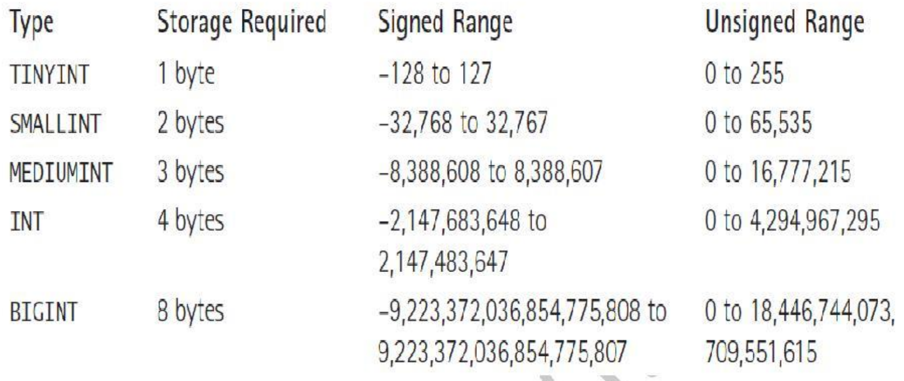
Module 3: MySQL Datatypes & Attributes

Integer:

- → □ Integer type includes
- □ For integer values declared with the UNSIGNED attribute, negative values are not allowed.



Type	Storage Required	Signed Range	Unsigned Range
TINYINT	1 byte	-128 to 127	0 to 255
SMALLINT	2 bytes	-32,768 to 32,767	0 to 65,535
MEDIUMINT	3 bytes	-8,388,608 to 8,388,607	0 to 16,777,215
INT	4 bytes	-2,147,683,648 to 2,147,483,647	0 to 4,294,967,295
BIGINT	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0 to 18,446,744,073, 709,551,615

Real:

- □ Real Datatype can be Floating-Point and Fixed-Point.
- □ Floating-Point:
  - → □ It includes
  - □ They represent values in the native floating point format used by the server hosts CPU.
  - □ The values are subject to rounding error.
- □ Fixed-Point:
  - → □ It includes
  - □ They don't represent values in the native format and hence not processed efficiently.
  - □ The values are not subject to rounding error hence popular choice for financial applications involving currency calculations.
- □ Note: With Real we can also optionally specify the precision and scale which indicate the number of significant digits and the number of decimal places to the right of the decimal point.

Char:

- → □ It is a
- □ We define CHAR(n) where n is the maximum length of acceptable values from 0 to 255.
- □ Eg: CHAR(30) requires 30 characters for each value.
- □ Values shorter than the designated length are padded with spaces to that length when they are stored.

Varchar:

- → □ It is a
- □ We define VARCHAR(n) where n is the maximum length of acceptable values from 0 to 65,535.
- □ Values stored in VARCHAR column takes only the number of characters required to store it (+ 1 or 2 bytes to record the strings length).

Enum:

- → □ Enum types are used
- □ Enum column can specify upto 65,535 values.
- □ To store a value , we code a single text string.
- □ If the value is not one of the list of strings, then insert operation fails.
- □ With enum we can also specify default values, which would be used if nothing is specified for that column.

Set:

- → □ Set types are used
- □ Set column can specify upto 64 values.
- □ To store a value we code a single string with one or more values separated by commas.
- □ If the values are not in list of strings, then insert operation fails.
- □ With set we can also specify default values, which would be used if nothing is specified for that column.

Numeric Column Attributes:

- □ **UNSIGNED:**
  - □ declaring a column as unsigned implies that column can have **only positive values**.
- □ **ZEROFILL:**
  - □ It is used to display leading zeros of a number based on display width.

- **AUTO_INCREMENT:**
  - It only works with integer numbers.
  - There can be only one AUTO_INCREMENT column in a table.
  - It automatically adds 1 to the current maximum value.
  - The default value starts at 1 and increases by 1 for every record that is inserted.
  - The start value can be changed by using AUTO_INCREMENT=number.
  - Once the max value of the datatype is reached it does not start over.
  - If we delete rows then too the values won't be reused.

General Column Attributes:

- **NULL and NOT NULL:**
  - They indicate whether a column can contain null values.
  - The default attribute for column is to allow NULL values.
- **DEFAULT:**
  - It provides a column with default value to be used when we create new record but **don't explicitly specify a value** for the column.

Note:

- It is an error to specify a default value NULL for a NOT NULL column.
- It is an error to specify default value negative number for an UNSIGNED numeric column.

MCQs

Q1) Consider a table having a column with datatype as tinyint and no records
What is the result if we perform the following insert: insert into kp1 values (123.45);

Options:

A. table has one record 123. B. error with no record inserted.
C. table has one record 123.45 D. table has one record 124.

Solution:

Q2) Consider a table having a column with datatype as tinyint and no records.
What is the result if we perform the following insert: insert into kp1 values ('a');

Options:

A. table has one record 65. B. table has one record 97.
C. table has one record a. D. error with no record inserted.

Solution:

Q3) Consider a table having a column with datatype as float(4,2) and no records.
What is the result if we perform the following insert:

insert into kp2 values(100);

Options:

A. table has one records: 100 B. table has one records: 100.00
C. table has one record : 11 D. error with no record inserted.

Solution:

Q4) Consider a table having a column with datatype as varchar(5) and no records. What is the result if we perform the following insert:

```
insert into kp4 values('kamalcls');
```

Options:

A. table has one records: kamal B. error with no record inserted.

C. table has one record : alcls D. table has one records: kamalcls

Solution:

Q5) what is the result of the following create table statement:

```
create table kp6 (a int not null default null);
```

Options:

A. table created with no records B. error table could not be created.

C. table created with one record null. D. Table created with one record 0.

Solution:

Q6) consider the following snippet:

```
mysql> create table kp7 (a int default 100);
mysql> insert into kp7 values(default) , (null);
```

What is the result?

Options:

A. table has one record 100 B. table has two records 100 and 101.

C. table has two records 100 and 100. D. table has two records with 100 and null

Solution:

Q7) consider the following snippet:

```
mysql> create table kp8 (a enum('yes', 'no'));
mysql> insert into kp8 values();
```

What is the result?

Options:

A. table has one record NULL B. table has one record yes

C. table has one record no D. table has one record which would be either yes or no

Solution:

Q8) consider the following snippet:

```
mysql> create table kp9(a int(5) );
mysql> insert into kp9 values(345678);
mysql> select * from kp9;
```

What is the result?

Options:

A. o/p: 34567 B. o/p: 345678

C. o/p: 34569 D. error.

Solution: