREATE TABLE t1 (c1 INT PRIMARY KEY) ROW\_FORMAT=COMPACT ENGINE=InnoDB;**

mysql> **ALTER TABLE t1 ENGINE=MyISAM;**

mysql> **SHOW CREATE TABLE t1\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t1

Create Table: CREATE TABLE `t1` (

`c1` int NOT NULL,

PRIMARY KEY (`c1`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci ROW\_FORMAT=COMPACT

When creating a table with [strict mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_strict_mode) disabled, the storage engine's default row format is used if the specified row format is not supported. The actual row format of the table is reported in the **Row\_format** column in response to [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status). [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table) shows the row format that was specified in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement.

#### 13.7.7.11 SHOW CREATE TRIGGER Statement

SHOW CREATE TRIGGER ***trigger\_name***

This statement shows the [**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger) statement that creates the named trigger. This statement requires the [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_trigger) privilege for the table associated with the trigger.

mysql> **SHOW CREATE TRIGGER ins\_sum\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Trigger: ins\_sum

sql\_mode: ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,

NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,

ERROR\_FOR\_DIVISION\_BY\_ZERO,

NO\_ENGINE\_SUBSTITUTION

SQL Original Statement: CREATE DEFINER=`me`@`localhost` TRIGGER `ins\_sum`

BEFORE INSERT ON `account`

FOR EACH ROW SET @sum = @sum + NEW.amount

character\_set\_client: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

Database Collation: utf8mb4\_0900\_ai\_ci

Created: 2018-08-08 10:10:12.61

[**SHOW CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-trigger) output has these columns:

**Trigger**: The trigger name.

**sql\_mode**: The SQL mode in effect when the trigger executes.

**SQL Original Statement**: The [**CREATE TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-trigger) statement that defines the trigger.

**character\_set\_client**: The session value of the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable when the trigger was created.

**collation\_connection**: The session value of the [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variable when the trigger was created.

**Database Collation**: The collation of the database with which the trigger is associated.

**Created**: The date and time when the trigger was created. This is a **TIMESTAMP(2)** value (with a fractional part in hundredths of seconds) for triggers.

Trigger information is also available from the **INFORMATION\_SCHEMA** [**TRIGGERS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-triggers-table) table. See [Section 26.3.45, “The INFORMATION\_SCHEMA TRIGGERS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-triggers-table).

#### 13.7.7.12 SHOW CREATE USER Statement

SHOW CREATE USER ***user***

This statement shows the [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user) statement that creates the named user. An error occurs if the user does not exist. The statement requires the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for the **mysql** system schema, except to see information for the current user. For the current user, the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for the **mysql.user** system table is required for display of the password hash in the **IDENTIFIED AS** clause; otherwise, the hash displays as **<secret>**.

To name the account, use the format described in [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names). The host name part of the account name, if omitted, defaults to **'%'**. It is also possible to specify [**CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) or [**CURRENT\_USER()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_current-user) to refer to the account associated with the current session.

Password hash values displayed in the **IDENTIFIED WITH** clause of output from [**SHOW CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-user) may contain unprintable characters that have adverse effects on terminal displays and in other environments. Enabling the [**print\_identified\_with\_as\_hex**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_print_identified_with_as_hex) system variable (available as of MySQL 8.0.17) causes [**SHOW CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-user) to display such hash values as hexadecimal strings rather than as regular string literals. Hash values that do not contain unprintable characters still display as regular string literals, even with this variable enabled.

mysql> **CREATE USER 'u1'@'localhost' IDENTIFIED BY 'secret';**

mysql> **SET print\_identified\_with\_as\_hex = ON;**

mysql> **SHOW CREATE USER 'u1'@'localhost'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CREATE USER for u1@localhost: CREATE USER 'u1'@'localhost'

IDENTIFIED WITH 'caching\_sha2\_password'

AS 0x244124303035240C7745603626313D613C4C10633E0A104B1E14135A544A7871567245614F4872344643546336546F624F6C7861326932752F45622F4F473273597557627139

REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK

PASSWORD HISTORY DEFAULT PASSWORD REUSE INTERVAL DEFAULT

PASSWORD REQUIRE CURRENT DEFAULT

To display the privileges granted to an account, use the [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) statement. See [Section 13.7.7.21, “SHOW GRANTS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants).

#### 13.7.7.13 SHOW CREATE VIEW Statement

SHOW CREATE VIEW ***view\_name***

This statement shows the [**CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-view) statement that creates the named view.

mysql> **SHOW CREATE VIEW v\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

View: v

Create View: CREATE ALGORITHM=UNDEFINED

DEFINER=`bob`@`localhost`

SQL SECURITY DEFINER VIEW

`v` AS select 1 AS `a`,2 AS `b`

character\_set\_client: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

**character\_set\_client** is the session value of the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable when the view was created. **collation\_connection** is the session value of the [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variable when the view was created.

Use of [**SHOW CREATE VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-view) requires the [**SHOW VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-view) privilege, and the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for the view in question.

View information is also available from the **INFORMATION\_SCHEMA** [**VIEWS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-views-table) table. See [Section 26.3.48, “The INFORMATION\_SCHEMA VIEWS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-views-table).

MySQL lets you use different [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) settings to tell the server the type of SQL syntax to support. For example, you might use the [**ANSI**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_ansi) SQL mode to ensure MySQL correctly interprets the standard SQL concatenation operator, the double bar (**||**), in your queries. If you then create a view that concatenates items, you might worry that changing the [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) setting to a value different from [**ANSI**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sqlmode_ansi) could cause the view to become invalid. But this is not the case. No matter how you write out a view definition, MySQL always stores it the same way, in a canonical form. Here is an example that shows how the server changes a double bar concatenation operator to a [**CONCAT()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_concat) function:

mysql> **SET sql\_mode = 'ANSI';**

Query OK, 0 rows affected (0.00 sec)

mysql> **CREATE VIEW test.v AS SELECT 'a' || 'b' as col1;**

Query OK, 0 rows affected (0.01 sec)

mysql> **SHOW CREATE VIEW test.v\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

View: v

Create View: CREATE VIEW "v" AS select concat('a','b') AS "col1"

...

1 row in set (0.00 sec)

The advantage of storing a view definition in canonical form is that changes made later to the value of [**sql\_mode**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_mode) do not affect the results from the view. However an additional consequence is that comments prior to [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) are stripped from the definition by the server.

#### 13.7.7.14 SHOW DATABASES Statement

SHOW {DATABASES | SCHEMAS}

[LIKE '***pattern***' | WHERE ***expr***]

[**SHOW DATABASES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-databases) lists the databases on the MySQL server host. [**SHOW SCHEMAS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-databases) is a synonym for [**SHOW DATABASES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-databases). The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which database names to match. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

You see only those databases for which you have some kind of privilege, unless you have the global [**SHOW DATABASES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-databases) privilege. You can also get this list using the [**mysqlshow**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlshow) command.

If the server was started with the [--skip-show-database](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-show-database) option, you cannot use this statement at all unless you have the [**SHOW DATABASES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-databases) privilege.

MySQL implements databases as directories in the data directory, so this statement simply lists directories in that location. However, the output may include names of directories that do not correspond to actual databases.

Database information is also available from the **INFORMATION\_SCHEMA** [**SCHEMATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-schemata-table) table. See [Section 26.3.31, “The INFORMATION\_SCHEMA SCHEMATA Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-schemata-table).

**Caution**

Because any static global privilege is considered a privilege for all databases, any static global privilege enables a user to see all database names with [**SHOW DATABASES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-databases) or by examining the [**SCHEMATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-schemata-table) table of **INFORMATION\_SCHEMA**, except databases that have been restricted at the database level by partial revokes.

#### 13.7.7.15 SHOW ENGINE Statement

SHOW ENGINE ***engine\_name*** {STATUS | MUTEX}

[**SHOW ENGINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) displays operational information about a storage engine. It requires the [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process) privilege. The statement has these variants:

SHOW ENGINE INNODB STATUS

SHOW ENGINE INNODB MUTEX

SHOW ENGINE PERFORMANCE\_SCHEMA STATUS

[**SHOW ENGINE INNODB STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) displays extensive information from the standard **InnoDB** Monitor about the state of the **InnoDB** storage engine. For information about the standard monitor and other **InnoDB** Monitors that provide information about **InnoDB** processing, see [Section 15.17, “InnoDB Monitors”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-monitors).

[**SHOW ENGINE INNODB MUTEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) displays **InnoDB** [mutex](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_mutex) and [rw-lock](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_rw_lock) statistics.

**Note**

**InnoDB** mutexes and rwlocks can also be monitored using [Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html) tables. See [Section 15.16.2, “Monitoring InnoDB Mutex Waits Using Performance Schema”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#monitor-innodb-mutex-waits-performance-schema).

Mutex statistics collection is configured dynamically using the following options:

To enable the collection of mutex statistics, run:

SET GLOBAL innodb\_monitor\_enable='latch';

To reset mutex statistics, run:

SET GLOBAL innodb\_monitor\_reset='latch';

To disable the collection of mutex statistics, run:

SET GLOBAL innodb\_monitor\_disable='latch';

Collection of mutex statistics for [**SHOW ENGINE INNODB MUTEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) can also be enabled by setting [**innodb\_monitor\_enable='all'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_enable), or disabled by setting [**innodb\_monitor\_disable='all'**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_monitor_disable).

[**SHOW ENGINE INNODB MUTEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) output has these columns:

**Type**

Always **InnoDB**.

**Name**

For mutexes, the **Name** field reports only the mutex name. For rwlocks, the **Name** field reports the source file where the rwlock is implemented, and the line number in the file where the rwlock is created. The line number is specific to your version of MySQL.

**Status**

The mutex status. This field reports the number of spins, waits, and calls. Statistics for low-level operating system mutexes, which are implemented outside of **InnoDB**, are not reported.

**spins** indicates the number of spins.

**waits** indicates the number of mutex waits.

**calls** indicates how many times the mutex was requested.

**SHOW ENGINE INNODB MUTEX** does not list mutexes and rw-locks for each buffer pool block, as the amount of output would be overwhelming on systems with a large buffer pool. **SHOW ENGINE INNODB MUTEX** does, however, print aggregate **BUF\_BLOCK\_MUTEX** spin, wait, and call values for buffer pool block mutexes and rw-locks. **SHOW ENGINE INNODB MUTEX** also does not list any mutexes or rw-locks that have never been waited on (**os\_waits=0**). Thus, **SHOW ENGINE INNODB MUTEX** only displays information about mutexes and rw-locks outside of the buffer pool that have caused at least one OS-level [wait](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_wait).

Use [**SHOW ENGINE PERFORMANCE\_SCHEMA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engine) to inspect the internal operation of the Performance Schema code:

mysql> **SHOW ENGINE PERFORMANCE\_SCHEMA STATUS\G**

...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Type: performance\_schema

Name: events\_waits\_history.size

Status: 76

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Type: performance\_schema

Name: events\_waits\_history.count

Status: 10000

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Type: performance\_schema

Name: events\_waits\_history.memory

Status: 760000

...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 57. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Type: performance\_schema

Name: performance\_schema.memory

Status: 26459600

...

This statement is intended to help the DBA understand the effects that different Performance Schema options have on memory requirements.

**Name** values consist of two parts, which name an internal buffer and a buffer attribute, respectively. Interpret buffer names as follows:

An internal buffer that is not exposed as a table is named within parentheses. Examples: **(pfs\_cond\_class).size**, **(pfs\_mutex\_class).memory**.

An internal buffer that is exposed as a table in the **performance\_schema** database is named after the table, without parentheses. Examples: **events\_waits\_history.size**, **mutex\_instances.count**.

A value that applies to the Performance Schema as a whole begins with **performance\_schema**. Example: **performance\_schema.memory**.

Buffer attributes have these meanings:

**size** is the size of the internal record used by the implementation, such as the size of a row in a table. **size** values cannot be changed.

**count** is the number of internal records, such as the number of rows in a table. **count** values can be changed using Performance Schema configuration options.

For a table, ***tbl\_name*.memory** is the product of **size** and **count**. For the Performance Schema as a whole, **performance\_schema.memory** is the sum of all the memory used (the sum of all other **memory** values).

In some cases, there is a direct relationship between a Performance Schema configuration parameter and a **SHOW ENGINE** value. For example, **events\_waits\_history\_long.count** corresponds to [**performance\_schema\_events\_waits\_history\_long\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#sysvar_performance_schema_events_waits_history_long_size). In other cases, the relationship is more complex. For example, **events\_waits\_history.count** corresponds to [**performance\_schema\_events\_waits\_history\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#sysvar_performance_schema_events_waits_history_size) (the number of rows per thread) multiplied by [**performance\_schema\_max\_thread\_instances**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#sysvar_performance_schema_max_thread_instances) ( the number of threads).

**SHOW ENGINE NDB STATUS.** If the server has the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine enabled, **SHOW ENGINE NDB STATUS** displays cluster status information such as the number of connected data nodes, the cluster connectstring, and cluster binary log epochs, as well as counts of various Cluster API objects created by the MySQL Server when connected to the cluster. Sample output from this statement is shown here:

mysql> **SHOW ENGINE NDB STATUS;**

+------------+-----------------------+--------------------------------------------------+

| Type | Name | Status |

+------------+-----------------------+--------------------------------------------------+

| ndbcluster | connection | cluster\_node\_id=7,

connected\_host=198.51.100.103, connected\_port=1186, number\_of\_data\_nodes=4,

number\_of\_ready\_data\_nodes=3, connect\_count=0 |

| ndbcluster | NdbTransaction | created=6, free=0, sizeof=212 |

| ndbcluster | NdbOperation | created=8, free=8, sizeof=660 |

| ndbcluster | NdbIndexScanOperation | created=1, free=1, sizeof=744 |

| ndbcluster | NdbIndexOperation | created=0, free=0, sizeof=664 |

| ndbcluster | NdbRecAttr | created=1285, free=1285, sizeof=60 |

| ndbcluster | NdbApiSignal | created=16, free=16, sizeof=136 |

| ndbcluster | NdbLabel | created=0, free=0, sizeof=196 |

| ndbcluster | NdbBranch | created=0, free=0, sizeof=24 |

| ndbcluster | NdbSubroutine | created=0, free=0, sizeof=68 |

| ndbcluster | NdbCall | created=0, free=0, sizeof=16 |

| ndbcluster | NdbBlob | created=1, free=1, sizeof=264 |

| ndbcluster | NdbReceiver | created=4, free=0, sizeof=68 |

| ndbcluster | binlog | latest\_epoch=155467, latest\_trans\_epoch=148126,

latest\_received\_binlog\_epoch=0, latest\_handled\_binlog\_epoch=0,

latest\_applied\_binlog\_epoch=0 |

+------------+-----------------------+--------------------------------------------------+

The **Status** column in each of these rows provides information about the MySQL server's connection to the cluster and about the cluster binary log's status, respectively. The **Status** information is in the form of comma-delimited set of name/value pairs.

The **connection** row's **Status** column contains the name/value pairs described in the following table.

| **Name** | **Value** |
| --- | --- |
| **cluster\_node\_id** | The node ID of the MySQL server in the cluster |
| **connected\_host** | The host name or IP address of the cluster management server to which the MySQL server is connected |
| **connected\_port** | The port used by the MySQL server to connect to the management server (**connected\_host**) |
| **number\_of\_data\_nodes** | The number of data nodes configured for the cluster (that is, the number of **[ndbd]** sections in the cluster config.ini file) |
| **number\_of\_ready\_data\_nodes** | The number of data nodes in the cluster that are actually running |
| **connect\_count** | The number of times this [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) has connected or reconnected to cluster data nodes |

The **binlog** row's **Status** column contains information relating to NDB Cluster Replication. The name/value pairs it contains are described in the following table.

| **Name** | **Value** |
| --- | --- |
| **latest\_epoch** | The most recent epoch most recently run on this MySQL server (that is, the sequence number of the most recent transaction run on the server) |
| **latest\_trans\_epoch** | The most recent epoch processed by the cluster's data nodes |
| **latest\_received\_binlog\_epoch** | The most recent epoch received by the binary log thread |
| **latest\_handled\_binlog\_epoch** | The most recent epoch processed by the binary log thread (for writing to the binary log) |
| **latest\_applied\_binlog\_epoch** | The most recent epoch actually written to the binary log |

See [Section 23.6, “NDB Cluster Replication”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#mysql-cluster-replication), for more information.

The remaining rows from the output of **SHOW ENGINE NDB STATUS** which are most likely to prove useful in monitoring the cluster are listed here by **Name**:

**NdbTransaction**: The number and size of **NdbTransaction** objects that have been created. An **NdbTransaction** is created each time a table schema operation (such as [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) or [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table)) is performed on an [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) table.

**NdbOperation**: The number and size of **NdbOperation** objects that have been created.

**NdbIndexScanOperation**: The number and size of **NdbIndexScanOperation** objects that have been created.

**NdbIndexOperation**: The number and size of **NdbIndexOperation** objects that have been created.

**NdbRecAttr**: The number and size of **NdbRecAttr** objects that have been created. In general, one of these is created each time a data manipulation statement is performed by an SQL node.

**NdbBlob**: The number and size of **NdbBlob** objects that have been created. An **NdbBlob** is created for each new operation involving a [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) column in an [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) table.

**NdbReceiver**: The number and size of any **NdbReceiver** object that have been created. The number in the **created** column is the same as the number of data nodes in the cluster to which the MySQL server has connected.

**Note**

**SHOW ENGINE NDB STATUS** returns an empty result if no operations involving [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables have been performed during the current session by the MySQL client accessing the SQL node on which this statement is run.

#### 13.7.7.16 SHOW ENGINES Statement

SHOW [STORAGE] ENGINES

[**SHOW ENGINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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.

For information about MySQL storage engines, see [Chapter 15, *The InnoDB Storage Engine*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), and [Chapter 16, *Alternative Storage Engines*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html).

mysql> **SHOW ENGINES\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: ARCHIVE

Support: YES

Comment: Archive storage engine

Transactions: NO

XA: NO

Savepoints: NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: BLACKHOLE

Support: YES

Comment: /dev/null storage engine (anything you write to it disappears)

Transactions: NO

XA: NO

Savepoints: NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: MRG\_MYISAM

Support: YES

Comment: Collection of identical MyISAM tables

Transactions: NO

XA: NO

Savepoints: NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: FEDERATED

Support: NO

Comment: Federated MySQL storage engine

Transactions: NULL

XA: NULL

Savepoints: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: MyISAM

Support: YES

Comment: MyISAM storage engine

Transactions: NO

XA: NO

Savepoints: NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 6. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: PERFORMANCE\_SCHEMA

Support: YES

Comment: Performance Schema

Transactions: NO

XA: NO

Savepoints: NO

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 7. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Engine: InnoDB

Support: DEFAULT

Comment: Supports transactions, row-level locking, and foreign keys

Transactions: YES

XA: YES

Savepoints: YES

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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: CSV

Support: YES

Comment: CSV storage engine

Transactions: NO

XA: NO

Savepoints: NO

The output from [**SHOW ENGINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engines) may vary according to the MySQL version used and other factors.

[**SHOW ENGINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-engines) output has these columns:

**Engine**

The name of the storage engine.

**Support**

The server's level of support for the storage engine, as shown in the following table.

| **Value** | **Meaning** |
| --- | --- |
| **YES** | The engine is supported and is active |
| **DEFAULT** | Like **YES**, plus this is the default engine |
| **NO** | The engine is not supported |
| **DISABLED** | The engine is supported but has been disabled |

A value of **NO** means that the server was compiled without support for the engine, so it cannot be enabled at runtime.

