UNIQUE Constraint in MySQL

When you want a column or columns not to accept any duplicate values, then you need to apply UNIQUE Constraint for that column or columns in MySQL. That means the UNIQUE Constraint is useful to restrict storing of duplicate data row values in a given column or combination of columns. But it accepts NULL values in that column.

Multiple columns in a single table can have the UNIQUE Constraint. We can apply the UNIQUE constraint on any data type column such as INT, VARCHAR, etc.

Create database EmployeeDB;

Use EmployeeDB;

CREATE TABLE Employee (

Id INT UNIQUE,

Name VARCHAR(50) NOT NULL,

Email VARCHAR(50) UNIQUE,

Department VARCHAR(50) NOT NULL

);

**DESCRIBE Employee;**

**INSERT INTO Employee (Id, Name, Email, Department) VALUES (1, ‘Anurag’, ‘Anurag@gamil.com’, ‘IT’);**

**INSERT INTO Employee (Id, Name, Email, Department) VALUES (1, ‘Sambit’, ‘Sambit@gmail.com’, ‘IT’);**

When we try to execute the above INSERT SQL statement, we will get the following error. This is because we are trying to insert duplicate Id values into the Id column.  
**Error Code: 1062. Duplicate entry ‘1’ for key ’employee.Id’**

Now, try to execute the below SQL Statement.  
**INSERT INTO Employee (Id, Name, Email, Department) VALUES (2, ‘Sambit’, ‘Anurag@gmail.com’, ‘IT’);**

Now, when you try to execute the above INSERT SQL statement, you should get the following error. This is because now we are trying to insert duplicate Email values into the Email column.  
**Error Code: 1062. Duplicate entry ‘Anurag@gamil.com’ for key ’employee.Email’**

When we try to insert a duplicate value (other than null) into a column on which the UNIQUE constraint is applied, then the SQL statement gets terminated by displaying an error message showing the table and column name. Now, execute the below INSERT Statement.  
**INSERT INTO Employee (Id, Name, Email, Department) VALUES (NULL, ‘Sambit’, NULL, ‘IT’);**

The above SQL statement will be executed successfully. This is because the UNIQUE Constraint allows null value.

**How to add UNIQUE Constraints to existing Columns in MySQL?**

It is also possible to add UNIQUE Constraints to existing columns in MySQL. Let us understand this with an example. First, delete all the data from the Employee table by executing the below TRUNCATE statement.  
**TRUNCATE Table Employee;**

In our Employee table, suppose we want to add UNIQUE Constraint on the Name column. Then we can do the same by executing the below SQL Statement.  
**ALTER TABLE Employee ADD UNIQUE (Name);**

Check Constraints in MySQL

**Check Constraints in MySQL**

The MySQL Check Constraint is used to enforce domain integrity. Domain integrity means the values that are going to store in a table column must be followed by some defined rules such as range, type, and format. In other words, we can say that Check Constraint ensures the valid entries for a given column value by restricting the type of the value, the format of the data, or the range of possible values.

The CHECK Constraint in MySQL can be applied to single or multiple columns of a table. The CHECK Constraint in MySQL can be defined while creating a new table or using ALTER TABLE statement for the already existing table.

The MySQL Check constraints can be created at two different levels

1. **Column-Level Check Constraints:** When we create the check constraints at the column level then they are applied only to that column of the table.
2. **Table-level Check Constraints:** When we create the check constraints at the table level, then it can be referred from any column(s) within that table.

A table can contain any number of check constraints and will apply to any column data type like integer, character, and decimal, date, etc.

**Column-Level Check Constraints in MySQL**

Let’s understand How to Create Column Level Check constraints with an example. Please have a look at the below CREATE Table Statement.

CREATE DATABASE EmployeeDB;

USE EmployeeDB;

CREATE TABLE Employees(

EmployeeID INT NOT NULL CHECK(EmployeeID BETWEEN 100 AND 1000),

NAME VARCHAR(50) NOT NULL,

AGE INT NOT NULL CHECK (AGE >= 18),

DeptID INT CHECK(DeptID > 0 AND DeptID < 100),

SALARY DECIMAL (18, 2)

);

In the above CREATE Table statement, we have applied three check constraints.

1. The First CHECK Statement is applied on the EmployeeId column which will only allow the values in the range between 100 and 1000.
2. The Second CHECK Constraint is applied on AGE Column which will only allow the values which are greater than equals to 18.
3. The third CHECK Constraint is applied on the DeptID column which will only allow the values if it is greater than 0 and less than 100.

