**TINYINT** 1 byte

**SMALLINT** 2 bytes

**MEDIUMINT** 3 bytes

**INT, INTEGER** 4 bytes

**BIGINT** 8 bytes

**FLOAT(p)** 4 bytes if 0 <= p <= 24, 8 bytes if 25 <= p <= 53

**FLOAT** 4 bytes

**DOUBLE [PRECISION], REAL** 8 bytes

**DECIMAL(M,D), NUMERIC(M,D):** Varies

**BIT(M):** approximately (M+7)/8 bytes

Now, you see that, the storage requirements of DECIMAL(M,D) or NUMERIC(M,D) is mentioned as varies.

Because, Values for [DECIMAL](https://dev.mysql.com/doc/refman/5.7/en/fixed-point-types.html" \o "11.2.2 Fixed-Point Types (Exact Value) - DECIMAL, NUMERIC) (and [NUMERIC](https://dev.mysql.com/doc/refman/5.7/en/fixed-point-types.html" \o "11.2.2 Fixed-Point Types (Exact Value) - DECIMAL, NUMERIC)) columns are represented using a binary format that packs nine decimal (base 10) digits into four bytes. Storage for the integer and fractional parts of each value are determined separately. Each multiple of nine digits requires four bytes, and the “leftover” digits require some fraction of four bytes. The storage required for excess digits is given by the following table.

| **Leftover Digits** | **Number of Bytes** |
| --- | --- |
| 0 | 0 |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 2 |
| 5 | 3 |
| 6 | 3 |
| 7 | 4 |
| 8 | 4 |

For TIME, DATETIME, and TIMESTAMP columns, the storage required for tables created before MySQL 5.6.4 differs from tables created from 5.6.4 on. This is due to a change in 5.6.4 that permits these types to have a fractional part, which requires from 0 to 3 bytes.

| **Data Type** | **Storage Required Before MySQL 5.6.4** | **Storage Required as of MySQL 5.6.4** |
| --- | --- | --- |
| **[YEAR](https://dev.mysql.com/doc/refman/5.7/en/year.html" \o "11.3.3 The YEAR Type)** | 1 byte | 1 byte |
| **[DATE](https://dev.mysql.com/doc/refman/5.7/en/datetime.html" \o "11.3.1 The DATE, DATETIME, and TIMESTAMP Types)** | 3 bytes | 3 bytes |
| **[TIME](https://dev.mysql.com/doc/refman/5.7/en/time.html" \o "11.3.2 The TIME Type)** | 3 bytes | 3 bytes + fractional seconds storage |
| **[DATETIME](https://dev.mysql.com/doc/refman/5.7/en/datetime.html" \o "11.3.1 The DATE, DATETIME, and TIMESTAMP Types)** | 8 bytes | 5 bytes + fractional seconds storage |
| **[TIMESTAMP](https://dev.mysql.com/doc/refman/5.7/en/datetime.html" \o "11.3.1 The DATE, DATETIME, and TIMESTAMP Types)** | 4 bytes | 4 bytes + fractional seconds storage |

As of MySQL 5.6.4, storage for YEAR and DATE remains unchanged. However, TIME, DATETIME, and TIMESTAMP are represented differently. DATETIME is packed more efficiently, requiring 5 rather than 8 bytes for the nonfractional part, and all three parts have a fractional part that requires from 0 to 3 bytes, depending on the fractional seconds precision of stored values.

| **Fractional Seconds Precision** | **Storage Required** |
| --- | --- |
| 0 | 0 bytes |
| 1, 2 | 1 byte |
| 3, 4 | 2 bytes |
| 5, 6 | 3 bytes |

So, TIME(0) uses 3 bytes, TIME(1) and TIME(2) 4 bytes, TIME(3) and TIME(4) four bytes. That’s what it means.

**CHAR(M):** M × w bytes, 0 <= M <= 255, where w is the number of bytes required for the maximum-length character in the character set.

**VARCHAR(M):** Now, before MySQL 5.0.6 a VARCHAR column can store upto 255 characters. From 5.0.6 onwards, MySQL can store upto 65,535 characters. L + 1 bytes if column values require 0 − 255 bytes, L + 2 bytes if values may require more than 255 bytes.

**BINARY(M):** M bytes0<=M<=255.

**VARBINARY(M):** *L* + 1 bytes if column values require 0 − 255 bytes, *L* + 2 bytes if values may require more than 255 bytes.

**TINYBLOB, TINYTEXT:** L+1 bytes where L<28 (The maximum number of characters stored in TINYBLOB or TINYTEXT is 255)

**BLOB, TEXT:** L+2 bytes. Now, the maximum number of characters stored in BLOB or TEXT is 216-1.

**MEDIUMBLOB, MEDIUMTEXT:** L+3 bytes. Now, the maximum number of characters stored in MEDIUMBLOB or MEDIUMTEXT is 224-1.

**LONGBLOB, LONGTEXT:** L+4 bytes. Now, the maximum number of characters stored in LONGBLOB or LONGTEXT is 232-1.

**ENUM('value1','value2',...):** 1 or 2 bytes, depending on the number of enumeration values (65,535 values maximum)

**SET('value1','value2',...):** 1, 2, 3, 4, or 8 bytes, depending on the number of set members (64 members maximum)