A value of **DISABLED** occurs either because the server was started with an option that disables the engine, or because not all options required to enable it were given. In the latter case, the error log should contain a reason indicating why the option is disabled. See [Section 5.4.2, “The Error Log”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#error-log).

You might also see **DISABLED** for a storage engine if the server was compiled to support it, but was started with a --skip-***engine\_name*** option. For the [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) storage engine, **DISABLED** means the server was compiled with support for NDB Cluster, but was not started with the [--ndbcluster](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html#option_mysqld_ndbcluster) option.

All MySQL servers support **MyISAM** tables. It is not possible to disable **MyISAM**.

**Comment**

A brief description of the storage engine.

**Transactions**

Whether the storage engine supports transactions.

**XA**

Whether the storage engine supports XA transactions.

**Savepoints**

Whether the storage engine supports savepoints.

Storage engine information is also available from the **INFORMATION\_SCHEMA** [**ENGINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-engines-table) table. See [Section 26.3.13, “The INFORMATION\_SCHEMA ENGINES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-engines-table).

#### 13.7.7.17 SHOW ERRORS Statement

SHOW ERRORS [LIMIT [***offset***,] ***row\_count***]

SHOW COUNT(\*) ERRORS

[**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors) is a diagnostic statement that is similar to [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings), except that it displays information only for errors, rather than for errors, warnings, and notes.

The **LIMIT** clause has the same syntax as for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. See [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

The [**SHOW COUNT(\*) ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors) statement displays the number of errors. You can also retrieve this number from the [**error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_error_count) variable:

SHOW COUNT(\*) ERRORS;

SELECT @@error\_count;

[**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors) and [**error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_error_count) apply only to errors, not warnings or notes. In other respects, they are similar to [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) and [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count). In particular, [**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors) cannot display information for more than [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) messages, and [**error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_error_count) can exceed the value of [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) if the number of errors exceeds [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count).

For more information, see [Section 13.7.7.42, “SHOW WARNINGS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings).

#### 13.7.7.18 SHOW EVENTS Statement

SHOW EVENTS

[{FROM | IN} ***schema\_name***]

[LIKE '***pattern***' | WHERE ***expr***]

This statement displays information about Event Manager events, which are discussed in [Section 25.4, “Using the Event Scheduler”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#event-scheduler). It requires the [**EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_event) privilege for the database from which the events are to be shown.

In its simplest form, [**SHOW EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events) lists all of the events in the current schema:

mysql> **SELECT CURRENT\_USER(), SCHEMA();**

+----------------+----------+

| CURRENT\_USER() | SCHEMA() |

+----------------+----------+

| jon@ghidora | myschema |

+----------------+----------+

1 row in set (0.00 sec)

mysql> **SHOW EVENTS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Db: myschema

Name: e\_daily

Definer: jon@ghidora

Time zone: SYSTEM

Type: RECURRING

Execute at: NULL

Interval value: 1

Interval field: DAY

Starts: 2018-08-08 11:06:34

Ends: NULL

Status: ENABLED

Originator: 1

character\_set\_client: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

Database Collation: utf8mb4\_0900\_ai\_ci

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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_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 [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

[**SHOW EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events) output has these columns:

**Db**

The name of the schema (database) to which the event belongs.

**Name**

The name of the event.

**Definer**

The account of the user who created the event, in **'*user\_name*'@'*host\_name*'** format.

**Time zone**

The event time zone, which is the time zone used for scheduling the event and that is in effect within the event as it executes. The default value is **SYSTEM**.

**Type**

The event repetition type, either **ONE TIME** (transient) or **RECURRING** (repeating).

**Execute At**

For a one-time event, this is the [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value specified in the **AT** clause of the [**CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-event) statement used to create the event, or of the last [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) statement that modified the event. The value shown in this column reflects the addition or subtraction of any **INTERVAL** value included in the event's **AT** clause. For example, if an event is created using **ON SCHEDULE AT CURRENT\_TIMESTAMP + '1:6' DAY\_HOUR**, and the event was created at 2018-02-09 14:05:30, the value shown in this column would be **'2018-02-10 20:05:30'**. If the event's timing is determined by an **EVERY** clause instead of an **AT** clause (that is, if the event is recurring), the value of this column is **NULL**.

**Interval Value**

For a recurring event, the number of intervals to wait between event executions. For a transient event, the value of this column is always **NULL**.

**Interval Field**

The time units used for the interval which a recurring event waits before repeating. For a transient event, the value of this column is always **NULL**.

**Starts**

The start date and time for a recurring event. This is displayed as a [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value, and is **NULL** if no start date and time are defined for the event. For a transient event, this column is always **NULL**. For a recurring event whose definition includes a **STARTS** clause, this column contains the corresponding [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value. As with the **Execute At** column, this value resolves any expressions used. If there is no **STARTS** clause affecting the timing of the event, this column is **NULL**

**Ends**

For a recurring event whose definition includes a **ENDS** clause, this column contains the corresponding [**DATETIME**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) value. As with the **Execute At** column, this value resolves any expressions used. If there is no **ENDS** clause affecting the timing of the event, this column is **NULL**.

**Status**

The event status. One of **ENABLED**, **DISABLED**, or **SLAVESIDE\_DISABLED**. **SLAVESIDE\_DISABLED** indicates that the creation of the event occurred on another MySQL server acting as a replication source and replicated to the current MySQL server which is acting as a replica, but the event is not presently being executed on the replica. For more information, see [Section 17.5.1.16, “Replication of Invoked Features”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-invoked). information.

**Originator**

The server ID of the MySQL server on which the event was created; used in replication. This value may be updated by [**ALTER EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-event) to the server ID of the server on which that statement occurs, if executed on a source server. The default value is 0.

**character\_set\_client**

The session value of the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable when the event was created.

**collation\_connection**

The session value of the [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variable when the event was created.

**Database Collation**

The collation of the database with which the event is associated.

For more information about **SLAVESIDE\_DISABLED** and the **Originator** column, see [Section 17.5.1.16, “Replication of Invoked Features”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-invoked).

Times displayed by [**SHOW EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events) are given in the event time zone, as discussed in [Section 25.4.4, “Event Metadata”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#events-metadata).

Event information is also available from the **INFORMATION\_SCHEMA** [**EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table) table. See [Section 26.3.14, “The INFORMATION\_SCHEMA EVENTS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table).

The event action statement is not shown in the output of [**SHOW EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-events). Use [**SHOW CREATE EVENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-event) or the **INFORMATION\_SCHEMA** [**EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-events-table) table.

#### 13.7.7.19 SHOW FUNCTION CODE Statement

SHOW FUNCTION CODE ***func\_name***

This statement is similar to [**SHOW PROCEDURE CODE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-procedure-code) but for stored functions. See [Section 13.7.7.27, “SHOW PROCEDURE CODE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-procedure-code).

#### 13.7.7.20 SHOW FUNCTION STATUS Statement

SHOW FUNCTION STATUS

[LIKE '***pattern***' | WHERE ***expr***]

This statement is similar to [**SHOW PROCEDURE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-procedure-status) but for stored functions. See [Section 13.7.7.28, “SHOW PROCEDURE STATUS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-procedure-status).

#### 13.7.7.21 SHOW GRANTS Statement

SHOW GRANTS

[FOR ***user\_or\_role***

[USING ***role*** [, ***role***] ...]]

***user\_or\_role***: {

***user*** (see [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names))

| ***role*** (see [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names).

}

This statement displays the privileges and roles that are assigned to a MySQL user account or role, in the form of [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statements that must be executed to duplicate the privilege and role assignments.

**Note**

To display nonprivilege information for MySQL accounts, use the [**SHOW CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-user) statement. See [Section 13.7.7.12, “SHOW CREATE USER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-user).

[**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) requires the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege for the **mysql** system schema, except to display privileges and roles for the current user.

To name the account or role for [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants), use the same format as for the [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant) statement (for example, **'jeffrey'@'localhost'**):

mysql> **SHOW GRANTS FOR 'jeffrey'@'localhost';**

+------------------------------------------------------------------+

| Grants for jeffrey@localhost |

+------------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO `jeffrey`@`localhost` |

| GRANT SELECT, INSERT, UPDATE ON `db1`.\* TO `jeffrey`@`localhost` |

+------------------------------------------------------------------+

The host part, if omitted, defaults to **'%'**. For additional information about specifying account and role names, see [Section 6.2.4, “Specifying Account Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-names), and [Section 6.2.5, “Specifying Role Names”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#role-names).

To display the privileges granted to the current user (the account 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 equivalent syntax) is used in definer context, such as within a stored procedure that executes with definer rather than invoker privileges, the grants displayed are those of the definer and not the invoker.

In MySQL 8.0 compared to previous series, [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) no longer displays [**ALL PRIVILEGES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_all) in its global-privileges output because the meaning of [**ALL PRIVILEGES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_all) at the global level varies depending on which dynamic privileges are defined. Instead, [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) explictly lists each granted global privilege:

mysql> **SHOW GRANTS FOR 'root'@'localhost';**

+---------------------------------------------------------------------+

| Grants for root@localhost |

+---------------------------------------------------------------------+

| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, |

| SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DATABASES, |

| SUPER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION |

| SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, |

| ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, CREATE TABLESPACE, |

| CREATE ROLE, DROP ROLE ON \*.\* TO `root`@`localhost` WITH GRANT |

| OPTION |

| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION |

+---------------------------------------------------------------------+

Applications that process [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) output should be adjusted accordingly.

At the global level, [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) applies to all granted static global privileges if granted for any of them, but applies individually to granted dynamic privileges. [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) displays global privileges this way:

One line listing all granted static privileges, if there are any, including **WITH GRANT OPTION** if appropriate.

One line listing all granted dynamic privileges for which [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) is granted, if there are any, including **WITH GRANT OPTION**.

One line listing all granted dynamic privileges for which [**GRANT OPTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_grant-option) is not granted, if there are any, without **WITH GRANT OPTION**.

With the optional **USING** clause, [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) enables you to examine the privileges associated with roles for the user. Each role named in the **USING** clause must be granted to the user.

Suppose that user **u1** is assigned roles **r1** and **r2**, as follows:

CREATE ROLE 'r1', 'r2';

GRANT SELECT ON db1.\* TO 'r1';

GRANT INSERT, UPDATE, DELETE ON db1.\* TO 'r2';

CREATE USER 'u1'@'localhost' IDENTIFIED BY 'u1pass';

GRANT 'r1', 'r2' TO 'u1'@'localhost';

[**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) without **USING** shows the granted roles:

mysql> **SHOW GRANTS FOR 'u1'@'localhost';**

+---------------------------------------------+

| Grants for u1@localhost |

+---------------------------------------------+

| GRANT USAGE ON \*.\* TO `u1`@`localhost` |

| GRANT `r1`@`%`,`r2`@`%` TO `u1`@`localhost` |

+---------------------------------------------+

Adding a **USING** clause causes the statement to also display the privileges associated with each role named in the clause:

mysql> **SHOW GRANTS FOR 'u1'@'localhost' USING 'r1';**

+---------------------------------------------+

| Grants for u1@localhost |

+---------------------------------------------+

| GRANT USAGE ON \*.\* TO `u1`@`localhost` |

| GRANT SELECT ON `db1`.\* TO `u1`@`localhost` |

| GRANT `r1`@`%`,`r2`@`%` TO `u1`@`localhost` |

+---------------------------------------------+

mysql> **SHOW GRANTS FOR 'u1'@'localhost' USING 'r2';**

+-------------------------------------------------------------+

| Grants for u1@localhost |

+-------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO `u1`@`localhost` |

| GRANT INSERT, UPDATE, DELETE ON `db1`.\* TO `u1`@`localhost` |

| GRANT `r1`@`%`,`r2`@`%` TO `u1`@`localhost` |

+-------------------------------------------------------------+

mysql> **SHOW GRANTS FOR 'u1'@'localhost' USING 'r1', 'r2';**

+---------------------------------------------------------------------+

| Grants for u1@localhost |

+---------------------------------------------------------------------+

| GRANT USAGE ON \*.\* TO `u1`@`localhost` |

| GRANT SELECT, INSERT, UPDATE, DELETE ON `db1`.\* TO `u1`@`localhost` |

| GRANT `r1`@`%`,`r2`@`%` TO `u1`@`localhost` |

+---------------------------------------------------------------------+

**Note**

A privilege granted to an account is always in effect, but a role is not. The active roles for an account can differ across and within sessions, depending on the value of the [**activate\_all\_roles\_on\_login**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_activate_all_roles_on_login) system variable, the account default roles, and whether [**SET ROLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-role) has been executed within a session.

MySQL 8.0.16 and higher supports partial revokes of global privileges, such that a global privilege can be restricted from applying to particular schemas (see [Section 6.2.12, “Privilege Restriction Using Partial Revokes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#partial-revokes)). To indicate which global schema privileges have been revoked for particular schemas, **SHOW GRANTS** output includes **REVOKE** statements:

mysql> **SET PERSIST partial\_revokes = ON;**

mysql> **CREATE USER u1;**

mysql> **GRANT SELECT, INSERT, DELETE ON \*.\* TO u1;**

mysql> **REVOKE SELECT, INSERT ON mysql.\* FROM u1;**

mysql> **REVOKE DELETE ON world.\* FROM u1;**

mysql> **SHOW GRANTS FOR u1;**

+--------------------------------------------------+

| Grants for u1@% |

+--------------------------------------------------+

| GRANT SELECT, INSERT, DELETE ON \*.\* TO `u1`@`%` |

| REVOKE SELECT, INSERT ON `mysql`.\* FROM `u1`@`%` |

| REVOKE DELETE ON `world`.\* FROM `u1`@`%` |

+--------------------------------------------------+

[**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) does not display privileges that are available to the named account but are granted to a different account. For example, if an anonymous account exists, the named account might be able to use its privileges, but [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) does not display them.

[**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) displays mandatory roles named in the [**mandatory\_roles**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_mandatory_roles) system variable value as follows:

[**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) without a **FOR** clause displays privileges for the current user, and includes mandatory roles.

[**SHOW GRANTS FOR *user***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) displays privileges for the named user, and does not include mandatory roles.

This behavior is for the benefit of applications that use the output of [**SHOW GRANTS FOR *user***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) to determine which privileges are granted explicitly to the named user. Were that output to include mandatory roles, it would be difficult to distinguish roles granted explicitly to the user from mandatory roles.

For the current user, applications can determine privileges with or without mandatory roles by using [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) or [**SHOW GRANTS FOR CURRENT\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants), respectively.

#### 13.7.7.22 SHOW INDEX Statement

SHOW [EXTENDED] {INDEX | INDEXES | KEYS}

{FROM | IN} ***tbl\_name***

[{FROM | IN} ***db\_name***]

[WHERE ***expr***]

[**SHOW INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index) returns table index information. The format resembles that of the **SQLStatistics** call in ODBC. This statement requires some privilege for any column in the table.

mysql> **SHOW INDEX FROM City\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: city

Non\_unique: 0

Key\_name: PRIMARY

Seq\_in\_index: 1

Column\_name: ID

Collation: A

Cardinality: 4188

Sub\_part: NULL

Packed: NULL

Null:

Index\_type: BTREE

Comment:

Index\_comment:

Visible: YES

Expression: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: city

Non\_unique: 1

Key\_name: CountryCode

Seq\_in\_index: 1

Column\_name: CountryCode

Collation: A

Cardinality: 232

Sub\_part: NULL

Packed: NULL

Null:

Index\_type: BTREE

Comment:

Index\_comment:

Visible: YES

Expression: NULL

An alternative to ***tbl\_name* FROM *db\_name*** syntax is ***db\_name***.***tbl\_name***. These two statements are equivalent:

SHOW INDEX FROM mytable FROM mydb;

SHOW INDEX FROM mydb.mytable;

The optional **EXTENDED** keyword causes the output to include information about hidden indexes that MySQL uses internally and are not accessible by users.

The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

[**SHOW INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index) returns the following fields:

**Table**

The name of the table.

**Non\_unique**

0 if the index cannot contain duplicates, 1 if it can.

**Key\_name**

The name of the index. If the index is the primary key, the name is always **PRIMARY**.

**Seq\_in\_index**

The column sequence number in the index, starting with 1.

**Column\_name**

The column name. See also the description for the **Expression** column.

**Collation**

How the column is sorted in the index. This can have values **A** (ascending), **D** (descending), or **NULL** (not sorted).

**Cardinality**

An estimate of the number of unique values in the index. To update this number, run [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) or (for **MyISAM** tables) [**myisamchk -a**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk).

**Cardinality** is counted based on statistics stored as integers, so the value is not necessarily exact even for small tables. The higher the cardinality, the greater the chance that MySQL uses the index when doing joins.

**Sub\_part**

The index prefix. That is, the number of indexed characters if the column is only partly indexed, **NULL** if the entire column is indexed.

**Note**

Prefix limits are measured in bytes. However, prefix lengths for index specifications in [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table), [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table), and [**CREATE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index) statements are interpreted as number of characters for nonbinary string types ([**CHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**VARCHAR**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#char), [**TEXT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)) and number of bytes for binary string types ([**BINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**VARBINARY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#binary-varbinary), [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob)). Take this into account when specifying a prefix length for a nonbinary string column that uses a multibyte character set.

For additional information about index prefixes, see [Section 8.3.5, “Column Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#column-indexes), and [Section 13.1.15, “CREATE INDEX Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index).

**Packed**

Indicates how the key is packed. **NULL** if it is not.

**Null**

Contains **YES** if the column may contain **NULL** values and **''** if not.

**Index\_type**

The index method used (**BTREE**, **FULLTEXT**, **HASH**, **RTREE**).

**Comment**

Information about the index not described in its own column, such as **disabled** if the index is disabled.

**Index\_comment**

Any comment provided for the index with a **COMMENT** attribute when the index was created.

**Visible**

Whether the index is visible to the optimizer. See [Section 8.3.12, “Invisible Indexes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#invisible-indexes).

**Expression**

MySQL 8.0.13 and higher supports functional key parts (see [Functional Key Parts](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-index-functional-key-parts)), which affects both the **Column\_name** and **Expression** columns:

For a nonfunctional key part, **Column\_name** indicates the column indexed by the key part and **Expression** is **NULL**.

For a functional key part, **Column\_name** column is **NULL** and **Expression** indicates the expression for the key part.

Information about table indexes is also available from the **INFORMATION\_SCHEMA** [**STATISTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-statistics-table) table. See [Section 26.3.34, “The INFORMATION\_SCHEMA STATISTICS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-statistics-table). The extended information about hidden indexes is available only using **SHOW EXTENDED INDEX**; it cannot be obtained from the [**STATISTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-statistics-table) table.

You can list a table's indexes with the [**mysqlshow -k *db\_name* *tbl\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlshow) command.

#### 13.7.7.23 SHOW MASTER STATUS Statement

SHOW MASTER STATUS

This statement provides status information about the binary log files of the source server. It requires the [**REPLICATION CLIENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-client) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

Example:

mysql> **SHOW MASTER STATUS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: source-bin.000002

Position: 1307

Binlog\_Do\_DB: test

Binlog\_Ignore\_DB: manual, mysql

Executed\_Gtid\_Set: 3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5

1 row in set (0.00 sec)

When global transaction IDs are in use, **Executed\_Gtid\_Set** shows the set of GTIDs for transactions that have been executed on the source. This is the same as the value for the [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable on this server, as well as the value for **Executed\_Gtid\_Set** in the output of [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) on this server.

#### 13.7.7.24 SHOW OPEN TABLES Statement

SHOW OPEN TABLES

[{FROM | IN} ***db\_name***]

[LIKE '***pattern***' | WHERE ***expr***]

[**SHOW OPEN TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-open-tables) lists the non-**TEMPORARY** tables that are currently open in the table cache. See [Section 8.4.3.1, “How MySQL Opens and Closes Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#table-cache). The **FROM** clause, if present, restricts the tables shown to those present in the ***db\_name*** database. The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which table names to match. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

[**SHOW OPEN TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-open-tables) output has these columns:

**Database**

The database containing the table.

