https://www.simplilearn.com/tutorials/sql-tutorial/what-is-normalization-in-sql

Myql

**Query cache**

The query cache does not return stale data. When tables are modified, any relevant entries in the query cache are flushed.

The query cache is deprecated as of MySQL 5.7.20, and is removed in MySQL 8.0.

The query cache is not supported for partitioned tables, and is automatically disabled for queries involving partitioned tables. The query cache cannot be enabled for such queries.

To disable the query cache at server startup, set the [query\_cache\_size](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_query_cache_size) system variable to 0. By disabling the query cache code, there is no noticeable overhead.

Be cautious about sizing the query cache excessively large, which increases the overhead required to maintain the cache, possibly beyond the benefit of enabling it. Sizes in tens of megabytes are usually beneficial. Sizes in the hundreds of megabytes might not be.

**Replication**

Replication enables data from one MySQL database server (known as a source) to be copied to one or more MySQL database servers (known as replicas). Replication is asynchronous by default; replicas do not need to be connected permanently to receive updates from a source. Depending on the configuration, you can replicate all databases, selected databases, or even selected tables within a database.

Advantages of replication in MySQL include:

Scale-out solutions - spreading the load among multiple replicas to improve performance. In this environment, all writes and updates must take place on the source server. Reads, however, may take place on one or more replicas. This model can improve the performance of writes (since the source is dedicated to updates), while dramatically increasing read speed across an increasing number of replicas.

Data security - because the replica can pause the replication process, it is possible to run backup services on the replica without corrupting the corresponding source data.

Analytics - live data can be created on the source, while the analysis of the information can take place on the replica without affecting the performance of the source.

Long-distance data distribution - you can use replication to create a local copy of data for a remote site to use, without permanent access to the source.

The traditional method is based on replicating events from the source's binary log, and requires the log files and positions in them to be synchronized between source and replica. The newer method based on global transaction identifiers (GTIDs) is transactional and therefore does not require working with log files or positions within these files, which greatly simplifies many common replication tasks.

Replication using GTIDs guarantees consistency between source and replica as long as all transactions committed on the source have also been applied on the replica.

**Natural Language Full-Text Searches**

By default or with the IN NATURAL LANGUAGE MODE modifier, the [MATCH()](https://dev.mysql.com/doc/refman/8.0/en/fulltext-search.html#function_match) function performs a natural language search for a string against a text collection. A collection is a set of one or more columns included in a FULLTEXT index. The search string is given as the argument to AGAINST(). For each row in the table, [MATCH()](https://dev.mysql.com/doc/refman/8.0/en/fulltext-search.html#function_match) returns a relevance value; that is, a similarity measure between the search string and the text in that row in the columns named in the [MATCH()](https://dev.mysql.com/doc/refman/8.0/en/fulltext-search.html#function_match) list

mysql> SELECT \* FROM articles

-> WHERE MATCH (title,body)

-> AGAINST ('database' IN NATURAL LANGUAGE MODE);

**Storage Engines in Mysql**

InnoDB and MYISAM, are storage engines for MySQL.

These two differ on their locking implementation: InnoDB locks the particular row in the table, and MyISAM locks the entire MySQL table.

You can specify the type by giving MYISAM OR InnoDB while creating a table in DB.

**InnoDB** is a transactional storage engine of MySQL whereas **MyISAM** is a non-transactional storage engine. In other words, InnoDB follows the ACID properties to maintain the integrity of data but MyISAM doesn't follow ACID properties thus failing to maintain the integrity of the data.

MyISAM does not follow ACID as opposed to InnoDB which follows transactions to maintain integrity of the data.

MyISAM supports concurrent inserts: If a table has no free blocks in the middle of the data file, you can INSERT new rows into it at the same time that other threads are reading from the table. [MySqlDoc](https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html)

That is why, MyISAM is faster and takes less space. For instance, the MySQL MyISAM Storage Engine does not support tranactions.[constraints of MySQL MYISAM](https://dev.mysql.com/doc/refman/5.6/en/myisam-storage-engine.html) There is a bit called [concurrent-insert](https://dev.mysql.com/doc/refman/5.6/en/server-system-variables.html#sysvar_concurrent_insert) By default, the variable is set to 1 and concurrent inserts are handled as just described. If it is set to 0, concurrent inserts are disabled. If it is set to 2, concurrent inserts at the end of the table are permitted even for tables that have deleted rows. An INSERT statement can be executed to add rows to the end of the table with select at same time if there are no holes/deleted rows in middle of table (at time of concurrent insert).

The default isolation level og mysql InnoDB is "Read Repeatable". For MyISAM, there is no transaction. InnoDB uses row level locking while MyISAM can only use table level locking that is why InnoDB has crash revovery is better than MyISAM. One has to **manually acquire the table level lock** in MyISAM if one wants to avoid the concurrency effects.

InnoDB is the default NOT myISAM <https://dev.mysql.com/doc/refman/5.7/en/innodb-introduction.html> "InnoDB is the default MySQL storage engine. Unless you have configured a different default storage engine, issuing a CREATE TABLE statement without an ENGINE= clause creates an InnoDB table"

Msql supports Json data type.

* [Creating JSON Values](https://dev.mysql.com/doc/refman/8.0/en/json.html#json-values)
* [Normalization, Merging, and Autowrapping of JSON Values](https://dev.mysql.com/doc/refman/8.0/en/json.html#json-normalization)
* [Searching and Modifying JSON Values](https://dev.mysql.com/doc/refman/8.0/en/json.html#json-paths)
* [JSON Path Syntax](https://dev.mysql.com/doc/refman/8.0/en/json.html#json-path-syntax)
* [Comparison and Ordering of JSON Values](https://dev.mysql.com/doc/refman/8.0/en/json.html#json-comparison)
* [Converting between JSON and non-JSON values](https://dev.mysql.com/doc/refman/8.0/en/json.html#json-converting-between-types)
* [Aggregation of JSON Values](https://dev.mysql.com/doc/refman/8.0/en/json.html#json-aggregation)
* mysql> CREATE TABLE t1 (jdoc JSON);
* Query OK, 0 rows affected (0.20 sec)
* mysql> INSERT INTO t1 VALUES('{"key1": "value1", "key2": "value2"}');
* Query OK, 1 row affected (0.01 sec)

https://dev.mysql.com/doc/refman/8.0/en/functions.html

|  |  |  |  |
| --- | --- | --- | --- |
| [MIN()](https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html#function_min) | Return the minimum value |  |  |
| [MINUTE()](https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_minute) | Return the minute from the argument |  |  |
| [MOD()](https://dev.mysql.com/doc/refman/8.0/en/mathematical-functions.html#function_mod) | Return the remainder |  |  |
| [MONTH()](https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_month) | Return the month from the date passed |  |  |
| [MONTHNAME()](https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_monthname) | Return the name of the month |  |  |

SELECT year, SUM(profit) AS profit

FROM sales

GROUP BY year;

**Datatypes supported by mysql**

**Numeric data types:**

MySQL supports all standard SQL numeric data types. These types include the exact numeric data types ([INTEGER](https://dev.mysql.com/doc/refman/8.0/en/integer-types.html), [SMALLINT](https://dev.mysql.com/doc/refman/8.0/en/integer-types.html), [DECIMAL](https://dev.mysql.com/doc/refman/8.0/en/fixed-point-types.html), and [NUMERIC](https://dev.mysql.com/doc/refman/8.0/en/fixed-point-types.html)), as well as the approximate numeric data types ([FLOAT](https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html), [REAL](https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html), and [DOUBLE PRECISION](https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html)). The keyword [INT](https://dev.mysql.com/doc/refman/8.0/en/integer-types.html) is a synonym for [INTEGER](https://dev.mysql.com/doc/refman/8.0/en/integer-types.html), and the keywords [DEC](https://dev.mysql.com/doc/refman/8.0/en/fixed-point-types.html) and [FIXED](https://dev.mysql.com/doc/refman/8.0/en/fixed-point-types.html) are synonyms for [DECIMAL](https://dev.mysql.com/doc/refman/8.0/en/fixed-point-types.html). MySQL treats [DOUBLE](https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html) as a synonym for [DOUBLE PRECISION](https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html) (a nonstandard extension). MySQL also treats [REAL](https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html) as a synonym for [DOUBLE PRECISION](https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html) (a nonstandard variation), unless the [REAL\_AS\_FLOAT](https://dev.mysql.com/doc/refman/8.0/en/sql-mode.html#sqlmode_real_as_float) SQL mode is enabled.

The [BIT](https://dev.mysql.com/doc/refman/8.0/en/bit-type.html) data type stores bit values and is supported for [MyISAM](https://dev.mysql.com/doc/refman/8.0/en/myisam-storage-engine.html" \o "16.2 The MyISAM Storage Engine), [MEMORY](https://dev.mysql.com/doc/refman/8.0/en/memory-storage-engine.html), [InnoDB](https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html), and [NDB](https://dev.mysql.com/doc/refman/8.0/en/mysql-cluster.html) tables.

