**Constraints in MySQL with Examples**

I am going to discuss **Constraints in MySQL** with Examples. At the end of this article, you will understand what are constraints, why do we need constraints, and the different types of constraints available in MySQL with Examples.

**What is Data Integrity in MySQL?**

While learning Constraints we need to understand one database term called Data Integrity. Data integrity means the data contained in the database is accurate, consistent, and reliable. To provide data integrity, RDBMS provides us a set of integrity constraints that ensures that the data entered into the database is going to be accurate, consistent, and reliable. This is the reason why the end-user can trust the data stored in the database.

**What are Constraints in MySQL?**

The MySQL Constraints define specific rules to the column(s) data in a database table. While inserting, updating, or deleting the data rows, if the rules of the constraint are not followed, the system will display an error message and the action will be terminated. The SQL Constraints are defined while creating a new table. We can also alter the table and add new SQL Constraints. The MySQL Constraints are mainly used to maintain data integrity.

**Why do we need Constraints?**

The Constraints in MySQL are basically used to restrict the insertion of unwanted data in the database. That means they are mainly used to maintain data integrity. We can create the constraint on single or multiple columns of a table in MySQL

**What are the different types of Constraints available in MySQL?**

MySQL supports the following six types of constraints for maintaining data integrity.

1. [**NOT NULL Constraint**](https://dotnettutorials.net/lesson/not-null-constraint-mysql/)
2. [**UNIQUE KEY Constraint**](https://dotnettutorials.net/lesson/unique-constraint-mysql/)
3. [**Default Constraint**](https://dotnettutorials.net/lesson/default-constraint-mysql/)
4. [**CHECK KEY Constraint**](https://dotnettutorials.net/lesson/check-constraints-in-mysql/)
5. [**PRIMARY KEY Constraint**](https://dotnettutorials.net/lesson/primary-key-constraint-mysql/)
6. [**FOREIGN KEY Constraint.**](https://dotnettutorials.net/lesson/foreign-key-constraint-mysql/)

Before understanding the constraints in MySQL, first, we need to understand NULL in MySQL.

**Understanding NULL in MySQL:**

NULL represents the absence of data or value in a column of a table in MySQL. It’s neither ZERO nor EMPTY. In order to understand this, let us create the following StudentDB database and Student table. Here, StudentId is an int data type and it is not null. On the other hand, FullName and RegdNo fields are NULL types means they accept a NULL value.

Create database StudentDB;

Use StudentDB;

CREATE TABLE Student (

StudentId INT NOT NULL,

FullName VARCHAR(50) NULL,

RegdNo INT NULL

);

Once you create the above StudentDB and Student table, let us add some data into the student table by executing the below SQL Script.

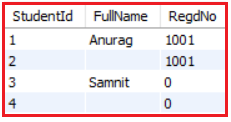
INSERT INTO Student(StudentId, FullName, RegdNo) VALUES(1, 'Anurag', '1001');

INSERT INTO Student(StudentId, FullName, RegdNo) VALUES(2, '', '1001');

INSERT INTO Student(StudentId, FullName, RegdNo) VALUES(3, 'Samnit', 0);

INSERT INTO Student(StudentId, FullName, RegdNo) VALUES(4, '', 0);

Now, if you issue a select query against the student table as **SELECT \* FROM Student;** then you will get the following result set.



As you can see in the above image, for StudentId 2, FullName is empty and for StudentId 3, the RegdNo column Value is 0. Further, for StudentId 4, the FullName is Empty as well as the RegdNo is 0. Here, zero (0) and empty do not represent NULL, you need to consider them as values.

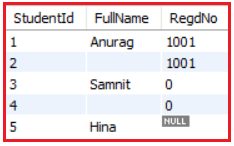
Whenever you don’t have any value, then you have to put it as Null. For example, let say we have one student whose RerdNo is not yet generated and supposed to come in later. Then, in that case, we can put a NULL in the RegdNo column. Please execute the below SQL Query which will insert one student with Null RegdNo.

**INSERT INTO Student (StudentId, FullName, RegdNo) VALUES (5, ‘Hina’, Null);**

Once you execute the above SQL Query, now execute the below SELECT Statement.

**SELECT \* FROM Student;**

Once you execute the above SELECT Query, you will get the following result set and you can see for the newly created student (StudentId 5) the RegdNo is null which is nothing but you can say the RegdNo is absent at the moment.



Now, I hope you understand NULL in MySQL.

**Imposing Constraint in MySQL:**

We can impose constraints on a table in two different ways

1. Imposing constraints on Column level
2. Imposing constraints on Table level

In the first case, we need to provide the constraint information next to the column name whereas in the second case we first define all the columns, and then we need to define the constraints for the columns.

**Note:** We cannot impose a NOT NULL constraint on table level. It is possible only for the rest of the four constraints.

**Imposing MySQL Constraint at Column Level:**

Let us understand this with an example. As you can see in the following Create Table statement, we have specified Primary Key, NOT NULL, and UNIQUE Constraints directly at the column level.

CREATE TABLE Employee

(

Id INT PRIMARY KEY,

NAME VARCHAR(30) NOT NULL,

Emailid VARCHAR(100) UNIQUE

);

**Imposing MySQL Constraint at Table Level:**

Let us understand how to impose constraint at table level with an example. As you can see in the following Create Table statement, first, we created all the columns and then we created constraints and while creating the constraint we need to specify the column name to which the constraint belongs.

CREATE TABLE Customer

(

Id INT,

NAME VARCHAR(30),

Email VARCHAR(100),

PRIMARY KEY(Id),

UNIQUE(Email)

);

**Note:** There is no difference in behavior whether we imposed the constraint at the table level or at the column level but if a constraint is imposed at the table level, then we have the advantage of imposing composite constraints.

**Creating Constraint with user-defined name in MySQL:**

Let us understand how to provide a user-defined name to the constraint’s examples. In order to understand this, please execute the below SQL Script which will create the Employee table. If you further notice, here we created two unique constraints by giving our own names such as **ID\_ Unique** and **Email\_Unique**.

**CREATE** **TABLE** Employee

(

Id **INT**,

**NAME** **VARCHAR**(30),

Email **VARCHAR**(100),

**CONSTRAINT** ID\_ **Unique** **UNIQUE** (Id),

**CONSTRAINT** Email\_Unique **UNIQUE**(Email)

);

**What are Composite Constraints in SQL Server?**

It is possible in MySQL to create a constraint-based on multiple columns and when we do so, it is called Composite Constraints. Let us understand the composite constraint with an example.

As you can see in the following Create table statement, we have imposed the Name\_Email\_Unique unique constraint at the table level and here we specified the Name and Email column. That means this Name\_Email\_Unique constraint is a composite constraint as it is created based on two columns (Name and Email).

**CREATE** **TABLE** EmployeeAddress

(

Id **INT**,

Name **VARCHAR**(50),

Email **VARCHAR**(10),

**CONSTRAINT** Name\_Email\_Unique **UNIQUE** (Name, Email)

);