**Table**

The table name.

**In\_use**

The number of table locks or lock requests there are for the table. For example, if one client acquires a lock for a table using **LOCK TABLE t1 WRITE**, **In\_use** is 1. If another client issues **LOCK TABLE t1 WRITE** while the table remains locked, the client blocks, waiting for the lock, but the lock request causes **In\_use** to be 2. If the count is zero, the table is open but not currently being used. **In\_use** is also increased by the [**HANDLER ... OPEN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler) statement and decreased by [**HANDLER ... CLOSE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler).

**Name\_locked**

Whether the table name is locked. Name locking is used for operations such as dropping or renaming tables.

If you have no privileges for a table, it does not show up in the output from [**SHOW OPEN TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-open-tables).

#### 13.7.7.25 SHOW PLUGINS Statement

SHOW PLUGINS

[**SHOW PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins) displays information about server plugins.

Example of [**SHOW PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins) output:

mysql> **SHOW PLUGINS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: binlog

Status: ACTIVE

Type: STORAGE ENGINE

Library: NULL

License: GPL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: CSV

Status: ACTIVE

Type: STORAGE ENGINE

Library: NULL

License: GPL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: MEMORY

Status: ACTIVE

Type: STORAGE ENGINE

Library: NULL

License: GPL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: MyISAM

Status: ACTIVE

Type: STORAGE ENGINE

Library: NULL

License: GPL

...

[**SHOW PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins) output has these columns:

**Name**

The name used to refer to the plugin in statements such as [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) and [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin).

**Status**

The plugin status, one of **ACTIVE**, **INACTIVE**, **DISABLED**, **DELETING**, or **DELETED**.

**Type**

The type of plugin, such as **STORAGE ENGINE**, **INFORMATION\_SCHEMA**, or **AUTHENTICATION**.

**Library**

The name of the plugin shared library file. This is the name used to refer to the plugin file in statements such as [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) and [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin). This file is located in the directory named by the [**plugin\_dir**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_plugin_dir) system variable. If the library name is **NULL**, the plugin is compiled in and cannot be uninstalled with [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin).

**License**

How the plugin is licensed (for example, **GPL**).

For plugins installed with [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin), the **Name** and **Library** values are also registered in the **mysql.plugin** system table.

For information about plugin data structures that form the basis of the information displayed by [**SHOW PLUGINS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-plugins), see [The MySQL Plugin API](https://dev.mysql.com/doc/extending-mysql/8.0/en/plugin-api.html).

Plugin information is also available from the **INFORMATION\_SCHEMA** **.PLUGINS** table. See [Section 26.3.22, “The INFORMATION\_SCHEMA PLUGINS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-plugins-table).

#### 13.7.7.26 SHOW PRIVILEGES Statement

SHOW PRIVILEGES

[**SHOW PRIVILEGES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-privileges) shows the list of system privileges that the MySQL server supports. The privileges displayed include all static privileges, and all currently registered dynamic privileges.

mysql> **SHOW PRIVILEGES\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Privilege: Alter

Context: Tables

Comment: To alter the table

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Privilege: Alter routine

Context: Functions,Procedures

Comment: To alter or drop stored functions/procedures

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Privilege: Create

Context: Databases,Tables,Indexes

Comment: To create new databases and tables

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Privilege: Create routine

Context: Databases

Comment: To use CREATE FUNCTION/PROCEDURE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Privilege: Create temporary tables

Context: Databases

Comment: To use CREATE TEMPORARY TABLE

...

Privileges belonging to a specific user are displayed by the [**SHOW GRANTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants) statement. See [Section 13.7.7.21, “SHOW GRANTS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-grants), for more information.

#### 13.7.7.27 SHOW PROCEDURE CODE Statement

SHOW PROCEDURE CODE ***proc\_name***

This statement is a MySQL 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-function-code), displays information about stored functions (see [Section 13.7.7.19, “SHOW FUNCTION CODE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-function-code)).

To use either statement, you must be the user named as the routine **DEFINER**, have the [**SHOW\_ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-routine) privilege, or have the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege at the global level.

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.

mysql> **DELIMITER //**

mysql> **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 (0.01 sec)

mysql> **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 |

+-----+----------------------------------------+

4 rows in set (0.00 sec)

mysql> **CREATE FUNCTION test.hello (s CHAR(20))**

**RETURNS CHAR(50) DETERMINISTIC**

**RETURN CONCAT('Hello, ',s,'!');**

Query OK, 0 rows affected (0.00 sec)

mysql> **SHOW FUNCTION CODE test.hello;**

+-----+---------------------------------------+

| Pos | Instruction |

+-----+---------------------------------------+

| 0 | freturn 254 concat('Hello, ',s@0,'!') |

+-----+---------------------------------------+

1 row in set (0.00 sec)

In this example, the nonexecutable **BEGIN** and **END** statements have disappeared, and for the **DECLARE *variable\_name*** statement, only the executable part appears (the part where the default is assigned). For each statement that is taken from source, there is a code word **stmt** followed by a type (9 means **DROP**, 5 means [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), and so on). The final row contains an instruction **jump 2**, meaning **GOTO instruction #2**.

#### 13.7.7.28 SHOW PROCEDURE STATUS Statement

SHOW PROCEDURE STATUS

[LIKE '***pattern***' | WHERE ***expr***]

This statement is a MySQL 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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-function-status), displays information about stored functions (see [Section 13.7.7.20, “SHOW FUNCTION STATUS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-function-status)).

To use either statement, you must be the user named as the routine **DEFINER**, have the [**SHOW\_ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-routine) privilege, have the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privilege at the global level, or have the [**CREATE ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_create-routine), [**ALTER ROUTINE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_alter-routine), or [**EXECUTE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_execute) privilege granted at a scope that includes the routine.

The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which procedure or function names to match. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

mysql> **SHOW PROCEDURE STATUS LIKE 'sp1'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Db: test

Name: sp1

Type: PROCEDURE

Definer: testuser@localhost

Modified: 2018-08-08 13:54:11

Created: 2018-08-08 13:54:11

Security\_type: DEFINER

Comment:

character\_set\_client: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

Database Collation: utf8mb4\_0900\_ai\_ci

mysql> **SHOW FUNCTION STATUS LIKE 'hello'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Db: test

Name: hello

Type: FUNCTION

Definer: testuser@localhost

Modified: 2020-03-10 11:10:03

Created: 2020-03-10 11:10:03

Security\_type: DEFINER

Comment:

character\_set\_client: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

Database Collation: utf8mb4\_0900\_ai\_ci

**character\_set\_client** is the session value of the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable when the routine was created. **collation\_connection** is the session value of the [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variable when the routine was created. **Database Collation** is the collation of the database with which the routine is associated.

Stored routine information is also available from the **INFORMATION\_SCHEMA** [**PARAMETERS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-parameters-table) and [**ROUTINES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-routines-table) tables. See [Section 26.3.20, “The INFORMATION\_SCHEMA PARAMETERS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-parameters-table), and [Section 26.3.30, “The INFORMATION\_SCHEMA ROUTINES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-routines-table).

#### 13.7.7.29 SHOW PROCESSLIST Statement

SHOW [FULL] PROCESSLIST

The MySQL process list indicates the operations currently being performed by the set of threads executing within the server. The [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) statement is one source of process information. For a comparison of this statement with other sources, see [Sources of Process Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#processlist-sources).

**Note**

As of MySQL 8.0.22, an alternative implementation for [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) is available based on the Performance Schema [**processlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-processlist-table) table, which, unlike the default [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) implementation, does not require a mutex and has better performance characteristics. For details, see [Section 27.12.21.5, “The processlist Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-processlist-table).

If you have the [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process) privilege, you can see all threads, even those belonging to other users. Otherwise (without the [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process) privilege), nonanonymous users have access to information about their own threads but not threads for other users, and anonymous users have no access to thread information.

Without the **FULL** keyword, [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) displays only the first 100 characters of each statement in the **Info** field.

The [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) statement is very useful if you get the “too many connections” error message and want to find out what is going on. MySQL reserves one extra connection to be used by accounts that have the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege), to ensure that administrators should always be able to connect and check the system (assuming that you are not giving this privilege to all your users).

Threads can be killed with the [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) statement. See [Section 13.7.8.4, “KILL Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill).

Example of [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) output:

mysql> SHOW FULL PROCESSLIST\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 1

User: system user

Host:

db: NULL

Command: Connect

Time: 1030455

State: Waiting for master to send event

Info: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 2

User: system user

Host:

db: NULL

Command: Connect

Time: 1004

State: Has read all relay log; waiting for the slave

I/O thread to update it

Info: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 3112

User: replikator

Host: artemis:2204

db: NULL

Command: Binlog Dump

Time: 2144

State: Has sent all binlog to slave; waiting for binlog to be updated

Info: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 3113

User: replikator

Host: iconnect2:45781

db: NULL

Command: Binlog Dump

Time: 2086

State: Has sent all binlog to slave; waiting for binlog to be updated

Info: NULL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Id: 3123

User: stefan

Host: localhost

db: apollon

Command: Query

Time: 0

State: NULL

Info: SHOW FULL PROCESSLIST

[**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) output has these columns:

**Id**

The connection identifier. This is the same value displayed in the **ID** column of the **INFORMATION\_SCHEMA** [**PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table) table, displayed in the **PROCESSLIST\_ID** column of the Performance Schema [**threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-threads-table) table, and returned by the [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id) function within the thread.

**User**

The MySQL user who issued the statement. A value of **system user** refers to a nonclient thread spawned by the server to handle tasks internally, for example, a delayed-row handler thread or an I/O or SQL thread used on replica hosts. For **system user**, there is no host specified in the **Host** column. **unauthenticated user** refers to a thread that has become associated with a client connection but for which authentication of the client user has not yet occurred. **event\_scheduler** refers to the thread that monitors scheduled events (see [Section 25.4, “Using the Event Scheduler”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#event-scheduler)).

**Note**

A **User** value of **system user** is distinct from the [**SYSTEM\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-user) privilege. The former designates internal threads. The latter distinguishes the system user and regular user account categories (see [Section 6.2.11, “Account Categories”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#account-categories)).

**Host**

The host name of the client issuing the statement (except for **system user**, for which there is no host). The host name for TCP/IP connections is reported in ***host\_name*:*client\_port*** format to make it easier to determine which client is doing what.

**db**

The default database for the thread, or **NULL** if none has been selected.

**Command**

The type of command the thread is executing on behalf of the client, or **Sleep** if the session is idle. For descriptions of thread commands, see [Section 8.14, “Examining Server Thread (Process) Information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#thread-information). The value of this column corresponds to the **COM\_*xxx*** commands of the client/server protocol and **Com\_*xxx*** status variables. See [Section 5.1.10, “Server Status Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-status-variables).

**Time**

The time in seconds that the thread has been in its current state. For a replica SQL thread, the value is the number of seconds between the timestamp of the last replicated event and the real time of the replica host. See [Section 17.2.3, “Replication Threads”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-implementation-details).

**State**

An action, event, or state that indicates what the thread is doing. For descriptions of **State** values, see [Section 8.14, “Examining Server Thread (Process) Information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#thread-information).

Most states correspond to very quick operations. If a thread stays in a given state for many seconds, there might be a problem that needs to be investigated.

**Info**

The statement the thread is executing, or **NULL** if it is executing no statement. The statement might be the one sent to the server, or an innermost statement if the statement executes other statements. For example, if a **CALL** statement executes a stored procedure that is executing a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, the **Info** value shows the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement.

#### 13.7.7.30 SHOW PROFILE Statement

SHOW PROFILE [***type*** [, ***type***] ... ]

[FOR QUERY ***n***]

[LIMIT ***row\_count*** [OFFSET ***offset***]]

***type***: {

ALL

| BLOCK IO

| CONTEXT SWITCHES

| CPU

| IPC

| MEMORY

| PAGE FAULTS

| SOURCE

| SWAPS

}

The [**SHOW PROFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile) and [**SHOW PROFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profiles) statements display profiling information that indicates resource usage for statements executed during the course of the current session.

**Note**

The [**SHOW PROFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile) and [**SHOW PROFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profiles) statements are deprecated; expect them to be removed in a future MySQL release. Use the [Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html) instead; see [Section 27.19.1, “Query Profiling Using Performance Schema”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-query-profiling).

To control profiling, use the [**profiling**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_profiling) session variable, which has a default value of 0 (**OFF**). Enable profiling by setting [**profiling**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_profiling) to 1 or **ON**:

mysql> **SET profiling = 1;**

[**SHOW PROFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profiles) displays a list of the most recent statements sent to the server. The size of the list is controlled by the [**profiling\_history\_size**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_profiling_history_size) session variable, which has a default value of 15. The maximum value is 100. Setting the value to 0 has the practical effect of disabling profiling.

All statements are profiled except [**SHOW PROFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile) and [**SHOW PROFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profiles), so neither of those statements appears in the profile list. Malformed statements are profiled. For example, **SHOW PROFILING** is an illegal statement, and a syntax error occurs if you try to execute it, but it shows up in the profiling list.

[**SHOW PROFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile) displays detailed information about a single statement. Without the **FOR QUERY *n*** clause, the output pertains to the most recently executed statement. If **FOR QUERY *n*** is included, [**SHOW PROFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile) displays information for statement ***n***. The values of ***n*** correspond to the **Query\_ID** values displayed by [**SHOW PROFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profiles).

The **LIMIT *row\_count*** clause may be given to limit the output to ***row\_count*** rows. If **LIMIT** is given, **OFFSET *offset*** may be added to begin the output ***offset*** rows into the full set of rows.

By default, [**SHOW PROFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile) displays **Status** and **Duration** columns. The **Status** values are like the **State** values displayed by [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist), although there might be some minor differences in interpretion for the two statements for some status values (see [Section 8.14, “Examining Server Thread (Process) Information”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#thread-information)).

Optional ***type*** values may be specified to display specific additional types of information:

**ALL** displays all information

**BLOCK IO** displays counts for block input and output operations

**CONTEXT SWITCHES** displays counts for voluntary and involuntary context switches

**CPU** displays user and system CPU usage times

**IPC** displays counts for messages sent and received

**MEMORY** is not currently implemented

**PAGE FAULTS** displays counts for major and minor page faults

**SOURCE** displays the names of functions from the source code, together with the name and line number of the file in which the function occurs

**SWAPS** displays swap counts

Profiling is enabled per session. When a session ends, its profiling information is lost.

mysql> **SELECT @@profiling;**

+-------------+

| @@profiling |

+-------------+

| 0 |

+-------------+

1 row in set (0.00 sec)

mysql> **SET profiling = 1;**

Query OK, 0 rows affected (0.00 sec)

mysql> **DROP TABLE IF EXISTS t1;**

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> **CREATE TABLE T1 (id INT);**

Query OK, 0 rows affected (0.01 sec)

mysql> **SHOW PROFILES;**

+----------+----------+--------------------------+

| Query\_ID | Duration | Query |

+----------+----------+--------------------------+

| 0 | 0.000088 | SET PROFILING = 1 |

| 1 | 0.000136 | DROP TABLE IF EXISTS t1 |

| 2 | 0.011947 | CREATE TABLE t1 (id INT) |

+----------+----------+--------------------------+

3 rows in set (0.00 sec)

mysql> **SHOW PROFILE;**

+----------------------+----------+

| Status | Duration |

+----------------------+----------+

| checking permissions | 0.000040 |

| creating table | 0.000056 |

| After create | 0.011363 |

| query end | 0.000375 |

| freeing items | 0.000089 |

| logging slow query | 0.000019 |

| cleaning up | 0.000005 |

+----------------------+----------+

7 rows in set (0.00 sec)

mysql> **SHOW PROFILE FOR QUERY 1;**

+--------------------+----------+

| Status | Duration |

+--------------------+----------+

| query end | 0.000107 |

| freeing items | 0.000008 |

| logging slow query | 0.000015 |

| cleaning up | 0.000006 |

+--------------------+----------+

4 rows in set (0.00 sec)

mysql> **SHOW PROFILE CPU FOR QUERY 2;**

+----------------------+----------+----------+------------+

| Status | Duration | CPU\_user | CPU\_system |

+----------------------+----------+----------+------------+

| checking permissions | 0.000040 | 0.000038 | 0.000002 |

| creating table | 0.000056 | 0.000028 | 0.000028 |

| After create | 0.011363 | 0.000217 | 0.001571 |

| query end | 0.000375 | 0.000013 | 0.000028 |

| freeing items | 0.000089 | 0.000010 | 0.000014 |

| logging slow query | 0.000019 | 0.000009 | 0.000010 |

| cleaning up | 0.000005 | 0.000003 | 0.000002 |

+----------------------+----------+----------+------------+

7 rows in set (0.00 sec)

**Note**

Profiling is only partially functional on some architectures. For values that depend on the **getrusage()** system call, **NULL** is returned on systems such as Windows that do not support the call. In addition, profiling is per process and not per thread. This means that activity on threads within the server other than your own may affect the timing information that you see.

Profiling information is also available from the **INFORMATION\_SCHEMA** [**PROFILING**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-profiling-table) table. See [Section 26.3.24, “The INFORMATION\_SCHEMA PROFILING Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-profiling-table). For example, the following queries are equivalent:

SHOW PROFILE FOR QUERY 2;

SELECT STATE, FORMAT(DURATION, 6) AS DURATION

FROM INFORMATION\_SCHEMA.PROFILING

WHERE QUERY\_ID = 2 ORDER BY SEQ;

#### 13.7.7.31 SHOW PROFILES Statement

SHOW PROFILES

The [**SHOW PROFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profiles) statement, together with [**SHOW PROFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile), displays profiling information that indicates resource usage for statements executed during the course of the current session. For more information, see [Section 13.7.7.30, “SHOW PROFILE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile).

**Note**

The [**SHOW PROFILE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profile) and [**SHOW PROFILES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-profiles) statements are deprecated; expect it to be removed in a future MySQL release. Use the [Performance Schema](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html) instead; see [Section 27.19.1, “Query Profiling Using Performance Schema”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-query-profiling).

#### 13.7.7.32 SHOW RELAYLOG EVENTS Statement

SHOW RELAYLOG EVENTS

[IN '***log\_name***']

[FROM ***pos***]

[LIMIT [***offset***,] ***row\_count***]

[***channel\_option***]

***channel\_option***:

FOR CHANNEL ***channel***

Shows the events in the relay log of a replica. If you do not specify **'*log\_name*'**, the first relay log is displayed. This statement has no effect on the source. **SHOW RELAYLOG EVENTS** requires the [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) privilege.

The **LIMIT** clause has the same syntax as for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. See [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

**Note**

Issuing a [**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events) with no **LIMIT** clause could start a very time- and resource-consuming process because the server returns to the client the complete contents of the relay log (including all statements modifying data that have been received by the replica).

The optional **FOR CHANNEL *channel*** clause enables you to name which replication channel the statement applies to. Providing a **FOR CHANNEL *channel*** clause applies the statement to a specific replication channel. If no channel is named and no extra channels exist, the statement applies to the default channel.

When using multiple replication channels, if a [**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events) statement does not have a channel defined using a **FOR CHANNEL *channel*** clause an error is generated. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels) for more information.

[**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events) displays the following fields for each event in the relay log:

**Log\_name**

The name of the file that is being listed.

**Pos**

The position at which the event occurs.

**Event\_type**

An identifier that describes the event type.

**Server\_id**

The server ID of the server on which the event originated.

**End\_log\_pos**

The value of **End\_log\_pos** for this event in the source's binary log.