* **Out of range and overflow handling**

MySQL handles assignment of out-of-range values to columns and overflow during expression evaluation

* If strict SQL mode is enabled, MySQL rejects the out-of-range value with an error, and the insert fails, in accordance with the SQL standard.
* If no restrictive modes are enabled, MySQL clips the value to the appropriate endpoint of the column data type range and stores the resulting value instead.

Example:

CREATE TABLE t1 (i1 TINYINT, i2 TINYINT UNSIGNED);

**Strict mode :**

mysql> SET sql\_mode = 'TRADITIONAL';

mysql> INSERT INTO t1 (i1, i2) VALUES(256, 256);

ERROR 1264 (22003): Out of range value for column 'i1' at row 1

mysql> SELECT \* FROM t1;

Empty set (0.00 sec)

**Strict mode :**

mysql> SET sql\_mode = '';

mysql> INSERT INTO t1 (i1, i2) VALUES(256, 256);

mysql> SHOW WARNINGS;

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

| Level | Code | Message |

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

| Warning | 1264 | Out of range value for column 'i1' at row 1 |

| Warning | 1264 | Out of range value for column 'i2' at row 1 |

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

mysql> SELECT \* FROM t1;

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

| i1 | i2 |

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

| 127 | 255 |

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

**Date and Time data types**

The date and time data types for representing temporal values are [DATE](https://dev.mysql.com/doc/refman/8.0/en/datetime.html), [TIME](https://dev.mysql.com/doc/refman/8.0/en/time.html), [DATETIME](https://dev.mysql.com/doc/refman/8.0/en/datetime.html), [TIMESTAMP](https://dev.mysql.com/doc/refman/8.0/en/datetime.html), and [YEAR](https://dev.mysql.com/doc/refman/8.0/en/year.html). Each temporal type has a range of valid values, as well as a “zero” value that may be used when you specify an invalid value that MySQL cannot represent. The [TIMESTAMP](https://dev.mysql.com/doc/refman/8.0/en/datetime.html) and [DATETIME](https://dev.mysql.com/doc/refman/8.0/en/datetime.html) types have special automatic updating behavior.

For any [TIMESTAMP](https://dev.mysql.com/doc/refman/8.0/en/datetime.html) or [DATETIME](https://dev.mysql.com/doc/refman/8.0/en/datetime.html) column in a table, you can assign the current timestamp as the default value, the auto-update value, or both:

CREATE TABLE t1 (

ts TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

dt DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

CREATE TABLE t1 (

ts TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

dt DATETIME DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE t1 (

ts TIMESTAMP DEFAULT 0,

dt DATETIME DEFAULT 0

);

CREATE TABLE t1 (

ts TIMESTAMP DEFAULT 0 ON UPDATE CURRENT\_TIMESTAMP,

dt DATETIME DEFAULT 0 ON UPDATE CURRENT\_TIMESTAMP

);

**Difference between TIMESTAMP and DATETIME**

CREATE TABLE t1 (

ts1 TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP, -- default 0

ts2 TIMESTAMP NULL ON UPDATE CURRENT\_TIMESTAMP -- default NULL

);

CREATE TABLE t1 (

dt1 DATETIME ON UPDATE CURRENT\_TIMESTAMP, -- default NULL

dt2 DATETIME NOT NULL ON UPDATE CURRENT\_TIMESTAMP -- default 0

);

[**https://dev.mysql.com/doc/refman/8.0/en/timestamp-initialization.html**](https://dev.mysql.com/doc/refman/8.0/en/timestamp-initialization.html)

INSERT INTO t2 VALUES (CURRENT\_TIMESTAMP);

INSERT INTO t1 VALUES (NOW());

**View In Mysql**

A view can be created from many kinds of [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) statements. It can refer to base tables or other views. It can use joins, [UNION](https://dev.mysql.com/doc/refman/8.0/en/union.html), and subqueries. The [SELECT](https://dev.mysql.com/doc/refman/8.0/en/select.html) need not even refer to any tables. The following example defines a view that selects two columns from another table, as well as an expression calculated from those columns:

mysql> CREATE TABLE t (qty INT, price INT);

mysql> INSERT INTO t VALUES(3, 50), (5, 60);

mysql> CREATE VIEW v AS SELECT qty, price, qty\*price AS value FROM t;

mysql> SELECT \* FROM v;

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

| qty | price | value |

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

| 3 | 50 | 150 |

| 5 | 60 | 300 |

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

mysql> SELECT \* FROM v WHERE qty = 5;

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

| qty | price | value |

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

| 5 | 60 | 300 |

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

**Mysql Tranaction**