MySQL Data Types

Last update on August 19 2022 21:51:22 (UTC/GMT +8 hours)

What is data type

1. A data type specifies a particular type of data, such as integer, floating-point, Boolean etc.

2. A data type also specifies the possible values for that type, the operations that can be performed on that type and the way the values of that type are stored.

MySQL data types

MySQL supports a number of SQL standard data types in various categories. MySQL has **Numeric** Types, the **DATETIME**, **DATE**, and **TIMESTAMP** Types and **String** Types. Data types are discussed on this page are based on MySQL community server 5.6

MySQL Numeric Types

MySQL supports all standard SQL numeric data types which include INTEGER, SMALLINT, DECIMAL, and NUMERIC. It also supports the approximate numeric data types (FLOAT, REAL, and DOUBLE PRECISION). The keyword INT is a synonym for INTEGER, and the keywords DEC and FIXED are synonyms for DECIMAL. DOUBLE is a synonym for DOUBLE PRECISION (a nonstandard extension). REAL is a synonym for DOUBLE PRECISION (a nonstandard variation) unless the REAL_AS_FLOAT SQL mode is enabled. The BIT data type stores bit-field values and is supported for MyISAM, MEMORY, InnoDB, and NDB tables.

Integer types

SQL standard integer types INTEGER (or INT) and SMALLINT are supported by MySQL. As an extension to the standard, MySQL also supports the integer types TINYINT, MEDIUMINT, and BIGINT. Following table shows the required storage and range (maximum and minimum value for signed and unsigned integer) for each integer type.

Type	Length in Bytes	Minimum Value (Signed)	Maximum Value (Signed)	Minimum Value (Unsigned)	Maximum Value (Unsigned)

TINYINT	1	-128	127	0	255
SMALLINT	2	-32768	32767	0	65535
MEDIUMINT	3	-8388608	8388607 to	0	16777215
INT	4	-2147483648	2147483647	0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807	0	18446744073709551615

Floating-Point Types

The FLOAT and DOUBLE types represent approximate numeric data values. MySQL uses four bytes for single-precision values and eight bytes for double-precision values.

Types	Description
FLOAT	A precision from 0 to 23 results in a four-byte single-precision FLOAT column
DOUBLE	A precision from 24 to 53 results in an eight-byte double-precision DOUBLE column.

MySQL allows a nonstandard syntax: FLOAT(M,D) or REAL(M,D) or DOUBLE PRECISION(M,D). Here values can be stored up to M digits in total where D represents the decimal point. For example, a column defined as FLOAT(8,5) will look like -999.99999. MySQL performs rounding when storing values, so if you insert 999.00009 into a FLOAT(7,4) column, the approximate result is 999.0001.

Following table shows the required storage and range (maximum and minimum value for signed and unsigned integer) for each floating-point type.

Type	Length in Bytes	Minimum Value (Signed)	Maximum Value (Signed)	Minimum Value (Unsigned)	Maximum Va (Unsigned)
FLOAT	4	-3.402823466E+38	-1.175494351E-38	1.175494351E-38	3.402823466
DOUBLE	8	-1.7976931348623157E+ 308	-2.2250738585072014E- 308	0, and 2.2250738585072014E- 308	1.797693134 7E+ 308

Fixed-Point Types

Fixed-Point data types are used to preserve exact precision, for example with currency data. In MySQL DECIMAL and NUMERIC types store exact numeric data values. MySQL 5.6 stores DECIMAL values in binary format.

In standard SQL the syntax DECIMAL(5,2) (where 5 is the precision and 2 is the scale.) be able to store any value with five digits and two decimals. Therefore the value range will be from -999.99 to 999.99. The syntax DECIMAL(*M*) is equivalent to DECIMAL(*M*,0). Similarly, the syntax DECIMAL is equivalent to DECIMAL(*M*,0). MySQL supports both of these variant forms of DECIMAL syntax. The default value of *M* is 10. If the scale is 0, DECIMAL values contain no decimal point or fractional part.
The maximum number of digits for DECIMAL is 65, but the actual range for a given DECIMAL column can be constrained by the precision or scale for a given column.