**Info**

More detailed information about the event type. The format of this information depends on the event type.

For compressed transaction payloads, the **Transaction\_payload\_event** is first printed as a single unit, then it is unpacked and each event inside it is printed.

Some events relating to the setting of user and system variables are not included in the output from [**SHOW RELAYLOG EVENTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-relaylog-events). To get complete coverage of events within a relay log, use [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog).

#### 13.7.7.33 SHOW REPLICAS | SHOW SLAVE HOSTS Statement

{SHOW REPLICAS | SHOW SLAVE HOSTS}

Displays a list of replicas currently registered with the source. From MySQL 8.0.22, use [**SHOW REPLICAS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas) in place of [**SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-hosts), which is deprecated from that release. In releases before MySQL 8.0.22, use [**SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-hosts). **SHOW REPLICAS | SHOW SLAVE HOSTS** requires the [**REPLICATION SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-slave) privilege.

**SHOW REPLICAS | SHOW SLAVE HOSTS** should be executed on a server that acts as a replication source. The statement displays information about servers that are or have been connected as replicas, with each row of the result corresponding to one replica server, as shown here:

mysql> **SHOW REPLICAS**;

+------------+-----------+------+-----------+--------------------------------------+

| Server\_id | Host | Port | Source\_id | Replica\_UUID |

+------------+-----------+------+-----------+--------------------------------------+

| 10 | iconnect2 | 3306 | 3 | 14cb6624-7f93-11e0-b2c0-c80aa9429562 |

| 21 | athena | 3306 | 3 | 07af4990-f41f-11df-a566-7ac56fdaf645 |

+------------+-----------+------+-----------+--------------------------------------+

**Server\_id**: The unique server ID of the replica server, as configured in the replica server's option file, or on the command line with [--server-id=***value***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id).

**Host**: The host name of the replica server, as specified on the replica with the [--report-host](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_host) option. This can differ from the machine name as configured in the operating system.

**User**: The replica server user name, as specified on the replica with the [--report-user](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_user) option. Statement output includes this column only if the source server is started with the [--show-slave-auth-info](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_show-slave-auth-info) option.

**Password**: The replica server password, as specified on the replica with the [--report-password](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_password) option. Statement output includes this column only if the source server is started with the [--show-slave-auth-info](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_show-slave-auth-info) option.

**Port**: The port on the source to which the replica server is listening, as specified on the replica with the [--report-port](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port) option.

A zero in this column means that the replica port ([--report-port](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_report_port)) was not set.

**Source\_id**: The unique server ID of the source server that the replica server is replicating from. This is the server ID of the server on which **SHOW REPLICAS | SHOW SLAVE HOSTS** is executed, so this same value is listed for each row in the result.

**Replica\_UUID**: The globally unique ID of this replica, as generated on the replica and found in the replica's auto.cnf file.

#### 13.7.7.34 SHOW SLAVE HOSTS | SHOW REPLICAS Statement

{SHOW SLAVE HOSTS | SHOW REPLICAS}

Displays a list of replicas currently registered with the source. From MySQL 8.0.22, [**SHOW SLAVE HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-hosts) is deprecated and the alias [**SHOW REPLICAS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas) should be used instead. The statement works in the same way as before, only the terminology used for the statement and its output has changed. Both versions of the statement update the same status variables when used. Please see the documentation for [**SHOW REPLICAS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replicas) for a description of the statement.

#### 13.7.7.35 SHOW REPLICA | SLAVE STATUS Statement

SHOW {REPLICA | SLAVE} STATUS [FOR CHANNEL ***channel***]

This statement provides status information on essential parameters of the replica threads. From MySQL 8.0.22, use [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) in place of [**SHOW SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-status), which is deprecated from that release. In releases before MySQL 8.0.22, use [**SHOW SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-status). The statement requires the [**REPLICATION CLIENT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-client) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

**SHOW REPLICA | SLAVE STATUS** is nonblocking. When run concurrently with [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica), **SHOW REPLICA | SLAVE STATUS** returns without waiting for [**STOP REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#stop-replica) to finish shutting down the replication SQL thread or replication I/O thread (or both). This permits use in monitoring and other applications where getting an immediate response from **SHOW REPLICA | SLAVE STATUS** is more important than ensuring that it returned the latest data.

If you issue this statement using the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client, you can use a **\G** statement terminator rather than a semicolon to obtain a more readable vertical layout:

mysql> **SHOW REPLICA STATUS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Replica\_IO\_State: Waiting for source to send event

Source\_Host: localhost

Source\_User: repl

Source\_Port: 13000

Connect\_Retry: 60

Source\_Log\_File: source-bin.000002

Read\_Source\_Log\_Pos: 1307

Relay\_Log\_File: replica-relay-bin.000003

Relay\_Log\_Pos: 1508

Relay\_Source\_Log\_File: source-bin.000002

Replica\_IO\_Running: Yes

Replica\_SQL\_Running: Yes

Replicate\_Do\_DB:

Replicate\_Ignore\_DB:

Replicate\_Do\_Table:

Replicate\_Ignore\_Table:

Replicate\_Wild\_Do\_Table:

Replicate\_Wild\_Ignore\_Table:

Last\_Errno: 0

Last\_Error:

Skip\_Counter: 0

Exec\_Source\_Log\_Pos: 1307

Relay\_Log\_Space: 1858

Until\_Condition: None

Until\_Log\_File:

Until\_Log\_Pos: 0

Source\_SSL\_Allowed: No

Source\_SSL\_CA\_File:

Source\_SSL\_CA\_Path:

Source\_SSL\_Cert:

Source\_SSL\_Cipher:

Source\_SSL\_Key:

Seconds\_Behind\_Source: 0

Source\_SSL\_Verify\_Server\_Cert: No

Last\_IO\_Errno: 0

Last\_IO\_Error:

Last\_SQL\_Errno: 0

Last\_SQL\_Error:

Replicate\_Ignore\_Server\_Ids:

Source\_Server\_Id: 1

Source\_UUID: 3e11fa47-71ca-11e1-9e33-c80aa9429562

Source\_Info\_File:

SQL\_Delay: 0

SQL\_Remaining\_Delay: NULL

Replica\_SQL\_Running\_State: Reading event from the relay log

Source\_Retry\_Count: 10

Source\_Bind:

Last\_IO\_Error\_Timestamp:

Last\_SQL\_Error\_Timestamp:

Source\_SSL\_Crl:

Source\_SSL\_Crlpath:

Retrieved\_Gtid\_Set: 3e11fa47-71ca-11e1-9e33-c80aa9429562:1-5

Executed\_Gtid\_Set: 3e11fa47-71ca-11e1-9e33-c80aa9429562:1-5

Auto\_Position: 1

Replicate\_Rewrite\_DB:

Channel\_name:

Source\_TLS\_Version: TLSv1.2

Source\_public\_key\_path: public\_key.pem

Get\_source\_public\_key: 0

Network\_Namespace:

The Performance Schema provides tables that expose replication information. This is similar to the information available from the [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) statement, but represented in table form. For details, see [Section 27.12.11, “Performance Schema Replication Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-tables).

The following list describes the fields returned by [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status). For additional information about interpreting their meanings, see [Section 17.1.7.1, “Checking Replication Status”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-administration-status).

**Replica\_IO\_State**

A copy of the **State** field of the [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) output for the replica I/O thread. This tells you what the thread is doing: trying to connect to the source, waiting for events from the source, reconnecting to the source, and so on. For a listing of possible states, see [Section 8.14.5, “Replication I/O Thread States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#replica-io-thread-states).

**Source\_Host**

The source host that the replica is connected to.

**Source\_User**

The user name of the account used to connect to the source.

**Source\_Port**

The port used to connect to the source.

**Connect\_Retry**

The number of seconds between connect retries (default 60). This can be set with a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23).

**Source\_Log\_File**

The name of the source binary log file from which the I/O thread is currently reading.

**Read\_Source\_Log\_Pos**

The position in the current source binary log file up to which the I/O thread has read.

**Relay\_Log\_File**

The name of the relay log file from which the SQL thread is currently reading and executing.

**Relay\_Log\_Pos**

The position in the current relay log file up to which the SQL thread has read and executed.

**Relay\_Source\_Log\_File**

The name of the source binary log file containing the most recent event executed by the SQL thread.

**Replica\_IO\_Running**

Whether the replication I/O thread is started and has connected successfully to the source. Internally, the state of this thread is represented by one of the following three values:

**MYSQL\_REPLICA\_NOT\_RUN.** The replication I/O thread is not running. For this state, **Replica\_IO\_Running** is **No**.

**MYSQL\_REPLICA\_RUN\_NOT\_CONNECT.** The replication I/O thread is running, but is not connected to a replication source. For this state, **Replica\_IO\_Running** is **Connecting**.

**MYSQL\_REPLICA\_RUN\_CONNECT.** The replication I/O thread is running, and is connected to a replication source. For this state, **Replica\_IO\_Running** is **Yes**.

**Replica\_SQL\_Running**

Whether the replication SQL thread is started.

**Replicate\_Do\_DB**, **Replicate\_Ignore\_DB**

The names of any databases that were specified with the [--replicate-do-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-db) and [--replicate-ignore-db](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-db) options, or the [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement. If the **FOR CHANNEL** clause was used, the channel specific replication filters are shown. Otherwise, the replication filters for every replication channel are shown.

**Replicate\_Do\_Table**, **Replicate\_Ignore\_Table**, **Replicate\_Wild\_Do\_Table**, **Replicate\_Wild\_Ignore\_Table**

The names of any tables that were specified with the [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table), [--replicate-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-ignore-table), [--replicate-wild-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-do-table), and [--replicate-wild-ignore-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-wild-ignore-table) options, or the [**CHANGE REPLICATION FILTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter) statement. If the **FOR CHANNEL** clause was used, the channel specific replication filters are shown. Otherwise, the replication filters for every replication channel are shown.

**Last\_Errno**, **Last\_Error**

These columns are aliases for **Last\_SQL\_Errno** and **Last\_SQL\_Error**.

Issuing [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) or [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) resets the values shown in these columns.

**Note**

When the replication SQL thread receives an error, it reports the error first, then stops the SQL thread. This means that there is a small window of time during which **SHOW REPLICA | SLAVE STATUS** shows a nonzero value for **Last\_SQL\_Errno** even though **Replica\_SQL\_Running** still displays **Yes**.

**Skip\_Counter**

The current value of the [**sql\_slave\_skip\_counter**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_sql_slave_skip_counter) system variable. See [SET GLOBAL sql\_slave\_skip\_counter Statement](https://dev.mysql.com/doc/refman/5.6/en/set-global-sql-slave-skip-counter.html).

**Exec\_Source\_Log\_Pos**

The position in the current source binary log file to which the replication SQL thread has read and executed, marking the start of the next transaction or event to be processed. You can use this value with the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement's **SOURCE\_LOG\_POS** option (from MySQL 8.0.23) or the [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement's **MASTER\_LOG\_POS** option (before MySQL 8.0.23) when starting a new replica from an existing replica, so that the new replica reads from this point. The coordinates given by (**Relay\_Source\_Log\_File**, **Exec\_Source\_Log\_Pos**) in the source's binary log correspond to the coordinates given by (**Relay\_Log\_File**, **Relay\_Log\_Pos**) in the relay log.

Inconsistencies in the sequence of transactions from the relay log which have been executed can cause this value to be a “low-water mark”. In other words, transactions appearing before the position are guaranteed to have committed, but transactions after the position may have committed or not. If these gaps need to be corrected, use [**START REPLICA | SLAVE UNTIL SQL\_AFTER\_MTS\_GAPS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica). See [Section 17.5.1.34, “Replication and Transaction Inconsistencies”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-features-transaction-inconsistencies) for more information.

**Relay\_Log\_Space**

The total combined size of all existing relay log files.

**Until\_Condition**, **Until\_Log\_File**, **Until\_Log\_Pos**

The values specified in the **UNTIL** clause of the [**START REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-replica) statement.

**Until\_Condition** has these values:

**None** if no **UNTIL** clause was specified.

**Source** if the replica is reading until a given position in the source's binary log.

**Relay** if the replica is reading until a given position in its relay log.

**SQL\_BEFORE\_GTIDS** if the replication SQL thread is processing transactions until it has reached the first transaction whose GTID is listed in the **gtid\_set**.

**SQL\_AFTER\_GTIDS** if the replication threads are processing all transactions until the last transaction in the **gtid\_set** has been processed by both threads.

**SQL\_AFTER\_MTS\_GAPS** if a multithreaded replica's SQL threads are running until no more gaps are found in the relay log.

**Until\_Log\_File** and **Until\_Log\_Pos** indicate the log file name and position that define the coordinates at which the replication SQL thread stops executing.

For more information on **UNTIL** clauses, see [Section 13.4.2.8, “START SLAVE | REPLICA Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#start-slave).

**Source\_SSL\_Allowed**, **Source\_SSL\_CA\_File**, **Source\_SSL\_CA\_Path**, **Source\_SSL\_Cert**, **Source\_SSL\_Cipher**, **Source\_SSL\_CRL\_File**, **Source\_SSL\_CRL\_Path**, **Source\_SSL\_Key**, **Source\_SSL\_Verify\_Server\_Cert**

These fields show the SSL parameters used by the replica to connect to the source, if any.

**Source\_SSL\_Allowed** has these values:

**Yes** if an SSL connection to the source is permitted.

**No** if an SSL connection to the source is not permitted.

**Ignored** if an SSL connection is permitted but the replica server does not have SSL support enabled.

The values of the other SSL-related fields correspond to the values of the **SOURCE\_SSL\_\*** options of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23), or the **MASTER\_SSL\_\*** options of the [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23). See [Section 13.4.2.1, “CHANGE MASTER TO Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to).

**Seconds\_Behind\_Source**

This field is an indication of how “late” the replica is:

When the replica is actively processing updates, this field shows the difference between the current timestamp on the replica and the original timestamp logged on the source for the event currently being processed on the replica.

When no event is currently being processed on the replica, this value is 0.

In essence, this field measures the time difference in seconds between the replication SQL thread and the replication I/O thread. If the network connection between source and replica is fast, the replication I/O thread is very close to the source, so this field is a good approximation of how late the replication SQL thread is compared to the source. If the network is slow, this is not a good approximation; the replication SQL thread may quite often be caught up with the slow-reading replication I/O thread, so **Seconds\_Behind\_Source** often shows a value of 0, even if the replication I/O thread is late compared to the source. In other words, this column is useful only for fast networks.

This time difference computation works even if the source and replica do not have identical clock times, provided that the difference, computed when the replica I/O thread starts, remains constant from then on. Any changes, including NTP updates, can lead to clock skews that can make calculation of **Seconds\_Behind\_Source** less reliable.

In MySQL 8.0, this field is **NULL** (undefined or unknown) if the replication SQL thread is not running, or if the SQL thread has consumed all of the relay log and the replication I/O thread is not running. (In older versions of MySQL, this field was **NULL** if the replication SQL thread or the replication I/O thread was not running or was not connected to the source.) If the replication I/O thread is running but the relay log is exhausted, **Seconds\_Behind\_Source** is set to 0.

The value of **Seconds\_Behind\_Source** is based on the timestamps stored in events, which are preserved through replication. This means that if a source M1 is itself a replica of M0, any event from M1's binary log that originates from M0's binary log has M0's timestamp for that event. This enables MySQL to replicate [**TIMESTAMP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#datetime) successfully. However, the problem for **Seconds\_Behind\_Source** is that if M1 also receives direct updates from clients, the **Seconds\_Behind\_Source** value randomly fluctuates because sometimes the last event from M1 originates from M0 and sometimes is the result of a direct update on M1.

When using a multithreaded replica, you should keep in mind that this value is based on **Exec\_Source\_Log\_Pos**, and so may not reflect the position of the most recently committed transaction.

**Last\_IO\_Errno**, **Last\_IO\_Error**

The error number and error message of the most recent error that caused the replication I/O thread to stop. An error number of 0 and message of the empty string mean “no error.” If the **Last\_IO\_Error** value is not empty, the error values also appear in the replica's error log.

I/O error information includes a timestamp showing when the most recent I/O thread error occurred. This timestamp uses the format ***YYMMDD hh:mm:ss***, and appears in the **Last\_IO\_Error\_Timestamp** column.

Issuing [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) or [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) resets the values shown in these columns.

**Last\_SQL\_Errno**, **Last\_SQL\_Error**

The error number and error message of the most recent error that caused the replication SQL thread to stop. An error number of 0 and message of the empty string mean “no error.” If the **Last\_SQL\_Error** value is not empty, the error values also appear in the replica's error log.

If the replica is multithreaded, the replication SQL thread is the coordinator for worker threads. In this case, the **Last\_SQL\_Error** field shows exactly what the **Last\_Error\_Message** column in the Performance Schema [**replication\_applier\_status\_by\_coordinator**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-coordinator-table) table shows. The field value is modified to suggest that there may be more failures in the other worker threads which can be seen in the [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) table that shows each worker thread's status. If that table is not available, the replica error log can be used. The log or the [**replication\_applier\_status\_by\_worker**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-replication-applier-status-by-worker-table) table should also be used to learn more about the failure shown by [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) or the coordinator table.

SQL error information includes a timestamp showing when the most recent SQL thread error occurred. This timestamp uses the format ***YYMMDD hh:mm:ss***, and appears in the **Last\_SQL\_Error\_Timestamp** column.

Issuing [**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master) or [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) resets the values shown in these columns.

In MySQL 8.0, all error codes and messages displayed in the **Last\_SQL\_Errno** and **Last\_SQL\_Error** columns correspond to error values listed in [Server Error Message Reference](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html). This was not always true in previous versions. (Bug #11760365, Bug #52768)

**Replicate\_Ignore\_Server\_Ids**

Any server IDs that have been specified using the **IGNORE\_SERVER\_IDS** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement, so that the replica ignores events from these servers. This option is used in a circular or other multi-source replication setup when one of the servers is removed. If any server IDs have been set in this way, a comma-delimited list of one or more numbers is shown. If no server IDs have been set, the field is blank.

**Note**

The **Ignored\_server\_ids** value in the **slave\_master\_info** table also shows the server IDs to be ignored, but as a space-delimited list, preceded by the total number of server IDs to be ignored. For example, if a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement containing the **IGNORE\_SERVER\_IDS = (2,6,9)** option has been issued to tell a replica to ignore sources having the server ID 2, 6, or 9, that information appears as shown here:

Replicate\_Ignore\_Server\_Ids: 2, 6, 9

Ignored\_server\_ids: 3, 2, 6, 9

**Replicate\_Ignore\_Server\_Ids** filtering is performed by the I/O thread, rather than by the SQL thread, which means that events which are filtered out are not written to the relay log. This differs from the filtering actions taken by server options such [--replicate-do-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_replicate-do-table), which apply to the SQL thread.

**Note**

From MySQL 8.0, a deprecation warning is issued if **SET GTID\_MODE=ON** is issued when any channel has existing server IDs set with **IGNORE\_SERVER\_IDS**. Before starting GTID-based replication, use [**SHOW REPLICA | SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) to check for and clear all ignored server ID lists on the servers involved. You can clear a list by issuing a [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement containing the **IGNORE\_SERVER\_IDS** option with an empty list.

**Source\_Server\_Id**

The [**server\_id**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_id) value from the source.

**Source\_UUID**

The [**server\_uuid**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_server_uuid) value from the source.

**Source\_Info\_File**

The location of the master.info file, the use of which is now deprecated. By default from MySQL 8.0, a table is used instead for the replica's connection metadata repository.

**SQL\_Delay**

The number of seconds that the replica must lag the source.

**SQL\_Remaining\_Delay**

When **Replica\_SQL\_Running\_State** is **Waiting until MASTER\_DELAY seconds after source executed event**, this field contains the number of delay seconds remaining. At other times, this field is **NULL**.