Now, let us prove the above three statements. Please execute the below INSERT Statements.  
**INSERT INTO Employees values (100, ‘Pranaya’, 20, 1, 20000);**  
**INSERT INTO Employees values (101, ‘Rout’, 25, 2, 25000);**

When you execute the above two statements, two records are inserted into the Employees table. This is because the data we inserted satisfied all the Check Constraints. Now, try to execute the below INSERT Statement.

**INSERT INTO Employees values (10, ‘Anurag’, 20, 1, 20000);**  
When you try to execute the above SQL INSERT Statement, you will get the following error. This is because, here, we trying to INSERT an employee with the EmployeeId value 10 which does not satisfy the CHECK Constraint applied on the EmployeeId column.  
**Error Code: 3819. Check constraint ’employees\_chk\_1′ is violated.**

Now, try to execute the below INSERT Statement.  
**INSERT INTO Employees values (102, ‘Anurag’, 10, 1, 20000);**  
When you try to execute the above SQL INSERT Statement, you will get the following error. This is because, here, we trying to INSERT an employee with the AGE value 10 which does not satisfy the CHECK Constraint applied on the AGE column.  
**Error Code: 3819. Check constraint ’employees\_chk\_2′ is violated.**

Now, try to execute the below INSERT Statement.  
**INSERT INTO Employees values (102, ‘Anurag’, 20, 105, 20000);**  
When you try to execute the above SQL INSERT Statement, you will get the following error. This is because, here, we trying to INSERT an employee with the DeptID value 105 which does not satisfy the CHECK Constraint applied on the DeptID column.  
**Error Code: 3819. Check constraint ’employees\_chk\_3′ is violated.**

**How to DROP a CHECK Constraint in MySQL?**

Let us understand this with an example. Suppose, we want to drop the Check Constraint which is created on the Salary Column. To do so, you need to execute the below SQL Statement.

**ALTER TABLE Employees DROP CHECK CHK\_SALARY;**

Once you execute the above statement, now, try to execute the below INSERT Statement and it should be executed as expected as we remove the CHECK Constraint applied on the Salary column.

**INSERT INTO Employees values (104, ‘Priyanka’, 25, 10, 60000);**

**Table-level Check Constraints in MySQL**

It is also possible in MySQL to create the CHECK Constraint at the table level. When we create the check constraints at the table level, then that constraint can be referred from any column(s) within that table.

Let us understand this with an example. Please have a look at the below CREATE Table statement. Here, we have applied three CHECK Constraints at the table level on the PersonID, AGE, and DeptID columns.

CREATE TABLE Persons(

PersonID INT NOT NULL,

NAME VARCHAR(50) NOT NULL,

AGE INT NOT NULL ,

DeptID INT NOT NULL,

SALARY DECIMAL (18, 2),

CONSTRAINT CHK\_PersonID CHECK (PersonID BETWEEN 100 AND 1000),

CONSTRAINT CHK\_AGE CHECK (AGE >= 18),

CONSTRAINT CHK\_DeptID CHECK(DeptID > 0 AND DeptID < 100)

);

While imposing the Constraint at the table level, it is also possible to create the CHECK Constraints on multiple columns in MySQL. Let us understand this with an example. In the below CREATE TABLE statement, we applied the CHECK Constraint on the AGE and DeptID column.

CREATE TABLE Person(

PersonID INT NOT NULL,

NAME VARCHAR(50) NOT NULL,

AGE INT NOT NULL ,

DeptID INT NOT NULL,

SALARY DECIMAL (18, 2),

CONSTRAINT CHK\_AGE\_DepTID CHECK (AGE >= 18 AND DepTID > 0)

);

**Understanding the Primary Key Constraint in MySQL:**

CREATE TABLE Employee

(

Id INT PRIMARY KEY,

Name VARCHAR(500),

Email VARCHAR(50) PRIMARY KEY,

Department VARCHAR(50)

);

When you try to execute the above Create Table SQL query, you will get the following error. The error message clearly tells us that we are trying to create multiple primary keys which are not possible.  
**Error Code: 1068. Multiple primary key defined**

To overcome the above error, remove one primary key and execute the CREATE TABLE SQL statement as shown below.