Bit Value Types

The BIT data type is used to store bit-field values. A type of BIT(N) enables storage of N-bit values. N can range from 1 to 64.
To specify bit values, b'value' notation can be used. value is a binary value written using zeros and ones. For example, b'111' and b'10000000' represent 7 and 128, respectively

Numeric type attributes

MySQL supports an extension for optionally specifying the display width of integer data types in parentheses following the base keyword for the type

Types	Description
TYPE(N)	Where N is an integer and display width of the type is upto N digits.
ZEROFILL	The default padding of spaces is replaced with zeros. So, for a column INT(3) ZEROFILL, 7 is displayed as 007.

MySQL Date and Time Types

The date and time types represent DATE, TIME, DATETIME, TIMESTAMP, and YEAR. Each type has a range of valid values, as well as a “zero” value.

DATETIME, DATE, and TIMESTAMP Types

Types	Description	Display Format	Range

DATETIME	Use when you need values containing both date and time information.	YYYY-MM-DD HH:MM:SS	'1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
DATE	Use when you need only date information.	YYYY-MM-DD	'1000-01-01' to '9999-12-31'.
TIMESTAMP	Values are converted from the current time zone to UTC while storing and converted back from UTC to the current time zone when retrieved.	YYYY-MM-DD HH:MM:SS	'1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC

Time Type

MySQL fetches and displays TIME values in 'HH:MM:SS' format or 'HHH:MM:SS' format The range of. TIME values from '-838:59:59' to '838:59:59'. The hours part may be rather large because not only the TIME type can be used to represent the time of day, i.e. less than 24 hours, but also the passed time or a time of interval between two events.

The TIME values in MySQL can be recognized in many different formats, some of which can include a trailing fractional seconds part in up to 6 digits microseconds precision. The range for TIME values is '-838:59:59.000000' to '838:59:59.000000'.

MySQL explains abbreviated TIME values with colons as the time of the day. Suppose '09:10' means '09:10:00', not '00:09:10'. MySQL understands the abbreviated values without colons as that, the two rightmost digits represent seconds. For example, we think of '0910' and 0910 as meaning '09:10:00', i.e. 10 minutes after 9 o'clock but the reality is MySQL understand them as '00:09:10', i.e. 9 minutes and 10 seconds. So, be careful about using abbreviated time in MySQL.

By default, the values of time that lie outside the TIME are converted to the valid range of time values. For example, '-930:00:00' and '930:00:00' are converted to '-838:59:59' and '838:59:59'. Invalid TIME values are converted to '00:00:00', because '00:00:00' is itself a valid TIME value in MySQL.

Year Type

The YEAR type is a 1-byte type used to represent year values. It can be declared as YEAR(2) or YEAR(4) to specify a display width of two or four characters. If no width is given the default is

four characters

YEAR(4) and YEAR(2) have different display format but have the same range of values.
For 4-digit format, MySQL displays YEAR values in YYYY format, with a range of 1901 to 2155, or 0000.
For 2-digit format, MySQL displays only the last two (least significant) digits; for example, 70 (1970 or 2070) or 69 (2069).

You can specify YEAR values in a variety of formats:

String length	Range
4-digit string	'1901' to '2155'.
4-digit number	1901 to 2155.
1- or 2-digit string	'0' to '99'. Values in the ranges '0' to '69' and '70' to '99' are converted to YEAR values in the ranges 2000 to 2069 and 1970 to 1999.
1- or 2-digit number	1 to 99. Values in the ranges 1 to 69 and 70 to 99 are converted to YEAR values in the ranges 2001 to 2069 and 1970 to 1999.

String Types

The string types are CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM, and SET.

CHAR and VARCHAR Types

The CHAR and VARCHAR types are similar but differ in the way they are stored and retrieved. They also differ in maximum length and in whether trailing spaces are retained.

Types	Description	Display Format	Range in characters
CHAR	Contains non-binary strings. Length is fixed as you declare while creating a table. When stored, they are right-padded with spaces to the specified length.	Trailing spaces are removed.	The length can be any value from 0 to 255.