**Replica\_SQL\_Running\_State**

The state of the SQL thread (analogous to **Replica\_IO\_State**). The value is identical to the **State** value of the SQL thread as displayed by [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist). [Section 8.14.6, “Replication SQL Thread States”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#replica-sql-thread-states), provides a listing of possible states.

**Source\_Retry\_Count**

The number of times the replica can attempt to reconnect to the source in the event of a lost connection. This value can be set using the **SOURCE\_RETRY\_COUNT** | **MASTER\_RETRY\_COUNT** option of the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23), or the older [--master-retry-count](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#option_mysqld_master-retry-count) server option (still supported for backward compatibility).

**Source\_Bind**

The network interface that the replica is bound to, if any. This is set using the **SOURCE\_BIND** | **MASTER\_BIND** option for the [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) statement (from MySQL 8.0.23) or [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to) statement (before MySQL 8.0.23).

**Last\_IO\_Error\_Timestamp**

A timestamp in ***YYMMDD hh:mm:ss*** format that shows when the most recent I/O error took place.

**Last\_SQL\_Error\_Timestamp**

A timestamp in ***YYMMDD hh:mm:ss*** format that shows when the most recent SQL error occurred.

**Retrieved\_Gtid\_Set**

The set of global transaction IDs corresponding to all transactions received by this replica. Empty if GTIDs are not in use. See [GTID Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts-gtid-sets) for more information.

This is the set of all GTIDs that exist or have existed in the relay logs. Each GTID is added as soon as the **Gtid\_log\_event** is received. This can cause partially transmitted transactions to have their GTIDs included in the set.

When all relay logs are lost due to executing [**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica) or [**CHANGE REPLICATION SOURCE TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-source-to) | [**CHANGE MASTER TO**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-master-to), or due to the effects of the [--relay-log-recovery](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_recovery) option, the set is cleared. When [**relay\_log\_purge = 1**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_relay_log_purge), the newest relay log is always kept, and the set is not cleared.

**Executed\_Gtid\_Set**

The set of global transaction IDs written in the binary log. This is the same as the value for the global [**gtid\_executed**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#sysvar_gtid_executed) system variable on this server, as well as the value for **Executed\_Gtid\_Set** in the output of [**SHOW MASTER STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-master-status) on this server. Empty if GTIDs are not in use. See [GTID Sets](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-gtids-concepts-gtid-sets) for more information.

**Auto\_Position**

1 if GTID auto-positioning is in use for the channel, otherwise 0.

**Replicate\_Rewrite\_DB**

The **Replicate\_Rewrite\_DB** value displays any replication filtering rules that were specified. For example, if the following replication filter rule was set:

CHANGE REPLICATION FILTER REPLICATE\_REWRITE\_DB=((db1,db2), (db3,db4));

the **Replicate\_Rewrite\_DB** value displays:

Replicate\_Rewrite\_DB: (db1,db2),(db3,db4)

For more information, see [Section 13.4.2.2, “CHANGE REPLICATION FILTER Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#change-replication-filter).

**Channel\_name**

The replication channel which is being displayed. There is always a default replication channel, and more replication channels can be added. See [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels) for more information.

**Master\_TLS\_Version**

The TLS version used on the source. For TLS version information, see [Section 6.3.2, “Encrypted Connection TLS Protocols and Ciphers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#encrypted-connection-protocols-ciphers).

**Source\_public\_key\_path**

The path name to a file containing a replica-side copy of the public key required by the source for RSA key pair-based password exchange. The file must be in PEM format. This column applies to replicas that authenticate with the **sha256\_password** or **caching\_sha2\_password** authentication plugin.

If **Source\_public\_key\_path** is given and specifies a valid public key file, it takes precedence over **Get\_source\_public\_key**.

**Get\_source\_public\_key**

Whether to request from the source the public key required for RSA key pair-based password exchange. This column applies to replicas that authenticate with the **caching\_sha2\_password** authentication plugin. For that plugin, the source does not send the public key unless requested.

If **Source\_public\_key\_path** is given and specifies a valid public key file, it takes precedence over **Get\_source\_public\_key**.

**Network\_Namespace**

The network namespace name; empty if the connection uses the default (global) namespace. For information about network namespaces, see [Section 5.1.14, “Network Namespace Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#network-namespace-support). This column was added in MySQL 8.0.22.

#### 13.7.7.36 SHOW SLAVE | REPLICA STATUS Statement

SHOW {SLAVE | REPLICA} STATUS [FOR CHANNEL ***channel***]

This statement provides status information on essential parameters of the replica threads. From MySQL 8.0.22, [**SHOW SLAVE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-slave-status) is deprecated and the alias [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) should be used instead. The statement works in the same way as before, only the terminology used for the statement and its output has changed. Both versions of the statement update the same status variables when used. Please see the documentation for [**SHOW REPLICA STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-replica-status) for a description of the statement.

#### 13.7.7.37 SHOW STATUS Statement

SHOW [GLOBAL | SESSION] STATUS

[LIKE '***pattern***' | WHERE ***expr***]

**[SHOW STATUS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "show-status" \o "13.7.7.37 SHOW STATUS Statement)** provides server status information (see [Section 5.1.10, “Server Status Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-status-variables)). This statement does not require any privilege. It requires only the ability to connect to the server.

Status variable information is also available from these sources:

Performance Schema tables. See [Section 27.12.15, “Performance Schema Status Variable Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-status-variable-tables).

The [**mysqladmin extended-status**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) command. See [Section 4.5.2, “mysqladmin — A MySQL Server Administration Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin).

For [**SHOW STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-status), a [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which variable names to match. A **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

[**SHOW STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-status) accepts an optional **GLOBAL** or **SESSION** variable scope modifier:

With a **GLOBAL** modifier, the statement displays the global status values. A global status variable may represent status for some aspect of the server itself (for example, **Aborted\_connects**), or the aggregated status over all connections to MySQL (for example, **Bytes\_received** and **Bytes\_sent**). If a variable has no global value, the session value is displayed.

With a **SESSION** modifier, the statement displays the status variable values for the current connection. If a variable has no session value, the global value is displayed. **LOCAL** is a synonym for **SESSION**.

If no modifier is present, the default is **SESSION**.

The scope for each status variable is listed at [Section 5.1.10, “Server Status Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-status-variables).

Each invocation of the [**SHOW STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-status) statement uses an internal temporary table and increments the global [**Created\_tmp\_tables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Created_tmp_tables) value.

Partial output is shown here. The list of names and values may differ for your server. The meaning of each variable is given in [Section 5.1.10, “Server Status Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-status-variables).

mysql> **SHOW STATUS;**

+--------------------------+------------+

| Variable\_name | Value |

+--------------------------+------------+

| Aborted\_clients | 0 |

| Aborted\_connects | 0 |

| Bytes\_received | 155372598 |

| Bytes\_sent | 1176560426 |

| Connections | 30023 |

| Created\_tmp\_disk\_tables | 0 |

| Created\_tmp\_tables | 8340 |

| Created\_tmp\_files | 60 |

...

| Open\_tables | 1 |

| Open\_files | 2 |

| Open\_streams | 0 |

| Opened\_tables | 44600 |

| Questions | 2026873 |

...

| Table\_locks\_immediate | 1920382 |

| Table\_locks\_waited | 0 |

| Threads\_cached | 0 |

| Threads\_created | 30022 |

| Threads\_connected | 1 |

| Threads\_running | 1 |

| Uptime | 80380 |

+--------------------------+------------+

With a [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, the statement displays only rows for those variables with names that match the pattern:

mysql> **SHOW STATUS LIKE 'Key%';**

+--------------------+----------+

| Variable\_name | Value |

+--------------------+----------+

| Key\_blocks\_used | 14955 |

| Key\_read\_requests | 96854827 |

| Key\_reads | 162040 |

| Key\_write\_requests | 7589728 |

| Key\_writes | 3813196 |

+--------------------+----------+

#### 13.7.7.38 SHOW TABLE STATUS Statement

SHOW TABLE STATUS

[{FROM | IN} ***db\_name***]

[LIKE '***pattern***' | WHERE ***expr***]

[**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) works likes [**SHOW TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-tables), but provides a lot of information about each non-**TEMPORARY** table. You can also get this list using the [**mysqlshow --status *db\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlshow) command. The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which table names to match. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

This statement also displays information about views.

[**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) output has these columns:

**Name**

The name of the table.

**Engine**

The storage engine for the table. See [Chapter 15, *The InnoDB Storage Engine*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html), and [Chapter 16, *Alternative Storage Engines*](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\storage-engines.html).

For partitioned tables, **Engine** shows the name of the storage engine used by all partitions.

**Version**

This column is unused. With the removal of .frm files in MySQL 8.0, this column now reports a hardcoded value of **10**, which is the last .frm file version used in MySQL 5.7.

**Row\_format**

The row-storage format (**Fixed**, **Dynamic**, **Compressed**, **Redundant**, **Compact**). For **MyISAM** tables, **Dynamic** corresponds to what [**myisamchk -dvv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) reports as **Packed**.

**Rows**

The number of rows. Some storage engines, such as **MyISAM**, store the exact count. For other storage engines, such as **InnoDB**, this value is an approximation, and may vary from the actual value by as much as 40% to 50%. In such cases, use **SELECT COUNT(\*)** to obtain an accurate count.

The **Rows** value is **NULL** for **INFORMATION\_SCHEMA** tables.

For [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, the row count is only a rough estimate used in SQL optimization. (This is also true if the [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) table is partitioned.)

**Avg\_row\_length**

The average row length.

**Data\_length**

For **MyISAM**, **Data\_length** is the length of the data file, in bytes.

For **InnoDB**, **Data\_length** is the approximate amount of space allocated for the clustered index, in bytes. Specifically, it is the clustered index size, in pages, multiplied by the **InnoDB** page size.

Refer to the notes at the end of this section for information regarding other storage engines.

**Max\_data\_length**

For **MyISAM**, **Max\_data\_length** is maximum length of the data file. This is the total number of bytes of data that can be stored in the table, given the data pointer size used.

Unused for **InnoDB**.

Refer to the notes at the end of this section for information regarding other storage engines.

**Index\_length**

For **MyISAM**, **Index\_length** is the length of the index file, in bytes.

For **InnoDB**, **Index\_length** is the approximate amount of space allocated for non-clustered indexes, in bytes. Specifically, it is the sum of non-clustered index sizes, in pages, multiplied by the **InnoDB** page size.

Refer to the notes at the end of this section for information regarding other storage engines.

**Data\_free**

The number of allocated but unused bytes.

**InnoDB** tables report the free space of the tablespace to which the table belongs. For a table located in the shared tablespace, this is the free space of the shared tablespace. If you are using multiple tablespaces and the table has its own tablespace, the free space is for only that table. Free space means the number of bytes in completely free extents minus a safety margin. Even if free space displays as 0, it may be possible to insert rows as long as new extents need not be allocated.

For NDB Cluster, **Data\_free** shows the space allocated on disk for, but not used by, a Disk Data table or fragment on disk. (In-memory data resource usage is reported by the **Data\_length** column.)

For partitioned tables, this value is only an estimate and may not be absolutely correct. A more accurate method of obtaining this information in such cases is to query the **INFORMATION\_SCHEMA** [**PARTITIONS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-partitions-table) table, as shown in this example:

SELECT SUM(DATA\_FREE)

FROM INFORMATION\_SCHEMA.PARTITIONS

WHERE TABLE\_SCHEMA = 'mydb'

AND TABLE\_NAME = 'mytable';

For more information, see [Section 26.3.21, “The INFORMATION\_SCHEMA PARTITIONS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-partitions-table).

**Auto\_increment**

The next **AUTO\_INCREMENT** value.

**Create\_time**

When the table was created.

**Update\_time**

When the data file was last updated. For some storage engines, this value is **NULL**. For example, **InnoDB** stores multiple tables in its [system tablespace](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_system_tablespace) and the data file timestamp does not apply. Even with [file-per-table](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_file_per_table) mode with each **InnoDB** table in a separate **.ibd** file, [change buffering](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffering) can delay the write to the data file, so the file modification time is different from the time of the last insert, update, or delete. For **MyISAM**, the data file timestamp is used; however, on Windows the timestamp is not updated by updates, so the value is inaccurate.

**Update\_time** displays a timestamp value for the last [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) performed on **InnoDB** tables that are not partitioned. For MVCC, the timestamp value reflects the [**COMMIT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) time, which is considered the last update time. Timestamps are not persisted when the server is restarted or when the table is evicted from the **InnoDB** data dictionary cache.

**Check\_time**

When the table was last checked. Not all storage engines update this time, in which case, the value is always **NULL**.

For partitioned [**InnoDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html) tables, **Check\_time** is always **NULL**.

**Collation**

The table default collation. The output does not explicitly list the table default character set, but the collation name begins with the character set name.

**Checksum**

The live checksum value, if any.

**Create\_options**

Extra options used with [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table).

**Create\_options** shows **partitioned** for a partitioned table.

Prior to MySQL 8.0.16, **Create\_options** shows the **ENCRYPTION** clause specified for tables created in file-per-table tablespaces. As of MySQL 8.0.16, it shows the encryption clause for file-per-table tablespaces if the table is encrypted or if the specified encryption differs from the schema encryption. The encryption clause is not shown for tables created in general tablespaces. To identify encrypted file-per-table and general tablespaces, query the [**INNODB\_TABLESPACES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-innodb-tablespaces-table) **ENCRYPTION** column.

When creating a table with [strict mode](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_strict_mode) disabled, the storage engine's default row format is used if the specified row format is not supported. The actual row format of the table is reported in the **Row\_format** column. **Create\_options** shows the row format that was specified in the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement.

When altering the storage engine of a table, table options that are not applicable to the new storage engine are retained in the table definition to enable reverting the table with its previously defined options to the original storage engine, if necessary. **Create\_options** may show retained options.

**Comment**

The comment used when creating the table (or information as to why MySQL could not access the table information).

##### Notes

For **InnoDB** tables, [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) does not give accurate statistics except for the physical size reserved by the table. The row count is only a rough estimate used in SQL optimization.

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, the output of this statement shows appropriate values for the **Avg\_row\_length** and **Data\_length** columns, with the exception that [**BLOB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-types.html#blob) columns are not taken into account.

For [**NDB**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\mysql-cluster.html) tables, **Data\_length** includes data stored in main memory only; the **Max\_data\_length** and **Data\_free** columns apply to Disk Data.

For NDB Cluster Disk Data tables, **Max\_data\_length** shows the space allocated for the disk part of a Disk Data table or fragment. (In-memory data resource usage is reported by the **Data\_length** column.)

For **MEMORY** tables, the **Data\_length**, **Max\_data\_length**, and **Index\_length** values approximate the actual amount of allocated memory. The allocation algorithm reserves memory in large amounts to reduce the number of allocation operations.

For views, most columns displayed by [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status) are 0 or **NULL** except that **Name** indicates the view name, **Create\_time** indicates the creation time, and **Comment** says **VIEW**.

Table information is also available from the **INFORMATION\_SCHEMA** [**TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table) table. See [Section 26.3.38, “The INFORMATION\_SCHEMA TABLES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table).

#### 13.7.7.39 SHOW TABLES Statement

SHOW [EXTENDED] [FULL] TABLES

[{FROM | IN} ***db\_name***]

[LIKE '***pattern***' | WHERE ***expr***]

[**SHOW TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-tables) lists the non-**TEMPORARY** tables in a given database. You can also get this list using the [**mysqlshow *db\_name***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlshow) command. The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which table names to match. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

Matching performed by the **LIKE** clause is dependent on the setting of the [**lower\_case\_table\_names**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_lower_case_table_names) system variable.

The optional **EXTENDED** modifier causes [**SHOW TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-tables) to list hidden tables created by failed [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statements. These temporary tables have names beginning with **#sql** and can be dropped using [**DROP TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-table).

This statement also lists any views in the database. The optional **FULL** modifier causes [**SHOW TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-tables) to display a second output column with values of **BASE TABLE** for a table, **VIEW** for a view, or **SYSTEM VIEW** for an **INFORMATION\_SCHEMA** table.

If you have no privileges for a base table or view, it does not show up in the output from [**SHOW TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-tables) or [**mysqlshow db\_name**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlshow).

Table information is also available from the **INFORMATION\_SCHEMA** [**TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table) table. See [Section 26.3.38, “The INFORMATION\_SCHEMA TABLES Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-tables-table).

#### 13.7.7.40 SHOW TRIGGERS Statement

SHOW TRIGGERS

[{FROM | IN} ***db\_name***]

[LIKE '***pattern***' | WHERE ***expr***]

[**SHOW TRIGGERS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-triggers) lists the triggers currently defined for tables in a database (the default database unless a **FROM** clause is given). This statement returns results only for databases and tables for which you have the [**TRIGGER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_trigger) privilege. The [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which table names (not trigger names) to match and causes the statement to display triggers for those tables. The **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

For the **ins\_sum** trigger defined in [Section 25.3, “Using Triggers”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#triggers), the output of [**SHOW TRIGGERS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-triggers) is as shown here:

mysql> **SHOW TRIGGERS LIKE 'acc%'\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Trigger: ins\_sum

Event: INSERT

Table: account

Statement: SET @sum = @sum + NEW.amount

Timing: BEFORE

Created: 2018-08-08 10:10:12.61

sql\_mode: ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,

NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,

ERROR\_FOR\_DIVISION\_BY\_ZERO,

NO\_ENGINE\_SUBSTITUTION

Definer: me@localhost

character\_set\_client: utf8mb4

collation\_connection: utf8mb4\_0900\_ai\_ci

Database Collation: utf8mb4\_0900\_ai\_ci

[**SHOW TRIGGERS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-triggers) output has these columns:

**Trigger**

The name of the trigger.

**Event**

The trigger event. This is the type of operation on the associated table for which the trigger activates. The value is **INSERT** (a row was inserted), **DELETE** (a row was deleted), or **UPDATE** (a row was modified).

**Table**

The table for which the trigger is defined.

**Statement**

The trigger body; that is, the statement executed when the trigger activates.

**Timing**

Whether the trigger activates before or after the triggering event. The value is **BEFORE** or **AFTER**.

**Created**

The date and time when the trigger was created. This is a **TIMESTAMP(2)** value (with a fractional part in hundredths of seconds) for triggers.

**sql\_mode**

The SQL mode in effect when the trigger was created, and under which the trigger executes. For the permitted values, see [Section 5.1.11, “Server SQL Modes”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sql-mode).

**Definer**

The account of the user who created the trigger, in **'*user\_name*'@'*host\_name*'** format.

**character\_set\_client**

The session value of the [**character\_set\_client**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_character_set_client) system variable when the trigger was created.

**collation\_connection**

The session value of the [**collation\_connection**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_collation_connection) system variable when the trigger was created.

**Database Collation**

The collation of the database with which the trigger is associated.

Trigger information is also available from the **INFORMATION\_SCHEMA** [**TRIGGERS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-triggers-table) table. See [Section 26.3.45, “The INFORMATION\_SCHEMA TRIGGERS Table”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-triggers-table).

#### 13.7.7.41 SHOW VARIABLES Statement

SHOW [GLOBAL | SESSION] VARIABLES

[LIKE '***pattern***' | WHERE ***expr***]

**[SHOW VARIABLES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "show-variables" \o "13.7.7.41 SHOW VARIABLES Statement)** shows the values of MySQL system variables (see [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables)). This statement does not require any privilege. It requires only the ability to connect to the server.

System variable information is also available from these sources:

Performance Schema tables. See [Section 27.12.14, “Performance Schema System Variable Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-system-variable-tables).

The [**mysqladmin variables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) command. See [Section 4.5.2, “mysqladmin — A MySQL Server Administration Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin).

For [**SHOW VARIABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables), a [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause, if present, indicates which variable names to match. A **WHERE** clause can be given to select rows using more general conditions, as discussed in [Section 26.8, “Extensions to SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#extended-show).

[**SHOW VARIABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables) accepts an optional **GLOBAL** or **SESSION** variable scope modifier:

With a **GLOBAL** modifier, the statement displays global system variable values. These are the values used to initialize the corresponding session variables for new connections to MySQL. If a variable has no global value, no value is displayed.

With a **SESSION** modifier, the statement displays the system variable values that are in effect for the current connection. If a variable has no session value, the global value is displayed. **LOCAL** is a synonym for **SESSION**.

If no modifier is present, the default is **SESSION**.

The scope for each system variable is listed at [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables).

[**SHOW VARIABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-variables) is subject to a version-dependent display-width limit. For variables with very long values that are not completely displayed, use [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) as a workaround. For example:

SELECT @@GLOBAL.innodb\_data\_file\_path;

Most system variables can be set at server startup (read-only variables such as [**version\_comment**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_version_comment) are exceptions). Many can be changed at runtime with the [**SET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) statement. See [Section 5.1.9, “Using System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#using-system-variables), and [Section 13.7.6.1, “SET Syntax for Variable Assignment”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable).

Partial output is shown here. The list of names and values may differ for your server. [Section 5.1.8, “Server System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-system-variables), describes the meaning of each variable, and [Section 5.1.1, “Configuring the Server”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-configuration), provides information about tuning them.

mysql> **SHOW VARIABLES;**

+--------------------------------------------+------------------------------+

| Variable\_name | Value |

+--------------------------------------------+------------------------------+

| activate\_all\_roles\_on\_login | OFF |

| auto\_generate\_certs | ON |

| auto\_increment\_increment | 1 |

| auto\_increment\_offset | 1 |

| autocommit | ON |

| automatic\_sp\_privileges | ON |

| avoid\_temporal\_upgrade | OFF |

| back\_log | 151 |

| basedir | /usr/ |

| big\_tables | OFF |

| bind\_address | \* |

| binlog\_cache\_size | 32768 |

| binlog\_checksum | CRC32 |

| binlog\_direct\_non\_transactional\_updates | OFF |

| binlog\_error\_action | ABORT\_SERVER |

| binlog\_expire\_logs\_seconds | 2592000 |

| binlog\_format | ROW |

| binlog\_group\_commit\_sync\_delay | 0 |

| binlog\_group\_commit\_sync\_no\_delay\_count | 0 |

| binlog\_gtid\_simple\_recovery | ON |

| binlog\_max\_flush\_queue\_time | 0 |

| binlog\_order\_commits | ON |

| binlog\_row\_image | FULL |

| binlog\_row\_metadata | MINIMAL |

| binlog\_row\_value\_options | |

| binlog\_rows\_query\_log\_events | OFF |

| binlog\_stmt\_cache\_size | 32768 |

| binlog\_transaction\_dependency\_history\_size | 25000 |

| binlog\_transaction\_dependency\_tracking | COMMIT\_ORDER |

| block\_encryption\_mode | aes-128-ecb |

| bulk\_insert\_buffer\_size | 8388608 |

...

| max\_allowed\_packet | 67108864 |

| max\_binlog\_cache\_size | 18446744073709547520 |

| max\_binlog\_size | 1073741824 |

| max\_binlog\_stmt\_cache\_size | 18446744073709547520 |

| max\_connect\_errors | 100 |

| max\_connections | 151 |

| max\_delayed\_threads | 20 |

| max\_digest\_length | 1024 |

| max\_error\_count | 1024 |

| max\_execution\_time | 0 |

| max\_heap\_table\_size | 16777216 |

| max\_insert\_delayed\_threads | 20 |

| max\_join\_size | 18446744073709551615 |

...

| thread\_handling | one-thread-per-connection |

| thread\_stack | 286720 |

| time\_zone | SYSTEM |

| timestamp | 1530906638.765316 |

| tls\_version | TLSv1,TLSv1.1,TLSv1.2 |

| tmp\_table\_size | 16777216 |

| tmpdir | /tmp |

| transaction\_alloc\_block\_size | 8192 |

| transaction\_allow\_batching | OFF |

| transaction\_isolation | REPEATABLE-READ |

| transaction\_prealloc\_size | 4096 |

| transaction\_read\_only | OFF |

| transaction\_write\_set\_extraction | XXHASH64 |

| unique\_checks | ON |

| updatable\_views\_with\_limit | YES |

| version | 8.0.12 |

| version\_comment | MySQL Community Server - GPL |

| version\_compile\_machine | x86\_64 |

| version\_compile\_os | Linux |

| version\_compile\_zlib | 1.2.11 |

| wait\_timeout | 28800 |

| warning\_count | 0 |

| windowing\_use\_high\_precision | ON |

+--------------------------------------------+------------------------------+

With a [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_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**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause as shown:

SHOW VARIABLES LIKE 'max\_join\_size';

SHOW SESSION VARIABLES LIKE 'max\_join\_size';

To get a list of variables whose name match a pattern, use the **%** wildcard character in a [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) clause:

SHOW VARIABLES LIKE '%size%';

SHOW GLOBAL VARIABLES LIKE '%size%';

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.

#### 13.7.7.42 SHOW WARNINGS Statement

SHOW WARNINGS [LIMIT [***offset***,] ***row\_count***]

SHOW COUNT(\*) WARNINGS

[**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) is a diagnostic statement that displays information about the conditions (errors, warnings, and notes) resulting from executing a statement in the current session. Warnings are generated for DML statements such as [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update), and [**LOAD DATA**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-data) as well as DDL statements such as [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) and [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table).

The **LIMIT** clause has the same syntax as for the [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement. See [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).

[**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) is also used following [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain), to display the extended information generated by [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain). See [Section 8.8.3, “Extended EXPLAIN Output Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#explain-extended).

[**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) displays information about the conditions resulting from execution of the most recent nondiagnostic statement in the current session. If the most recent statement resulted in an error during parsing, [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) shows the resulting conditions, regardless of statement type (diagnostic or nondiagnostic).

The [**SHOW COUNT(\*) WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) diagnostic statement displays the total number of errors, warnings, and notes. You can also retrieve this number from the [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) system variable:

SHOW COUNT(\*) WARNINGS;

SELECT @@warning\_count;

A difference in these statements is that the first is a diagnostic statement that does not clear the message list. The second, because it is a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement is considered nondiagnostic and does clear the message list.

A related diagnostic statement, [**SHOW ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors), shows only error conditions (it excludes warnings and notes), and [**SHOW COUNT(\*) ERRORS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) statement displays the total number of errors. See [Section 13.7.7.17, “SHOW ERRORS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-errors). [**GET DIAGNOSTICS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics) can be used to examine information for individual conditions. See [Section 13.6.7.3, “GET DIAGNOSTICS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#get-diagnostics).

Here is a simple example that shows data-conversion warnings for [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert). The example assumes that strict SQL mode is disabled. With strict mode enabled, the warnings would become errors and terminate the [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert).

mysql> **CREATE TABLE t1 (a TINYINT NOT NULL, b CHAR(4));**

Query OK, 0 rows affected (0.05 sec)

mysql> **INSERT INTO t1 VALUES(10,'mysql'), (NULL,'test'), (300,'xyz');**

Query OK, 3 rows affected, 3 warnings (0.00 sec)

Records: 3 Duplicates: 0 Warnings: 3

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Warning

Code: 1265

Message: Data truncated for column 'b' at row 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Warning

Code: 1048

Message: Column 'a' cannot be null

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Level: Warning

Code: 1264

Message: Out of range value for column 'a' at row 3

3 rows in set (0.00 sec)

The [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) system variable controls the maximum number of error, warning, and note messages for which the server stores information, and thus the number of messages that [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings) displays. To change the number of messages the server can store, change the value of [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count).

[**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) controls only how many messages are stored, not how many are counted. The value of [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) is not limited by [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count), even if the number of messages generated exceeds [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count). The following example demonstrates this. The [**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) statement produces three warning messages (strict SQL mode is disabled for the example to prevent an error from occuring after a single conversion issue). Only one message is stored and displayed because [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) has been set to 1, but all three are counted (as shown by the value of [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count)):

mysql> **SHOW VARIABLES LIKE 'max\_error\_count';**

+-----------------+-------+

| Variable\_name | Value |

+-----------------+-------+

| max\_error\_count | 1024 |

+-----------------+-------+

1 row in set (0.00 sec)

mysql> **SET max\_error\_count=1, sql\_mode = '';**

Query OK, 0 rows affected (0.00 sec)

mysql> **ALTER TABLE t1 MODIFY b CHAR;**

Query OK, 3 rows affected, 3 warnings (0.00 sec)

Records: 3 Duplicates: 0 Warnings: 3

mysql> **SHOW WARNINGS;**

+---------+------+----------------------------------------+

| Level | Code | Message |

+---------+------+----------------------------------------+

| Warning | 1263 | Data truncated for column 'b' at row 1 |

+---------+------+----------------------------------------+

1 row in set (0.00 sec)

mysql> **SELECT @@warning\_count;**

+-----------------+

| @@warning\_count |

+-----------------+

| 3 |

+-----------------+

1 row in set (0.01 sec)

To disable message storage, set [**max\_error\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_error_count) to 0. In this case, [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) still indicates how many warnings occurred, but messages are not stored and cannot be displayed.

The [**sql\_notes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_notes) system variable controls whether note messages increment [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) and whether the server stores them. By default, [**sql\_notes**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_sql_notes) is 1, but if set to 0, notes do not increment [**warning\_count**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_warning_count) and the server does not store them:

mysql> **SET sql\_notes = 1;**

mysql> **DROP TABLE IF EXISTS test.no\_such\_table;**

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS;**

+-------+------+------------------------------------+

| Level | Code | Message |

+-------+------+------------------------------------+

| Note | 1051 | Unknown table 'test.no\_such\_table' |

+-------+------+------------------------------------+

1 row in set (0.00 sec)

mysql> **SET sql\_notes = 0;**

mysql> **DROP TABLE IF EXISTS test.no\_such\_table;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SHOW WARNINGS;**

Empty set (0.00 sec)

The MySQL server sends to each client a count indicating the total number of errors, warnings, and notes resulting from the most recent statement executed by that client. From the C API, this value can be obtained by calling [**mysql\_warning\_count()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-warning-count.html). See [mysql\_warning\_count()](https://dev.mysql.com/doc/c-api/8.0/en/mysql-warning-count.html).

In the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client, you can enable and disable automatic warnings display using the **warnings** and **nowarning** commands, respectively, or their shortcuts, **\W** and **\w** (see [Section 4.5.1.2, “mysql Client Commands”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql-commands)). For example:

mysql> **\W**

Show warnings enabled.

mysql> **SELECT 1/0;**

+------+

| 1/0 |

+------+

| NULL |

+------+

1 row in set, 1 warning (0.03 sec)

Warning (Code 1365): Division by 0

mysql> **\w**

Show warnings disabled.

### 13.7.8 Other Administrative Statements

[13.7.8.1 BINLOG Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#binlog)

[13.7.8.2 CACHE INDEX Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cache-index)

[13.7.8.3 FLUSH Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush)

[13.7.8.4 KILL Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill)

[13.7.8.5 LOAD INDEX INTO CACHE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-index)

[13.7.8.6 RESET Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset)

[13.7.8.7 RESET PERSIST Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist)

[13.7.8.8 RESTART Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart)

[13.7.8.9 SHUTDOWN Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#shutdown)

#### 13.7.8.1 BINLOG Statement

BINLOG '***str***'

[**BINLOG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#binlog) is an internal-use statement. It is generated by the [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) program as the printable representation of certain events in binary log files. (See [Section 4.6.9, “mysqlbinlog — Utility for Processing Binary Log Files”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog).) The **'*str*'** value is a base 64-encoded string the that server decodes to determine the data change indicated by the corresponding event.

To execute [**BINLOG**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#binlog) statements when applying [**mysqlbinlog**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqlbinlog) output, a user account requires the [**BINLOG\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_binlog-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege), or the [**REPLICATION\_APPLIER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_replication-applier) privilege plus the appropriate privileges to execute each log event.

This statement can execute only format description events and row events.

#### 13.7.8.2 CACHE INDEX Statement

CACHE INDEX {

***tbl\_index\_list*** [, ***tbl\_index\_list***] ...

| ***tbl\_name*** PARTITION (***partition\_list***)

}

IN ***key\_cache\_name***

***tbl\_index\_list***:

***tbl\_name*** [{INDEX|KEY} (***index\_name***[, ***index\_name***] ...)]

***partition\_list***: {

***partition\_name***[, ***partition\_name***] ...

| ALL

}

The [**CACHE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cache-index) statement assigns table indexes to a specific key cache. It applies only to **MyISAM** tables, including partitioned **MyISAM** tables. After the indexes have been assigned, they can be preloaded into the cache if desired with [**LOAD INDEX INTO CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-index).

The following statement assigns indexes from the tables **t1**, **t2**, and **t3** to the key cache named **hot\_cache**:

mysql> **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 |

+---------+--------------------+----------+----------+

The syntax of [**CACHE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cache-index) enables you to specify that only particular indexes from a table should be assigned to the cache. However, the implementation assigns all the table's indexes to the cache, so there is no reason to specify anything other than the table name.

The key cache referred to in a [**CACHE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cache-index) statement can be created by setting its size with a parameter setting statement or in the server parameter settings. For example:

SET GLOBAL keycache1.key\_buffer\_size=128\*1024;

Key cache parameters are accessed as members of a structured system variable. See [Section 5.1.9.5, “Structured System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#structured-system-variables).

A key cache must exist before you assign indexes to it, or an error occurs:

mysql> **CACHE INDEX t1 IN non\_existent\_cache;**

ERROR 1284 (HY000): Unknown key cache 'non\_existent\_cache'

By default, table indexes are assigned to the main (default) key cache created at the server startup. When a key cache is destroyed, all indexes assigned to it are reassigned to the default key cache.

Index assignment affects the server globally: If one client assigns an index to a given cache, this cache is used for all queries involving the index, no matter which client issues the queries.

[**CACHE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cache-index) is supported for partitioned **MyISAM** tables. You can assign one or more indexes for one, several, or all partitions to a given key cache. For example, you can do the following:

CREATE TABLE pt (c1 INT, c2 VARCHAR(50), INDEX i(c1))

ENGINE=MyISAM

PARTITION BY HASH(c1)

PARTITIONS 4;

SET GLOBAL kc\_fast.key\_buffer\_size = 128 \* 1024;

SET GLOBAL kc\_slow.key\_buffer\_size = 128 \* 1024;

CACHE INDEX pt PARTITION (p0) IN kc\_fast;

CACHE INDEX pt PARTITION (p1, p3) IN kc\_slow;

The previous set of statements performs the following actions:

Creates a partitioned table with 4 partitions; these partitions are automatically named **p0**, ..., **p3**; this table has an index named **i** on column **c1**.

Creates 2 key caches named **kc\_fast** and **kc\_slow**

Assigns the index for partition **p0** to the **kc\_fast** key cache and the index for partitions **p1** and **p3** to the **kc\_slow** key cache; the index for the remaining partition (**p2**) uses the server's default key cache.

If you wish instead to assign the indexes for all partitions in table **pt** to a single key cache named **kc\_all**, you can use either of the following two statements:

CACHE INDEX pt PARTITION (ALL) IN kc\_all;

CACHE INDEX pt IN kc\_all;

The two statements just shown are equivalent, and issuing either one has exactly the same effect. In other words, if you wish to assign indexes for all partitions of a partitioned table to the same key cache, the **PARTITION (ALL)** clause is optional.

When assigning indexes for multiple partitions to a key cache, the partitions need not be contiguous, and you need not list their names in any particular order. Indexes for any partitions not explicitly assigned to a key cache automatically use the server default key cache.

Index preloading is also supported for partitioned **MyISAM** tables. For more information, see [Section 13.7.8.5, “LOAD INDEX INTO CACHE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-index).

#### 13.7.8.3 FLUSH Statement

FLUSH [NO\_WRITE\_TO\_BINLOG | LOCAL] {

***flush\_option*** [, ***flush\_option***] ...

| ***tables\_option***

}

***flush\_option***: {

BINARY LOGS

| ENGINE LOGS

| ERROR LOGS

| GENERAL LOGS

| HOSTS

| LOGS

| PRIVILEGES

| OPTIMIZER\_COSTS

| RELAY LOGS [FOR CHANNEL ***channel***]

| SLOW LOGS

| STATUS

| USER\_RESOURCES

}

***tables\_option***: {

TABLES

| TABLES ***tbl\_name*** [, ***tbl\_name***] ...

| TABLES WITH READ LOCK

| TABLES ***tbl\_name*** [, ***tbl\_name***] ... WITH READ LOCK

| TABLES ***tbl\_name*** [, ***tbl\_name***] ... FOR EXPORT

}

The [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statement has several variant forms that clear or reload various internal caches, flush tables, or acquire locks. Each [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) operation requires the privileges indicated in its description.

**Note**

It is not possible to issue [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statements within stored functions or triggers. However, you may use [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) in stored procedures, so long as these are not called from stored functions or triggers. See [Section 25.8, “Restrictions on Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\stored-objects.html#stored-program-restrictions).

By default, the server writes [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statements to the binary log so that they replicate to replicas. To suppress logging, specify the optional **NO\_WRITE\_TO\_BINLOG** keyword or its alias **LOCAL**.

**Note**

[**FLUSH LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-logs), [**FLUSH BINARY LOGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-binary-logs), [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock) (with or without a table list), and [**FLUSH TABLES *tbl\_name* ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) are not written to the binary log in any case because they would cause problems if replicated to a replica.

The [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statement causes an implicit commit. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

The [**mysqladmin**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) utility provides a command-line interface to some flush operations, using commands such as **flush-hosts**, **flush-logs**, **flush-privileges**, **flush-status**, and **flush-tables**. See [Section 4.5.2, “mysqladmin — A MySQL Server Administration Program”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin).

Sending a **SIGHUP** or **SIGUSR1** signal to the server causes several flush operations to occur that are similar to various forms of the [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statement. Signals can be sent by the **root** system account or the system account that owns the server process. This enables the flush operations to be performed without having to connect to the server, which requires a MySQL account that has privileges sufficient for those operations. See [Section 4.10, “Unix Signal Handling in MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#unix-signal-response).

The [**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset) statement is similar to [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush). See [Section 13.7.8.6, “RESET Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset), for information about using [**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset) with replication.

The following list describes the permitted [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statement ***flush\_option*** values. For descriptions of the permitted ***tables\_option*** values, see [FLUSH TABLES Syntax](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-variants).

**[FLUSH BINARY LOGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-binary-logs)**

Closes and reopens any binary log file to which the server is writing. If binary logging is enabled, the sequence number of the binary log file is incremented by one relative to the previous file.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

**[FLUSH ENGINE LOGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-engine-logs)**

Closes and reopens any flushable logs for installed storage engines. This causes **InnoDB** to flush its logs to disk.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

**[FLUSH ERROR LOGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-error-logs)**

Closes and reopens any error log file to which the server is writing.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

**[FLUSH GENERAL LOGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-general-logs)**

Closes and reopens any general query log file to which the server is writing.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

This operation has no effect on tables used for the general query log (see [Section 5.4.1, “Selecting General Query Log and Slow Query Log Output Destinations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#log-destinations)).

**[FLUSH HOSTS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-hosts)**

Empties the host cache and the Performance Schema [**host\_cache**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-host-cache-table) table that exposes the cache contents, and unblocks any blocked hosts.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

For information about why host cache flushing might be advisable or desirable, see [Section 5.1.12.3, “DNS Lookups and the Host Cache”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#host-cache).

