MYSQL Basics

Numeric Data Types

INT − signed, from -2147483648 to 2147483647, unsigned, from 0 to 4294967295. You can specify a width of up to 11 digits.

- **TINYINT** − signed, from -128 to 127, unsigned, from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT** − signed, from -32768 to 32767, from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT** − signed, from -8388608 to 8388607, from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** − signed, from -9223372036854775808 to 9223372036854775807; from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
- **FLOAT(M,D)** − Total length is M, no. of decimals is D. default is 10,2. Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D)** − Total length is M, no. of decimals is D. default is 16,4. Decimal precision can go to 53 places for a DOUBLE.

Date and time Data Types

- **DATE** − A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, 1973-12-30.

- **DATETIME** − A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 1973-12-30 15:30:00.

- **TIMESTAMP** − A timestamp between midnight, January 1st, 1970 and sometime in 2037. Example: 19731230153000 (YYYYMMDDHHMMSS).

- **TIME** − Stores the time in a HH:MM:SS format.

- **YEAR(M)** − If M=2 then YEAR can be between 1970 to 2069 (70 to 69). If M=4 then YEAR can be 1901 to 2155. The default length is 4.

String Data Types

CHAR(M) − A fixed-length string between 1 and 255 characters. Defining a length is not required, but the default is 1.

- **VARCHAR(M)** − A variable-length string between 1 and 255 characters.
- **BLOB or TEXT** − A field with a maximum length of 65535 characters. "Binary Large Objects" are images or other types of files.
- **TINYBLOB or TINYTEXT** − maximum length of 255 characters. You do not specify a length.
- **MEDIUMBLOB or MEDIUMTEXT** − maximum length of 16777215 characters. You do not specify a length.
- **LONGBLOB or LONGTEXT** − maximum length of 4294967295 characters. You do not specify a length.
- **ENUM** − When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

How to Create a Table?

The general form to create a table is as follows:

CREATE TABLE table_name
(A_1 D_1, A_2 D_2,...A_n D_n,
(integrity-constraint$_1$),
(integrity-constraint$_2$),
.....
.....
(integrity-constraint$_n$));

Where, A_i=Attribute/filed/column name
and D_i= datatype/domain

Note: We will discuss integrity constraints later.

Example

Example 01:

Create a table student with column id, stName, fatherName, address, birthDay.

Solution:

```
create table student
(id number(10), stName varchar2(30), fatherName varchar2(30), address varchar2(40), birthDay date);
```

Example 02:

Create a table student with column id, stName, fatherName, address, birthDay and set id as a primary key.

Solution:

```
create table student
(id number(10), stName varchar2(30), fatherName varchar2(30), address varchar2(40), birthDay date, primary
key(id));
```

How to Create a Table form an Existing Table

The general form to create a table from an existing table is as follows:

CREATE TABLE table_name

[(A_1, A_2,...A_n)]

AS SELECT A_1, A_2, ...,A_n FROM existingTableName;

Note: You have to mentioned column names after AS SELECT that you are going to use in the target table.

Example

Example 03:

Create a table employee with columns emName, fatherName, address form a table student with column stName, fatherName, address.

Solution:

```
create table employee
(emName, fatherName, address)
AS SELECT stName, fatherName, address FROM student;
```

Example 04:

Create a table employee with exactly the same columns as student table.

Solution:

```
create table employee
AS SELECT * FROM student;
```

How to Add Additional Columns to an Existing Table

The general form to add additional columns to an existing table is as follows:

ALTER TABLE table_name
ADD (A_1 D_1, A_2 D_2,... A_n D_n);

Example 05:

Add deptName and hallName to student table.

Solution:

ALTER TABLE student
ADD (deptName varchar2(20), hallName varchar2(20))

How to Change Columns to an Existing Table

The general form to change columns in an existing table is as follows:

ALTER TABLE table_name
MODIFY (A_1 D_1, A_2 D_2,... A_n D_n);

Example 06:

Change column deptName to number to student table.

Solution:

```
ALTER TABLE student
MODIFY (deptName number(3))
```

How to DELETE Columns from an Existing Table

The general form to delete columns in an existing table is as follows:

```
ALTER TABLE table_name
   DROP COLUMN (A1, A2,...,An);
```

Note: If the column to delete has some data then we have to use UNUSED clause first.

```
ALTER TABLE table_name
   SET UNUSED COLUMN (A1, A2,...,An);
```

then perform

```
ALTER TABLE table_name
   DROP COLUMN (A1, A2,...,An);
```

Example

Example 07:

Delete column deptName which is empty and hallName which is not from student table.

