MySQL Constraints

The constraint in MySQL is used to specify the rule that allows or restricts what values/data will be stored in the table.

They provide a suitable method to ensure data accuracy and integrity inside the table. It also helps to limit the type of data that will be inserted inside the table.

Types of MySQL Constraints

Constraints in MySQL is classified into two types:

1. **Column Level Constraints:** These constraints are applied only to the single column that limits the type of particular column data.
2. **Table Level Constraints:** These constraints are applied to the entire table that limits the type of data for the whole table.

How to create constraints in MySQL

We can define the constraints during a table created by using the CREATE TABLE statement. MySQL also uses the ALTER TABLE statement to specify the constraints in the case of the existing table schema.

the following are the syntax to create a constraints in table:

Constraints used in MySQL

- **NOT NULL**
- CHECK
- DEFAULT
- PRIMARY KEY
- AUTO_INCREMENT
- UNIQUE
- ENUM
- FOREIGN KEY

```
CREATE TABLE Student (
   Id INTEGER,
   LastName TEXT NOT NULL,
   FirstName TEXT NOT NULL,
   City VARCHAR(35)
);
```

Execute the queries listed below to understand how it works:

1. mysql> **INSERT INTO** Student **VALUES**(1, 'Hanks', 'Peter', 'New York');

2. mysql> **INSERT INTO** Student **VALUES**(2, NULL, 'Amanda', 'Florida');

UNIQUE Constraint

This constraint ensures that all values inserted into the column will be unique. It means a column cannot stores duplicate values. MySQL allows us to use more than one column with UNIQUE constraint in a table.

CREATE TABLE ShirtBrands(Id **INTEGER**, BrandName **VARCHAR**(40) **UNIQUE**, **Size VARCHAR**(30));

INSERT INTO ShirtBrands(Id, BrandName, **Size**) **VALUES**(1, 'Pantaloons', 38), (2, 'Cantabil', 40);

INSERT INTO ShirtBrands(Id, BrandName, **Size**) **VALUES**(1, 'Raymond', 38), (2, 'Cantabil', 40);

CHECK Constraint

It controls the value in a particular column. It ensures that the inserted value in a column must be satisfied with the given condition. In other words, it determines whether the value associated with the column is valid or not with the given condition.

Before the version 8.0.16, MySQL uses the limited version of this constraint syntax, as given below:

CHECK (expr)

After the version 8.0.16, MySQL uses the CHECK constraints for all storage engines i.e., table constraint and column constraint, as given below:

Let us understand how a CHECK constraint works in MySQL. For example, the following statement creates a table "Persons" that contains CHECK constraint on the "Age" column. The CHECK constraint ensures that the inserted value in a column must be satisfied with the given condition means the Age of a person should be greater than or equal to 18:

```
CREATE TABLE Persons (
   ID int NOT NULL,
   Name varchar(45) NOT NULL,
   Age int CHECK (Age>=18)
);
```

Execute the listed queries to insert the values into the table:

```
mysql> INSERT INTO Persons(Id, Name, Age)
VALUES (1,'Robert', 28), (2, 'Joseph', 35), (3, 'Peter', 40);

mysql> INSERT INTO Persons(Id, Name, Age) VALUES (1,'Robert', 15);
```

DEFAULT Constraint

This constraint is used to set the default value for the particular column where we have not specified any value. It means the column must contain a value, including NULL.

For example, the following statement creates a table "Persons" that contains DEFAULT constraint on the "City" column. If we have not specified any value to the City column, it inserts the default value:

```
CREATE TABLE Persons (
   ID int NOT NULL,
   Name varchar(45) NOT NULL,
   Age int,
   City varchar(25) DEFAULT 'New York'
);
```

Execute the listed queries to insert the values into the table:

```
INSERT INTO Persons(Id, Name, Age, City)
VALUES (1,'Robert', 15, 'Florida'),
(2, 'Joseph', 35, 'California'),
(3, 'Peter', 40, 'Alaska');

INSERT INTO Persons(Id, Name, Age) VALUES (1,'Brayan', 15);


SELECT * FROM Persons;
```

PRIMARY KEY Constraint

This constraint is used to identify each record in a table uniquely. If the column contains primary key constraints, then it cannot be null or empty. A table may have duplicate columns, but it can contain only one primary key. It always contains unique value into a column.

The following statement creates a table "Person" and explains the use of this primary key more clearly:

```
CREATE TABLE Persons (
   ID int NOT NULL PRIMARY KEY,
   Name varchar(45) NOT NULL,
   Age int,
   City varchar(25));
```

Next, use the insert query to store data into a table:

```
INSERT INTO Persons(Id, Name, Age, City)
VALUES (1,'Robert', 15, 'Florida') ,
(2, 'Joseph', 35, 'California'),
(3, 'Peter', 40, 'Alaska');

INSERT INTO Persons(Id, Name, Age, City)
VALUES (1,'Stephen', 15, 'Florida');
```

AUTO_INCREMENT Constraint

This constraint automatically generates a unique number whenever we insert a new record into the table. Generally, we use this constraint for the primary key field in a table.

We can understand it with the following example where the id column going to be auto-incremented in the Animal table:

```
mysql> CREATE TABLE Animals(
id int NOT NULL AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (id));
```

Next, we need to insert the values into the "Animals" table:

```
mysql> INSERT INTO Animals (name) VALUES
('Tiger'),('Dog'),('Penguin'),
('Camel'),('Cat'),('Ostrich');
```

Now, execute the below statement to get the table data:

```
mysql> SELECT * FROM Animals;
```

ENUM Constraint

The ENUM data type in MySQL is a string object. It allows us to limit the value chosen from a list of permitted values in the column specification at the time of table creation. It is short for enumeration, which means that each column may have one of the specified possible values. It uses numeric indexes (1, 2, 3...) to represent string values.

The following illustration creates a table named "shirts" that contains three columns: id, name, and size. The column name "size" uses the ENUM data type that contains small, medium, large, and x-large sizes.

```
mysql> CREATE TABLE Shirts (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(35),
   size ENUM('small', 'medium', 'large', 'x-large')
);
```

Next, we need to insert the values into the "Shirts" table using the below statements:

```
mysql> INSERT INTO Shirts(id, name, size)
VALUES (1,'t-shirt', 'medium'),
(2, 'casual-shirt', 'small'),
(3, 'formal-shirt', 'large');
```

Now, execute the SELECT statement to see the inserted values into the table:

```
mysql> SELECT * FROM Shirts;
```

Foreign Key Constraint

This constraint is used to link two tables together. It is also known as the referencing key. A foreign key column matches the primary key field of another table. It means a foreign key field in one table refers to the primary key field of another table.

Let us consider the structure of these tables: Persons and Orders.

```
CREATE TABLE Persons (
   Person_ID int NOT NULL PRIMARY KEY,
   Name varchar(45) NOT NULL,
   Age int,
   City varchar(25)
);
```

```
CREATE TABLE Orders (
    Order_ID int NOT NULL PRIMARY KEY,
    Order_Num int NOT NULL,
    Person_ID int,
    FOREIGN KEY (Person_ID) REFERENCES Persons(Person_ID)
);
```

In the above table structures, we can see that the "Person_ID" field in the "Orders" table points to the "Person_ID" field in the "Persons" table. The "Person_ID" is the PRIMARY KEY in the "Persons" table, while the "Person_ID" column of the "Orders" table is a FOREIGN KEY.