CREATE TABLE Employee

(

Id INT PRIMARY KEY,

Name VARCHAR(500),

Email VARCHAR(50),

Department VARCHAR(50)

);

**DESCRIBE Employee;**

Let’s execute the following insert statement to insert a record into the Employee table  
**INSERT INTO Employee (Id, Name, Email, Department) VALUES (1, ‘Sambit’, ‘Sambit@gmail.com’, ‘IT’);**

**Adding Duplicate value in the Primary Key Column:**

Now let us see what happens when we try to insert a duplicate value into the Primary key column i.e. into the Id column. In our Employee table, one record exists with the Id 1. Now. Let us try to insert another employee with the same ID 1. Please try to execute the below insert statement to insert a duplicate record.  
**INSERT INTO Employee (Id, Name, Email, Department) VALUES (1, ‘Anurag’, ‘Anurag@gmail.com’, ‘IT’);**

When we try to execute the above SQL statement, it gives us the below error. That means the Primary Key constraint will not accept duplicate values in it.  
**Error Code: 1062. Duplicate entry ‘1’ for key ’employee.PRIMARY’**

**Inserting NULL in Primary Key Column in MySQL:**

Now, let us try to insert a NULL value into the primary key column i.e. Id column by executing the below insert statement.  
**INSERT INTO Employee (Id, Name, Email, Department) VALUES (NULL, ‘Anurag’, ‘Anurag@gmail.com’, ‘IT’);**

When we try to execute the above Insert SQL statement, it gives us the following error which clearly states that the Id value cannot be the null value which proves that Primary Key will not accept NULL.  
**Error Code: 1048. Column ‘Id’ cannot be null**

**Note:** So, the Primary key neither accepts NULL Values or Duplicate Values in MySQL.

As we already discussed except the NOT NULL constraint, all other constraints can be imposed either at the Table or Column level. Let us understand the same concept with PRIMARY KEY Constraint.

**What is a Foreign Key Constraint in MySQL?**

Creating the relationship between the database tables is one of the most important concepts in a database. The relationship between multiple tables provides a mechanism for linking the data stores in multiple tables and retrieving them in an efficient manner.

In order to create a link between two tables, we must specify a Foreign Key in one table that references a column in another table. That means the Foreign Key constraint in MySQL is used for binding two tables with each other and then verify the existence of one table data in other tables.

**Note:** A foreign key in one TABLE points to either a primary key or a unique key in another table in MySQL. The foreign key constraints are basically used to enforce referential integrity. In our upcoming articles, we will discuss referential integrity constraints in detail.

**How to Create Foreign Key Constraint in MySQL?**

To Create a Foreign Key Constraint in MySQL, we require the following things

1. We require two tables for linking with each other and those two tables must have a common column for binding the tables.
2. The common column that is present in both the tables need not have the same name but their data type must be the same.
3. The common column that is present under the parent table or master table is known as the reference key column and moreover, the reference key column should not contain any duplicate values. So, we need to impose either UNIQUE or PRIMARY key constraint on that column.
4. The common column which is present in the child or detailed table is known as the Foreign key column and we need to impose a Foreign key constraint on the column which refers to the reference key column of the master table.

If the above points are not clear at the moment, then don’t worry, we will try to explain all the above points with examples.

**Examples to understand FOREIGN KEY Constraints in MySQL.**

Let us understand how to create the primary key and foreign key relationship between two tables in MySQL. First, create a table with the name Department by using the PRIMARY KEY constraint by executing the below CREATE Table query. This table is going to be the parent table or master table which contains the reference key column. Here, we created the reference column (Id) using the Primary Key constraint.

CREATE DATABASE EmployeeDB;

USE EmployeeDB;

CREATE TABLE Department

(

Id INT PRIMARY KEY,

Name VARCHAR(50),

Location VARCHAR(50)

);

INSERT INTO Department (Id, Name, Location) VALUES (10, 'IT', 'Hyderabad');

INSERT INTO Department (Id, Name, Location) VALUES (20, 'HR', 'Delhi');

INSERT INTO Department (Id, Name, Location) VALUES (30, 'Finance', 'Mumbai');

CREATE TABLE Employee

(

ID INT PRIMARY KEY,

Name VARCHAR(30),

Salary INT,

DepartmentId INT,

FOREIGN KEY (DepartmentId) REFERENCES Department(Id)

);

INSERT into Employee VALUES (101, 'Anurag', 25000, 10);

INSERT into Employee VALUES (102, 'Pranaya', 32000, 20);

INSERT into Employee VALUES (103, 'Hina', 35000, 30);

**INSERT into Employee VALUES (104, ‘Sambit’, 52000, 40);**

When you try to execute the above INSERT SQL Statement, you will get the following error. It clearly says that you cannot add or update a child row if the foreign key value does not exist in the parent table. In this case, the foreign key value i.e. 40 that we want to insert in the employee table does not exist in the Department table and hence we get the error.