Solution:

```
ALTER TABLE student
   SET UNUSED COLUMN hallName;

ALTER TABLE student
   DROP COLUMN (deptName, hallName);
```

How to Rename an Existing Table

The general form to rename an existing table is as follows:

RENAME oldName TO newName;

Example 08:

Rename existing student table to studentInfo table.

Solution:

RENAME student to studentInfo;

How to Delete an Existing Table

The general form to delete an existing table is as follows:

```
DROP TABLE tableName;
```

Example 08:

Delete existing student table.

Solution:

```
DROP TABLE student;
```

How to Display Table Structure

```
DESCRIBE tableName;
```

How to Display List of Tables

- SELECT * FROM tab;

CONSTRAINT

The rules that are imposed during data input to a table. Oracle engine check data constraint during data insertion, update and delete.

Column level constraint:
These types of constraints are created during table creation.

Table level constraint:
These constraints are created after all columns are created.

CONSTRAINT TYPES:

CONSTRAINT TYPES:

1. Null/Not null
2. Unique
3. Primary key
4. Foreign key
5. Check

Null/Not null Constraint:

Normally every column (without primary key) can accept null value. Null and 0 are not same. Null implies not applicable. However, if you want any column must accept a value, then declare it as not null.

Example 39: (Column level constraint)

```
CREATE TABLE student
(roll number, stName varchar2(20) NOT NULL, address varchar2(20));
```

Note: Not Null is used only at column level constraint as shown above example.

Unique Constraint:

Unique constraint is used to create a unique key. Unique constraint guarantees that no duplicate value is inserted.

Example 41: (Column level constraint)
CREATE TABLE student
(roll number UNIQUE, stName varchar2(20) NOT NULL, address varchar2(20));

Note: No duplicate value is accepted in roll column.

Example 41: (Table level constraint)
CREATE TABLE student
(roll number, stName varchar2(20) not null, address varchar2(20),UNIQUE(roll));

Note: No duplicate value is accepted in roll column.

Primary key Constraint:

- This type of constraint is used to create a relationship among different tables. Primary key of one table and foreign key of another table are used to create this relationship.

- Single column primary key is called simple primary key.
- Multiple column primary key is called composite primary key.

Primary key Constraint:

Example 42: (Column level constraint)
CREATE TABLE student
(roll number PRIMARY KEY, stName varchar2(20) NOT NULL, address varchar2(20));

Note: No duplicate value is accepted in roll column.

Example 43: (Table level constraint)
CREATE TABLE student
(roll number, stName varchar2(20) not null, address varchar2(20),PRIMARY KEY(name, address));

Note: No duplicate value is accepted in name, address column pair.

Foreign key Constraint:

This type of constraint is used to create a relationship among different tables. Primary key of one table and foreign key of another table are used to create this relationship. Foreign key table is called child table. Primary key table is called parent table.

- Rules of Foreign key Constraint:

1. If the value is not in master table then no insert/update operation can be performed on foreign table.
2. If you mention ON DELETE CASCADE option, then if you delete any record from master table, the corresponding record from foreign table will automatically deleted.
3. Primary key column and foreign key column must be same and of same data type.

Foreign key Constraint:

Column level syntax:
columnName dataType(size) REFERENCES
tableName[(columnName)]
[ON DELETE CASCADE]

Example 44: (Column level constraint)

```
CREATE TABLE student_details
(roll number REFERENCES student,
fatherName varchar2(20) not null,
hallName varchar2(20),
PRIMARY KEY(roll));
```

Note: It automatically reference to roll column of student table.

Foreign key Constraint:

Table level syntax:
FOREIGN KEY(columnName [, columnName])
REFERENCES tableName[(columnName)]

Example 45: (Table level constraint)

```
CREATE TABLE student_details
(roll number,
fatherName varchar2(20) not null,
hallName varchar2(20),
PRIMARY KEY(roll),
FOREIGN KEY(roll) REFERNCES student);
```

Note: It automatically reference to roll column of student table.

Check Constraint:

Check constraint is used to check value during input. If the check result is true, input is possible otherwise it is false.

- Some check constraint example: suppose
  1. We want that account no must be between 1 and 10000.
  2. We want that account holder name must start with A.
  3. We want that account balance can never be negative.

Check Constraint:

Example 46:

```
CREATE TABLE account
(aNumber number check(aNumber BETWEEN 1 and 10000),
aHolderName varchar2(20) check(aHolderName like 'A%'),
balance number check(balance>0),
PRIMARY KEY(aNumber));
```

We can also add the following logical expression:

1. Check(city IN ('Dhaka', 'Rajshahi','Natore','Bogra'))
2. Check(name=upper(name)
3. Etc