**Note**

[**FLUSH HOSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-hosts) is deprecated as of MySQL 8.0.23; expect it to be removed in a future MySQL release. Instead, truncate the Performance Schema [**host\_cache**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-host-cache-table) table:

TRUNCATE TABLE performance\_schema.host\_cache;

The [**TRUNCATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#truncate-table) operation requires the [**DROP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_drop) privilege for the table rather than the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

**[FLUSH LOGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-logs)**

Closes and reopens any log file to which the server is writing.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

The effect of this operation is equivalent to the combined effects of these operations:

FLUSH BINARY LOGS

FLUSH ENGINE LOGS

FLUSH ERROR LOGS

FLUSH GENERAL LOGS

FLUSH RELAY LOGS

FLUSH SLOW LOGS

**[FLUSH OPTIMIZER\_COSTS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-optimizer-costs)**

Re-reads the cost model tables so that the optimizer starts using the current cost estimates stored in them.

This operation requires the [**FLUSH\_OPTIMIZER\_COSTS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-optimizer-costs) or [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

The server writes a warning to the error log for any unrecognized cost model table entries. For information about these tables, see [Section 8.9.5, “The Optimizer Cost Model”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#cost-model). This operation affects only sessions that begin subsequent to the flush. Existing sessions continue to use the cost estimates that were current when they began.

**[FLUSH PRIVILEGES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-privileges)**

Re-reads the privileges from the grant tables in the **mysql** system schema. As part of this operation, the server reads the **global\_grants** table containing dynamic privilege assignments and registers any unregistered privileges found there.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

If the [--skip-grant-tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-grant-tables) option was specified at server startup to disable the MySQL privilege system, [**FLUSH PRIVILEGES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-privileges) provides a way to enable the privilege system at runtime.

Resets failed-login tracking (or enables it if the server was started with [--skip-grant-tables](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_skip-grant-tables)) and unlocks any temporarily locked accounts. See [Section 6.2.15, “Password Management”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#password-management).

Frees memory cached by the server as a result of [**GRANT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#grant), [**CREATE USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-user), [**CREATE SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-server), and [**INSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#install-plugin) statements. This memory is not released by the corresponding [**REVOKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#revoke), [**DROP USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-user), [**DROP SERVER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#drop-server), and [**UNINSTALL PLUGIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#uninstall-plugin) statements, so for a server that executes many instances of the statements that cause caching, there is an increase in cached memory use unless it is freed with [**FLUSH PRIVILEGES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-privileges).

Clears the in-memory cache used by the **caching\_sha2\_password** authentication plugin. See [Cache Operation for SHA-2 Pluggable Authentication](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#caching-sha2-pluggable-authentication-cache-operation).

**[FLUSH RELAY LOGS [FOR CHANNEL](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-relay-logs)*[channel](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-relay-logs)*[]](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-relay-logs)**

Closes and reopens any relay log file to which the server is writing. If relay logging is enabled, the sequence number of the relay log file is incremented by one relative to the previous file.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

The **FOR CHANNEL *channel*** clause enables you to name which replication channel the operation applies to. Execute [**FLUSH RELAY LOGS FOR CHANNEL *channel***](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-relay-logs) to flush the relay log for a specific replication channel. If no channel is named and no extra replication channels exist, the operation applies to the default channel. If no channel is named and multiple replication channels exist, the operation applies to all replication channels. For more information, see [Section 17.2.2, “Replication Channels”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\replication.html#replication-channels).

**[FLUSH SLOW LOGS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-slow-logs)**

Closes and reopens any slow query log file to which the server is writing.

This operation requires the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

This operation has no effect on tables used for the slow query log (see [Section 5.4.1, “Selecting General Query Log and Slow Query Log Output Destinations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#log-destinations)).

**[FLUSH STATUS](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-status)**

Flushes status indicators.

This operation requires the [**FLUSH\_STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-status) or [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

This operation adds the session status from all active sessions to the global status variables, resets the status of all active sessions, and resets account, host, and user status values aggregated from disconnected sessions. See [Section 27.12.15, “Performance Schema Status Variable Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-status-variable-tables). This information may be of use when debugging a query. See [Section 1.6, “How to Report Bugs or Problems”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\introduction.html#bug-reports).

**[FLUSH USER\_RESOURCES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-user-resources)**

Resets all per-hour user resource indicators to zero.

This operation requires the [**FLUSH\_USER\_RESOURCES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-user-resources) or [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

Resetting resource indicators enables clients that have reached their hourly connection, query, or update limits to resume activity immediately. [**FLUSH USER\_RESOURCES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-user-resources) does not apply to the limit on maximum simultaneous connections that is controlled by the [**max\_user\_connections**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_max_user_connections) system variable. See [Section 6.2.20, “Setting Account Resource Limits”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#user-resources).

##### FLUSH TABLES Syntax

[**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) flushes tables, and, depending on the variant used, acquires locks. Any **TABLES** variant used in a [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statement must be the only option used. [**FLUSH TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) is a synonym for [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables).

**Note**

The descriptions here that indicate tables are flushed by closing them apply differently for **InnoDB**, which flushes table contents to disk but leaves them open. This still permits table files to be copied while the tables are open, as long as other activity does not modify them.

**[FLUSH TABLES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables)**

Closes all open tables, forces all tables in use to be closed, and flushes the prepared statement cache.

This operation requires the [**FLUSH\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-tables) or [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

For information about prepared statement caching, see [Section 8.10.3, “Caching of Prepared Statements and Stored Programs”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#statement-caching).

[**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) is not permitted when there is an active [**LOCK TABLES ... READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables). To flush and lock tables, use [**FLUSH TABLES *tbl\_name* ... WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock-with-list) instead.

**[FLUSH TABLES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-list)*[tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-list)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-list)*[tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-list)*[] ...](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-list)**

With a list of one or more comma-separated table names, this operation is like [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) with no names except that the server flushes only the named tables. If a named table does not exist, no error occurs.

This operation requires the [**FLUSH\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-tables) or [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

**[FLUSH TABLES WITH READ LOCK](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-read-lock)**

Closes all open tables and locks all tables for all databases with a global read lock.

This operation requires the [**FLUSH\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-tables) or [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege.

This operation is a very convenient way to get backups if you have a file system such as Veritas or ZFS that can take snapshots in time. Use [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to release the lock.

[**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock) acquires a global read lock rather than table locks, so it is not subject to the same behavior as [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) and [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) with respect to table locking and implicit commits:

[**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) implicitly commits any active transaction only if any tables currently have been locked with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables). The commit does not occur for [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) following [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock) because the latter statement does not acquire table locks.

Beginning a transaction causes table locks acquired with [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to be released, as though you had executed [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables). Beginning a transaction does not release a global read lock acquired with [**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock).

[**FLUSH TABLES WITH READ LOCK**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-with-read-lock) does not prevent the server from inserting rows into the log tables (see [Section 5.4.1, “Selecting General Query Log and Slow Query Log Output Destinations”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#log-destinations)).

**[FLUSH TABLES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-read-lock-with-list)*[tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-read-lock-with-list)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-read-lock-with-list)*[tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-read-lock-with-list)*[] ... WITH READ LOCK](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-with-read-lock-with-list)**

Flushes and acquires read locks for the named tables.

This operation requires the [**FLUSH\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-tables) or [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege. Because it acquires table locks, it also requires the [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_lock-tables) privilege for each table.

The operation first acquires exclusive metadata locks for the tables, so it waits for transactions that have those tables open to complete. Then the operation flushes the tables from the table cache, reopens the tables, acquires table locks (like [**LOCK TABLES ... READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables)), and downgrades the metadata locks from exclusive to shared. After the operation acquires locks and downgrades the metadata locks, other sessions can read but not modify the tables.

This operation applies only to existing base (non-**TEMPORARY)**tables. If a name refers to a base table, that table is used. If it refers to a **TEMPORARY** table, it is ignored. If a name applies to a view, an [**ER\_WRONG\_OBJECT**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_object) error occurs. Otherwise, an [**ER\_NO\_SUCH\_TABLE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_no_such_table) error occurs.

Use [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to release the locks, [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to release the locks and acquire other locks, or [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) to release the locks and begin a new transaction.

This [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) variant enables tables to be flushed and locked in a single operation. It provides a workaround for the restriction that [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) is not permitted when there is an active [**LOCK TABLES ... READ**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables).

This operation does not perform an implicit [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables), so an error results if you perform the operation while there is any active [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) or use it a second time without first releasing the locks acquired.

If a flushed table was opened with [**HANDLER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#handler), the handler is implicitly flushed and loses its position.

**[FLUSH TABLES](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-for-export-with-list)*[tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-for-export-with-list)*[[,](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-for-export-with-list)*[tbl\_name](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-for-export-with-list)*[] ... FOR EXPORT](file:///E:\\backup\\%E4%B8%8B%E8%BD%BD\\refman-8.0-en.html-chapter\\refman-8.0-en.html-chapter\\sql-statements.html" \l "flush-tables-for-export-with-list)**

This [**FLUSH TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables) variant applies to **InnoDB** tables. It ensures that changes to the named tables have been flushed to disk so that binary table copies can be made while the server is running.

This operation requires the [**FLUSH\_TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_flush-tables) or [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege. Because it acquires locks on tables in preparation for exporting them, it also requires the [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_lock-tables) and [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_select) privileges for each table.

The operation works like this:

It acquires shared metadata locks for the named tables. The operation blocks as long as other sessions have active transactions that have modified those tables or hold table locks for them. When the locks have been acquired, the operation blocks transactions that attempt to update the tables, while permitting read-only operations to continue.

It checks whether all storage engines for the tables support **FOR EXPORT**. If any do not, an [**ER\_ILLEGAL\_HA**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_illegal_ha) error occurs and the operation fails.

The operation notifies the storage engine for each table to make the table ready for export. The storage engine must ensure that any pending changes are written to disk.

The operation puts the session in lock-tables mode so that the metadata locks acquired earlier are not released when the **FOR EXPORT** operation completes.

This operation applies only to existing base (non-**TEMPORARY**) tables. If a name refers to a base table, that table is used. If it refers to a **TEMPORARY** table, it is ignored. If a name applies to a view, an [**ER\_WRONG\_OBJECT**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_wrong_object) error occurs. Otherwise, an [**ER\_NO\_SUCH\_TABLE**](https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html#error_er_no_such_table) error occurs.

**InnoDB** supports **FOR EXPORT** for tables that have their own [.ibd file](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_ibd_file) file (that is, tables created with the [**innodb\_file\_per\_table**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#sysvar_innodb_file_per_table) setting enabled). **InnoDB** ensures when notified by the **FOR EXPORT** operation that any changes have been flushed to disk. This permits a binary copy of table contents to be made while the **FOR EXPORT** operation is in effect because the .ibd file is transaction consistent and can be copied while the server is running. **FOR EXPORT** does not apply to **InnoDB** system tablespace files, or to **InnoDB** tables that have **FULLTEXT** indexes.

[**FLUSH TABLES ...FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) is supported for partitioned **InnoDB** tables.

When notified by **FOR EXPORT**, **InnoDB** writes to disk certain kinds of data that is normally held in memory or in separate disk buffers outside the tablespace files. For each table, **InnoDB** also produces a file named ***table\_name***.cfg in the same database directory as the table. The .cfg file contains metadata needed to reimport the tablespace files later, into the same or different server.

When the **FOR EXPORT** operation completes, **InnoDB** has flushed all [dirty pages](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_dirty_page) to the table data files. Any [change buffer](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\glossary.html#glos_change_buffer) entries are merged prior to flushing. At this point, the tables are locked and quiescent: The tables are in a transactionally consistent state on disk and you can copy the .ibd tablespace files along with the corresponding .cfg files to get a consistent snapshot of those tables.

For the procedure to reimport the copied table data into a MySQL instance, see [Section 15.6.1.3, “Importing InnoDB Tables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\innodb-storage-engine.html#innodb-table-import).

After you are done with the tables, use [**UNLOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to release the locks, [**LOCK TABLES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#lock-tables) to release the locks and acquire other locks, or [**START TRANSACTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#commit) to release the locks and begin a new transaction.

While any of these statements is in effect within the session, attempts to use [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) produce an error:

FLUSH TABLES ... WITH READ LOCK

FLUSH TABLES ... FOR EXPORT

LOCK TABLES ... READ

LOCK TABLES ... WRITE

While [**FLUSH TABLES ... FOR EXPORT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush-tables-for-export-with-list) is in effect within the session, attempts to use any of these statements produce an error:

FLUSH TABLES WITH READ LOCK

FLUSH TABLES ... WITH READ LOCK

FLUSH TABLES ... FOR EXPORT

#### 13.7.8.4 KILL Statement

KILL [CONNECTION | QUERY] ***processlist\_id***

Each connection to [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) runs in a separate thread. You can kill a thread with the **KILL *processlist\_id*** statement.

Thread processlist identifiers can be determined from the **ID** column of the **INFORMATION\_SCHEMA** [**PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\information-schema.html#information-schema-processlist-table) table, the **Id** column of [**SHOW PROCESSLIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-processlist) output, and the **PROCESSLIST\_ID** column of the Performance Schema [**threads**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-threads-table) table. The value for the current thread is returned by the [**CONNECTION\_ID()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_connection-id) function.

[**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) permits an optional **CONNECTION** or **QUERY** modifier:

[**KILL CONNECTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) is the same as [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) with no modifier: It terminates the connection associated with the given ***processlist\_id***, after terminating any statement the connection is executing.

[**KILL QUERY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) terminates the statement the connection is currently executing, but leaves the connection itself intact.

The ability to see which threads are available to be killed depends on the [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process) privilege:

Without [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process), you can see only your own threads.

With [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process), you can see all threads.

The ability to kill threads and statements depends on the [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) privilege and the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege:

Without [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) or [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super), you can kill only your own threads and statements.

With [**CONNECTION\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_connection-admin) or [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super), you can kill all threads and statements, except that to affect a thread or statement that is executing with the [**SYSTEM\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-user) privilege, your own session must additionally have the [**SYSTEM\_USER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-user) privilege.

You can also use the [**mysqladmin processlist**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) and [**mysqladmin kill**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) commands to examine and kill threads.

When you use [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill), a thread-specific kill flag is set for the thread. In most cases, it might take some time for the thread to die because the kill flag is checked only at specific intervals:

During [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) operations, for **ORDER BY** and **GROUP BY** loops, the flag is checked after reading a block of rows. If the kill flag is set, the statement is aborted.

[**ALTER TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#alter-table) operations that make a table copy check the kill flag periodically for each few copied rows read from the original table. If the kill flag was set, the statement is aborted and the temporary table is deleted.

The [**KILL**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) statement returns without waiting for confirmation, but the kill flag check aborts the operation within a reasonably small amount of time. Aborting the operation to perform any necessary cleanup also takes some time.

During [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) or [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) operations, the kill flag is checked after each block read and after each updated or deleted row. If the kill flag is set, the statement is aborted. If you are not using transactions, the changes are not rolled back.

[**GET\_LOCK()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_get-lock) aborts and returns **NULL**.

If the thread is in the table lock handler (state: **Locked**), the table lock is quickly aborted.

If the thread is waiting for free disk space in a write call, the write is aborted with a “disk full” error message.

[**EXPLAIN ANALYZE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain-analyze) aborts and prints the first row of output. This works in MySQL 8.0.20 and later.

**Warning**

Killing a [**REPAIR TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#repair-table) or [**OPTIMIZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#optimize-table) operation on a **MyISAM** table results in a table that is corrupted and unusable. Any reads or writes to such a table fail until you optimize or repair it again (without interruption).

#### 13.7.8.5 LOAD INDEX INTO CACHE Statement

LOAD INDEX INTO CACHE

***tbl\_index\_list*** [, ***tbl\_index\_list***] ...

***tbl\_index\_list***:

***tbl\_name***

[PARTITION (***partition\_list***)]

[{INDEX|KEY} (***index\_name***[, ***index\_name***] ...)]

[IGNORE LEAVES]

***partition\_list***: {

***partition\_name***[, ***partition\_name***] ...

| ALL

}

The [**LOAD INDEX INTO CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-index) statement preloads a table index into the key cache to which it has been assigned by an explicit [**CACHE INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#cache-index) statement, or into the default key cache otherwise.

[**LOAD INDEX INTO CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-index) applies only to **MyISAM** tables, including partitioned **MyISAM** tables. In addition, indexes on partitioned tables can be preloaded for one, several, or all partitions.

The **IGNORE LEAVES** modifier causes only blocks for the nonleaf nodes of the index to be preloaded.

**IGNORE LEAVES** is also supported for partitioned **MyISAM** tables.

The following statement preloads nodes (index blocks) of indexes for the tables **t1** and **t2**:

mysql> **LOAD INDEX INTO CACHE t1, t2 IGNORE LEAVES;**

+---------+--------------+----------+----------+

| Table | Op | Msg\_type | Msg\_text |

+---------+--------------+----------+----------+

| test.t1 | preload\_keys | status | OK |

| test.t2 | preload\_keys | status | OK |

+---------+--------------+----------+----------+

This statement preloads all index blocks from **t1**. It preloads only blocks for the nonleaf nodes from **t2**.

The syntax of [**LOAD INDEX INTO CACHE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-index) enables you to specify that only particular indexes from a table should be preloaded. However, the implementation preloads all the table's indexes into the cache, so there is no reason to specify anything other than the table name.

It is possible to preload indexes on specific partitions of partitioned **MyISAM** tables. For example, of the following 2 statements, the first preloads indexes for partition **p0** of a partitioned table **pt**, while the second preloads the indexes for partitions **p1** and **p3** of the same table:

LOAD INDEX INTO CACHE pt PARTITION (p0);

LOAD INDEX INTO CACHE pt PARTITION (p1, p3);

To preload the indexes for all partitions in table **pt**, you can use either of the following two statements:

LOAD INDEX INTO CACHE pt PARTITION (ALL);

LOAD INDEX INTO CACHE pt;

The two statements just shown are equivalent, and issuing either one has exactly the same effect. In other words, if you wish to preload indexes for all partitions of a partitioned table, the **PARTITION (ALL)** clause is optional.

When preloading indexes for multiple partitions, the partitions need not be contiguous, and you need not list their names in any particular order.

[**LOAD INDEX INTO CACHE ... IGNORE LEAVES**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#load-index) fails unless all indexes in a table have the same block size. To determine index block sizes for a table, use [**myisamchk -dv**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#myisamchk) and check the **Blocksize** column.

#### 13.7.8.6 RESET Statement

RESET ***reset\_option*** [, ***reset\_option***] ...

***reset\_option***: {

MASTER

| REPLICA

| SLAVE

}

The [**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset) statement is used to clear the state of various server operations. You must have the [**RELOAD**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_reload) privilege to execute [**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset).

For information about the [**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) statement that removes persisted global system variables, see [Section 13.7.8.7, “RESET PERSIST Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist).

[**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset) acts as a stronger version of the [**FLUSH**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush) statement. See [Section 13.7.8.3, “FLUSH Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#flush).

The [**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset) statement causes an implicit commit. See [Section 13.3.3, “Statements That Cause an Implicit Commit”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#implicit-commit).

The following list describes the permitted [**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset) statement ***reset\_option*** values:

[**RESET MASTER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-master)

Deletes all binary logs listed in the index file, resets the binary log index file to be empty, and creates a new binary log file.

[**RESET REPLICA | SLAVE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-replica)

Makes the replica forget its replication position in the source binary logs. Also resets the relay log by deleting any existing relay log files and beginning a new one. Use **RESET REPLICA** in place of **RESET SLAVE** from MySQL 8.0.22.