VARCHAR	Contains non-binary strings. Columns are variable-length strings.	As stored.	A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions.

BINARY and VARBINARY Types

The BINARY and VARBINARY types are similar to CHAR and VARCHAR, except that they contain binary strings rather than nonbinary strings.

Types	Description	Range in bytes
BINARY	Contains binary strings.	0 to 255
VARBINARY	Contains binary strings.	A value from 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in 5.0.3 and later versions.

BLOB and TEXT Types

A BLOB is a binary large object that can hold a variable amount of data. There are four types of BLOB, TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB. These differ only in the maximum length of the values they can hold.
The four TEXT types are TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT. These correspond to the four BLOB types and have the same maximum lengths and storage requirements.

Types	Description	Categories	Range
BLOB	Large binary object that containing a variable amount of data. Values are treated as binary strings.You don't need to specify length while creating a column.	TINYBLOB	Maximum length of 255 characters.
		MEDIUMBLOB	Maximum length of 16777215 characters.
		LONGBLOB	Maximum length of 4294967295 characters

TEXT	Values are treated as character strings having a character set.	TINYBLOB	Maximum length of 255 characters.
		MEDIUMBLOB	Maximum length of 16777215 characters.
		LONGBLOB	Maximum length of 4294967295 characters

ENUM Types

A string object whose value is chosen from a list of values given at the time of table creation. For example -

```
CREATE TABLE length (     length ENUM('small', 'medium', 'large') );
```

SET Types

A string object having zero or more comma separated values (maximum 64). Values are chosen from a list of values given at the time of table creation.

Difference between MySQL Datetime and Timestamp data Types

The DATETIME type is used when you need values containing both date and time information. MySQL retrieves and displays DATETIME values in 'YYYY-MM-DD HH:MM:SS' format. The supported range is 1000-01-01 00:00:00' to '9999-12-31 23:59:59'.

The TIMESTAMP data type is also used when you need values containing both date and time information. TIMESTAMP has a range of '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC

The major difference between DATETIME and TIMESTAMP is that TIMESTAMP values are converted from the current time zone to UTC while storing, and converted back from UTC to the current time zone when retrieved. The datetime data type value is unchanged.

See the following example :

The following command displays current time zone information :

```
mysql> show variables like '%time_zone%';
+------------------+--------------------+
| Variable_name    | Value              |
+------------------+--------------------+
| system_time_zone | India Standard Time |
| time_zone        | asia/calcutta      |
+------------------+--------------------+
2 rows in set (0.00 sec)
```

Let create a table with two fields udatetime (data type -> datetime) utimestamp (data type -> timestamp) and insert one record with now() (returns the current date and time) as a value in both fields.

```
MySQL> create table tempdate (udatetime datetime, utimestamp timesta
Query OK, 0 rows affected (0.32 sec)
```

```
MySQL> insert into tempdate values ((now()), (now()));
Query OK, 1 row affected (0.05 sec)
```

Now see the records;

At this point, the datetime and timestamp data types have same values.

```
mysql> select * from tempdate;
+---------------------+---------------------+
| udatetime           | utimestamp          |
+---------------------+---------------------+
| 2013-06-29 16:30:46 | 2013-06-29 16:30:46 |
+---------------------+---------------------+
1 row in set (0.00 sec)
```

In the above output, both the fields have same values. Let change the timezone and see the result.

```
MySQL> SET TIME_ZONE ='Europe/Paris';
Query OK, 0 rows affected (0.00 sec)
```

Now execute the following command :

Sample Output:

```
MySQL> select * from tempdate;
+---------------------+---------------------+
| udatetime           | utimestamp          |
+---------------------+---------------------+
| 2013-06-29 16:55:18 | 2013-06-29 13:25:18 |
+---------------------+---------------------+
1 row in set (0.00 sec)
```

The above output shows that udatetime (data type is DATETIME) is unchanged whereas utimestamp (data type is TIMESTAMP) is changed as the new timezone is set to 'Europe/Paris'.