**Rules to Follow while working with Foreign Key in MySQL:**

When we impose Foreign Key constraint and establish the relation between tables in MySQL, the following 3 rules come into the picture  
**Rule1:** We cannot insert a value into the foreign key column if that value is not existing in the reference key column of the parent (master) table.  
**Example: INSERT into Employee VALUES (104, ‘Sambit’, 52000, 40);**  
When we try to execute the above INSERT Statement, you will get the following error.  
**Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`employeedb`.`employee`, CONSTRAINT `employee\_ibfk\_1` FOREIGN KEY (`DepartmentId`) REFERENCES `department` (`Id`))**

**Rule2:** We cannot update the reference key value of a parent table if that the value has a corresponding child record in the child table without addressing what to do with the child records.  
**Example: UPDATE Department SET Id = 100 WHERE Id = 10;**  
As we have a child record with department id 10 in the Employee table, so, trying to update the same in the Department table will give you the following error.  
**Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`employeedb`.`employee`, CONSTRAINT `employee\_ibfk\_1` FOREIGN KEY (`DepartmentId`) REFERENCES `department` (`Id`))**

**Rule3:** We cannot delete a record from the parent table provided that the records reference key value has a child record in the child table without addressing what to do with the child record.  
**Example: DELETE FROM Department WHERE Id = 10;**  
As we have a child record with department id 10 in the Employee table, so, trying to delete the same in the Department table will give you the following error.  
**Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`employeedb`.`employee`, CONSTRAINT `employee\_ibfk\_1` FOREIGN KEY (`DepartmentId`) REFERENCES `department` (`Id`))**

**How to add Foreign Key Constraints to Existing Table?**

To understand this, let us first, create a table without Foreign Key by executing the below SQL Statement.

CREATE TABLE Employee1

(

ID INT PRIMARY KEY,

Name VARCHAR(30),

Salary INT,

DepartmentId INT

);

Now, once the table is created, we want to add a Foreign Key on the DepartmentId column which should refer to the Id column of the Department table. You can do the same by using the ALTER TABLE statement as shown in the below example.

**ALTER TABLE Employee1 ADD FOREIGN KEY (DepartmentId) REFERENCES Department(Id);**

**How to Delete a Foreign Key Constraints in MySQL?**

MySQL allows the ALTER TABLE statement to remove an existing foreign key from the table. The following syntax is used to drop a foreign key in MySQL.

**ALTER TABLE table\_name DROP FOREIGN KEY fk\_constraint\_name;**

Here, the **table\_name** is the name of a table from where we are going to remove the foreign key. The **constraint\_name** is the name of the foreign key that was added during the creation or alteration of a table. If you don’t know the name of an existing foreign key, then execute the following command.

**SHOW CREATE TABLE Employee1;**

Once you execute the above query, it will give you the following output.

You will be getting errors so instead we can use below command

**ALTER TABLE Employee1 DROP FOREIGN KEY employee1\_ibfk\_1;**

**What is the difference between Primary Key and Foreign Key Constraint in MySQL?**

**Primary Key in MySQL:**

1. The Primary Key Constraint in MySQL is uniquely identifying a record in a table.
2. Primary Key constraint neither accepts null values nor duplicate values on the column on which it is applied.
3. We can create only one Primary Key on a table in MySQL and that primary key constraint can be created either on a single column or multiple columns.

**Foreign Key in MySQL:**

1. The Foreign Key in MySQL is a field in a table that is a unique key (either primary or unique key) in another table.
2. A Foreign Key can accept both null values and duplicate values.
3. We can create more than one Foreign key on a table in MySQL.

**Can we create a table with multiple unique, foreign, or primary keys in MySQL?**

We can create a table in MySQL with multiple unique and foreign keys. But it is not possible to create a table with multiple primary Keys.

**Is it possible that a foreign key references a non-primary key in MySQL?**

Yes, it is possible. The point that you need to keep in mind is that a foreign key actually references a key that should contain unique values. So, it may be a primary key or a unique key as both keys maintain the uniqueness of the column of a table.

**Can a foreign key accept null values in MySQL?**

Yes, a foreign key in MySQL can accept NULL values. This is because a Foreign key can reference unique or non-primary keys which may hold NULL values.