#### 13.7.8.7 RESET PERSIST Statement

RESET PERSIST [[IF EXISTS] ***system\_var\_name***]

[**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) removes persisted global system variable settings from the mysqld-auto.cnf option file in the data directory. Removing a persisted system variable causes the variable no longer to be initialized from mysqld-auto.cnf at server startup. For more information about persisting system variables and the mysqld-auto.cnf file, see [Section 5.1.9.3, “Persisted System Variables”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#persisted-system-variables).

The privileges required for [**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) depend on the type of system variable to be removed:

For dynamic system variables, this statement requires the [**SYSTEM\_VARIABLES\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-variables-admin) privilege (or the deprecated [**SUPER**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_super) privilege).

For read-only system variables, this statement requires the [**SYSTEM\_VARIABLES\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_system-variables-admin) and [**PERSIST\_RO\_VARIABLES\_ADMIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_persist-ro-variables-admin) privileges.

See [Section 5.1.9.1, “System Variable Privileges”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#system-variable-privileges).

Depending on whether the variable name and **IF EXISTS** clauses are present, the [**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) statement has these forms:

To remove all persisted variables from mysqld-auto.cnf, use [**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) without naming any system variable:

RESET PERSIST;

You must have privileges for removing both dynamic and read-only system variables if mysqld-auto.cnf contains both kinds of variables.

To remove a specific persisted variable from mysqld-auto.cnf, name it in the statement:

RESET PERSIST ***system\_var\_name***;

This includes plugin system variables, even if the plugin is not currently installed. If the variable is not present in the file, an error occurs.

To remove a specific persisted variable from mysqld-auto.cnf, but produce a warning rather than an error if the variable is not present in the file, add an **IF EXISTS** clause to the previous syntax:

RESET PERSIST IF EXISTS ***system\_var\_name***;

[**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) is not affected by the value of the [**persisted\_globals\_load**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#sysvar_persisted_globals_load) system variable.

[**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) affects the contents of the Performance Schema [**persisted\_variables**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-persisted-variables-table) table because the table contents correspond to the contents of the mysqld-auto.cnf file. On the other hand, because [**RESET PERSIST**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset-persist) does not change variable values, it has no effect on the contents of the Performance Schema [**variables\_info**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\performance-schema.html#performance-schema-variables-info-table) table until the server is restarted.

For information about [**RESET**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset) statement variants that clear the state of other server operations, see [Section 13.7.8.6, “RESET Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#reset).

#### 13.7.8.8 RESTART Statement

RESTART

This statement stops and restarts the MySQL server. It requires the [**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_shutdown) privilege.

One use for [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) is when it is not possible or convenient to gain command-line access to the MySQL server on the server host to restart it. For example, [**SET PERSIST\_ONLY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#set-variable) can be used at runtime to make configuration changes to system variables that can be set only at server startup, but the server must still be restarted for those changes to take effect. The [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) statement provides a way to do so from within client sessions, without requiring command-line access on the server host.

**Note**

After executing a [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) statement, the client can expect the current connection to be lost. If auto-reconnect is enabled, the connection is reestablished after the server restarts. Otherwise, the connection must be reestablished manually.

A successful [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) operation requires [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) to be running in an environment that has a monitoring process available to detect a server shutdown performed for restart purposes:

In the presence of a monitoring process, [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) causes [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) to terminate such that the monitoring process can determine that it should start a new [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) instance.

If no monitoring process is present, [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) fails with an error.

These platforms provide the necessary monitoring support for the [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) statement:

Windows, when [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) is started as a Windows service or standalone. ([**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) forks, and one process acts as a monitor to the other, which acts as the server.)

Unix and Unix-like systems that use systemd or [**mysqld\_safe**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld-safe) to manage [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld).

To configure a monitoring environment such that [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) enables the [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) statement:

Set the **MYSQLD\_PARENT\_PID** environment variable to the value of the process ID of the process that starts [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld), before starting [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld).

When [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) performs a shutdown due to use of the [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) statement, it returns exit code 16.

When the monitoring process detects an exit code of 16, it starts [**mysqld**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqld) again. Otherwise, it exits.

Here is a minimal example as implemented in the **bash** shell:

#!/bin/bash

export MYSQLD\_PARENT\_PID=$$

export MYSQLD\_RESTART\_EXIT=16

while true ; do

bin/mysqld ***mysqld options here***

if [ $? -ne $MYSQLD\_RESTART\_EXIT ]; then

break

fi

done

On Windows, the forking used to implement [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) makes determining the server process to attach to for debugging more difficult. To alleviate this, starting the server with [--gdb](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_gdb) suppresses forking, in addition to its other actions done to set up a debugging environment. In non-debug settings, [--no-monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_no-monitor) may be used for the sole purpose of suppressing forking the monitor process. For a server started with either [--gdb](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_gdb) or [--no-monitor](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#option_mysqld_no-monitor), executing [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) causes the server to simply exit without restarting.

The [**Com\_restart**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx) status variable tracks the number of [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) statements. Because status variables are initialized for each server startup and do not persist across restarts, **Com\_restart** normally has a value of zero, but can be nonzero if [**RESTART**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#restart) statements were executed but failed.

#### 13.7.8.9 SHUTDOWN Statement

SHUTDOWN

This statement stops the MySQL server. It requires the [**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_shutdown) privilege.

[**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#shutdown) provides an SQL-level interface to the same functionality available using the [**mysqladmin shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysqladmin) command or the [**mysql\_shutdown()**](https://dev.mysql.com/doc/c-api/8.0/en/mysql-shutdown.html) C API function.

The [**Com\_shutdown**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#statvar_Com_xxx) status variable tracks the number of [**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#shutdown) statements. Because status variables are initialized for each server startup and do not persist across restarts, **Com\_shutdown** normally has a value of zero, but can be nonzero if [**SHUTDOWN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#shutdown) statements were executed but failed.

Another way to stop the server is to send it a **SIGTERM** signal, which can be done by **root** or the account that owns the server process. **SIGTERM** enables server shutdown to be performed without having to connect to the server. See [Section 4.10, “Unix Signal Handling in MySQL”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#unix-signal-response).

## 13.8 Utility Statements

[13.8.1 DESCRIBE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe)

[13.8.2 EXPLAIN Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain)

[13.8.3 HELP Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#help)

[13.8.4 USE Statement](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use)

### 13.8.1 DESCRIBE Statement

The [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) and [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) statements are synonyms, used either to obtain information about table structure or query execution plans. For more information, see [Section 13.7.7.5, “SHOW COLUMNS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns), and [Section 13.8.2, “EXPLAIN Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain).

### 13.8.2 EXPLAIN Statement

{EXPLAIN | DESCRIBE | DESC}

***tbl\_name*** [***col\_name*** | ***wild***]

{EXPLAIN | DESCRIBE | DESC}

[***explain\_type***]

{***explainable\_stmt*** | FOR CONNECTION ***connection\_id***}

{EXPLAIN | DESCRIBE | DESC} ANALYZE [FORMAT = TREE] ***select\_statement***

***explain\_type***: {

FORMAT = ***format\_name***

}

***format\_name***: {

TRADITIONAL

| JSON

| TREE

}

***explainable\_stmt***: {

SELECT statement

| TABLE statement

| DELETE statement

| INSERT statement

| REPLACE statement

| UPDATE statement

}

The [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) and [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) statements are synonyms. In practice, the [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) keyword is more often used to obtain information about table structure, whereas [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) is used to obtain a query execution plan (that is, an explanation of how MySQL would execute a query).

The following discussion uses the [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) and [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) keywords in accordance with those uses, but the MySQL parser treats them as completely synonymous.

[Obtaining Table Structure Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain-table-structure)

[Obtaining Execution Plan Information](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain-execution-plan)

[Obtaining Information with EXPLAIN ANALYZE](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain-analyze)

#### Obtaining Table Structure Information

[**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) provides information about the columns in a table:

mysql> **DESCRIBE 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 | |

+------------+----------+------+-----+---------+----------------+

[**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) is a shortcut for [**SHOW COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns). These statements also display information for views. The description for [**SHOW COLUMNS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns) provides more information about the output columns. See [Section 13.7.7.5, “SHOW COLUMNS Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-columns).

By default, [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) displays information about all columns in the table. ***col\_name***, if given, is the name of a column in the table. In this case, the statement displays information only for the named column. ***wild***, if given, is a pattern string. It can contain the SQL **%** and **\_** wildcard characters. In this case, the statement displays output only for the columns with names matching the string. There is no need to enclose the string within quotation marks unless it contains spaces or other special characters.

The [**DESCRIBE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#describe) statement is provided for compatibility with Oracle.

The [**SHOW CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-create-table), [**SHOW TABLE STATUS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-table-status), and [**SHOW INDEX**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-index) statements also provide information about tables. See [Section 13.7.7, “SHOW Statements”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show).

#### Obtaining Execution Plan Information

The [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) statement provides information about how MySQL executes statements:

[**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) works with [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select), [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete), [**INSERT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#insert), [**REPLACE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#replace), and [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) statements. In MySQL 8.0.19 and later, it also works with [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statements.

When [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) is used with an explainable statement, MySQL displays information from the optimizer about the statement execution plan. That is, MySQL explains how it would process the statement, including information about how tables are joined and in which order. For information about using [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) to obtain execution plan information, see [Section 8.8.2, “EXPLAIN Output Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#explain-output).

When [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) is used with **FOR CONNECTION *connection\_id*** rather than an explainable statement, it displays the execution plan for the statement executing in the named connection. See [Section 8.8.4, “Obtaining Execution Plan Information for a Named Connection”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#explain-for-connection).

For explainable statements, [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) produces additional execution plan information that can be displayed using [**SHOW WARNINGS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#show-warnings). See [Section 8.8.3, “Extended EXPLAIN Output Format”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#explain-extended).

[**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) is useful for examining queries involving partitioned tables. See [Section 24.3.5, “Obtaining Information About Partitions”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\partitioning.html#partitioning-info).

The **FORMAT** option can be used to select the output format. **TRADITIONAL** presents the output in tabular format. This is the default if no **FORMAT** option is present. **JSON** format displays the information in JSON format. In MySQL 8.0.16 and later, **TREE** provides tree-like output with more precise descriptions of query handling than the **TRADITIONAL** format; it is the only format which shows hash join usage (see [Section 8.2.1.4, “Hash Join Optimization”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\optimization.html#hash-joins)) and is always used for **EXPLAIN ANALYZE**.

[**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) requires the same privileges required to execute the explained statement. Additionally, [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) also requires the [**SHOW VIEW**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_show-view) privilege for any explained view. [**EXPLAIN ... FOR CONNECTION**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) also requires the [**PROCESS**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\security.html#priv_process) privilege if the specified connection belongs to a different user.

With the help of [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain), you can see where you should add indexes to tables so that the statement executes faster by using indexes to find rows. You can also use [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) to check whether the optimizer joins the tables in an optimal order. To give a hint to the optimizer to use a join order corresponding to the order in which the tables are named in a [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statement, begin the statement with **SELECT STRAIGHT\_JOIN** rather than just [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select). (See [Section 13.2.10, “SELECT Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select).)

The optimizer trace may sometimes provide information complementary to that of [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain). However, the optimizer trace format and content are subject to change between versions. For details, see [MySQL Internals: Tracing the Optimizer](https://dev.mysql.com/doc/internals/en/optimizer-tracing.html).

If you have a problem with indexes not being used when you believe that they should be, run [**ANALYZE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table) to update table statistics, such as cardinality of keys, that can affect the choices the optimizer makes. See [Section 13.7.3.1, “ANALYZE TABLE Statement”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#analyze-table).

**Note**

MySQL Workbench has a Visual Explain capability that provides a visual representation of [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) output. See [Tutorial: Using Explain to Improve Query Performance](https://dev.mysql.com/doc/workbench/en/wb-tutorial-visual-explain-dbt3.html).

#### Obtaining Information with EXPLAIN ANALYZE

MySQL 8.0.18 introduces **EXPLAIN ANALYZE**, which runs a statement and produces [**EXPLAIN**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#explain) output along with timing and additional, iterator-based, information about how the optimizer's expectations matched the actual execution. For each iterator, the following information is provided:

Estimated execution cost

(Some iterators are not accounted for by the cost model, and so are not included in the estimate.)

Estimated number of returned rows

Time to return first row

Time to return all rows (actual cost), in milliseconds

(When there are multiple loops, this figure shows the average time per loop.)

Number of rows returned by the iterator

Number of loops

The query execution information is displayed using the **TREE** output format, in which nodes represent iterators. **EXPLAIN ANALYZE** always uses the **TREE** output format. In MySQL 8.0.21 and later, this can optionally be specified explicitly using **FORMAT=TREE**; formats other than **TREE** remain unsupported.

**EXPLAIN ANALYZE** can be used with [**SELECT**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#select) statements, as well as with multi-table [**UPDATE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#update) and [**DELETE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#delete) statements. Beginning with MySQL 8.0.19, it can also be used with [**TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#table) statements.

Beginning with MySQL 8.0.20, you can terminate this statement using [**KILL QUERY**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#kill) or **CTRL-C**.

**EXPLAIN ANALYZE** cannot be used with **FOR CONNECTION**.

Example output:

mysql> **EXPLAIN ANALYZE SELECT \* FROM t1 JOIN t2 ON (t1.c1 = t2.c2)\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXPLAIN: -> Inner hash join (t2.c2 = t1.c1) (cost=4.70 rows=6)

(actual time=0.032..0.035 rows=6 loops=1)

-> Table scan on t2 (cost=0.06 rows=6)

(actual time=0.003..0.005 rows=6 loops=1)

-> Hash

-> Table scan on t1 (cost=0.85 rows=6)

(actual time=0.018..0.022 rows=6 loops=1)

mysql> EXPLAIN ANALYZE SELECT \* FROM t3 WHERE i > 8\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXPLAIN: -> Filter: (t3.i > 8) (cost=1.75 rows=5)

(actual time=0.019..0.021 rows=6 loops=1)

-> Table scan on t3 (cost=1.75 rows=15)

(actual time=0.017..0.019 rows=15 loops=1)

mysql> EXPLAIN ANALYZE SELECT \* FROM t3 WHERE pk > 17\G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EXPLAIN: -> Filter: (t3.pk > 17) (cost=1.26 rows=5)

(actual time=0.013..0.016 rows=5 loops=1)

-> Index range scan on t3 using PRIMARY (cost=1.26 rows=5)

(actual time=0.012..0.014 rows=5 loops=1)

The tables used in the example output were created by the statements shown here:

CREATE TABLE t1 (

c1 INTEGER DEFAULT NULL,

c2 INTEGER DEFAULT NULL

);

CREATE TABLE t2 (

c1 INTEGER DEFAULT NULL,

c2 INTEGER DEFAULT NULL

);

CREATE TABLE t3 (

pk INTEGER NOT NULL PRIMARY KEY,

i INTEGER DEFAULT NULL

);

Values shown for **actual time** in the output of this statement are expressed in milliseconds.

### 13.8.3 HELP Statement

HELP '***search\_string***'

The [**HELP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#help) statement returns online information from the MySQL Reference Manual. Its proper operation requires that the help tables in the **mysql** database be initialized with help topic information (see [Section 5.1.17, “Server-Side Help Support”](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\server-administration.html#server-side-help-support)).

The [**HELP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#help) statement searches the help tables for the given search string and displays the result of the search. The search string is not case-sensitive.

The search string can contain the wildcard characters **%** and **\_**. These have the same meaning as for pattern-matching operations performed with the [**LIKE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#operator_like) operator. For example, **HELP 'rep%'** returns a list of topics that begin with **rep**.

The HELP statement understands several types of search strings:

At the most general level, use **contents** to retrieve a list of the top-level help categories:

HELP 'contents'

For a list of topics in a given help category, such as **Data Types**, use the category name:

HELP 'data types'

For help on a specific help topic, such as the [**ASCII()**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html#function_ascii) function or the [**CREATE TABLE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#create-table) statement, use the associated keyword or keywords:

HELP 'ascii'

HELP 'create table'

In other words, the search string matches a category, many topics, or a single topic. You cannot necessarily tell in advance whether a given search string returns a list of items or the help information for a single help topic. However, you can tell what kind of response [**HELP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#help) returned by examining the number of rows and columns in the result set.

The following descriptions indicate the forms that the result set can take. Output for the example statements is shown using the familiar “tabular” or “vertical” format that you see when using the [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) client, but note that [**mysql**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\programs.html#mysql) itself reformats [**HELP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#help) result sets in a different way.

Empty result set

No match could be found for the search string.

Result set containing a single row with three columns

This means that the search string yielded a hit for the help topic. The result has three columns:

**name**: The topic name.

**description**: Descriptive help text for the topic.

**example**: Usage example or examples. This column might be blank.

Example: **HELP 'replace'**

Yields:

name: REPLACE

description: Syntax:

REPLACE(str,from\_str,to\_str)

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.

example: mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');

-> 'WwWwWw.mysql.com'

Result set containing multiple rows with two columns

This means that the search string matched many help topics. The result set indicates the help topic names:

**name**: The help topic name.

**is\_it\_category**: **Y** if the name represents a help category, **N** if it does not. If it does not, the **name** value when specified as the argument to the [**HELP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#help) statement should yield a single-row result set containing a description for the named item.

Example: **HELP 'status'**

Yields:

+-----------------------+----------------+

| name | is\_it\_category |

+-----------------------+----------------+

| SHOW | N |

| SHOW ENGINE | N |

| SHOW MASTER STATUS | N |

| SHOW PROCEDURE STATUS | N |

| SHOW SLAVE STATUS | N |

| SHOW STATUS | N |

| SHOW TABLE STATUS | N |

+-----------------------+----------------+

Result set containing multiple rows with three columns

This means the search string matches a category. The result set contains category entries:

**source\_category\_name**: The help category name.

**name**: The category or topic name

**is\_it\_category**: **Y** if the name represents a help category, **N** if it does not. If it does not, the **name** value when specified as the argument to the [**HELP**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#help) statement should yield a single-row result set containing a description for the named item.

Example: **HELP 'functions'**

Yields:

+----------------------+-------------------------+----------------+

| source\_category\_name | name | is\_it\_category |

+----------------------+-------------------------+----------------+

| Functions | CREATE FUNCTION | N |

| Functions | DROP FUNCTION | N |

| Functions | Bit Functions | Y |

| Functions | Comparison operators | Y |

| Functions | Control flow functions | Y |

| Functions | Date and Time Functions | Y |

| Functions | Encryption Functions | Y |

| Functions | Information Functions | Y |

| Functions | Logical operators | Y |

| Functions | Miscellaneous Functions | Y |

| Functions | Numeric Functions | Y |

| Functions | String Functions | Y |

+----------------------+-------------------------+----------------+

### 13.8.4 USE Statement

USE ***db\_name***

The [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement tells MySQL to use the named database as the default (current) database for subsequent statements. This statement requires some privilege for the database or some object within it.

The named database remains the default until the end of the session or another [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#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 name must be specified on a single line. Newlines in database names are not supported.

Making a particular database the default by means of the [**USE**](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\sql-statements.html#use) statement does not preclude accessing tables in other databases. The following example accesses the **author** table from the **db1** database and the **editor** table from the **db2** database:

USE db1;

SELECT author\_name,editor\_name FROM author,db2.editor

WHERE author.editor\_id = db2.editor.editor\_id;

|  |  |  |
| --- | --- | --- |
| [Prev](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\functions.html) | Up | [Next](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\data-dictionary.html) |
| Chapter 12 Functions and Operators | [Home](file:///E:\backup\%E4%B8%8B%E8%BD%BD\refman-8.0-en.html-chapter\refman-8.0-en.html-chapter\index.html) | Chapter 14 MySQL Data Dictionary |