SQL - Constraints

Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

The available constraints in SQL are:

- **NOT NULL**: This constraint tells us that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
- **UNIQUE:** This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.
- **PRIMARY KEY:** A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as the primary key.
- **FOREIGN KEY:** A Foreign key is a field which can uniquely identify each row in another table. And this constraint is used to specify a field as Foreign key.
- **CHECK:** This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
- **DEFAULT:** This constraint specifies a default value for the column when no value is specified by the user.

How to specify constraints?

We can specify constraints at the time of creating the table using the CREATE TABLE statement. We can also specify the constraints after creating a table using the ALTER TABLE statement.

Syntax :

```
CREATE TABLE table_name (
   column1 data_type constraint_name,
   column2 data_type constraint_name,
    .....
   columnN data_type constraint_name
);
```

1. NOT NULL :

If we specify a field in a table to be NOT NULL. Then the field will never accept null value. That is, you will not be allowed to insert a new row in the table without specifying any value to this field.

Example :

```
CREATE TABLE students (
   ID int NOT NULL,
   NAME varchar(10) NOT NULL,
   ADDRESS varchar(50)
);
```

2. UNIQUE :

This constraint helps to uniquely identify each row in the table. i.e. for a particular column, all the rows should have unique values. We can have more than one UNIQUE column in a table.

Example :

```
CREATE TABLE students (
  ID int NOT NULL UNIQUE,
  EMAIL varchar(60) NOT NULL UNIQUE,
  NAME varchar(10) NOT NULL,
);
```

3. PRIMARY KEY :

Primary Key is a field which uniquely identifies each row in the table. If a field in a table is the primary key, then the field will not be able to contain NULL values as well as all the rows should have unique values for this field. So, in other words we can say that this is a combination of NOT NULL and UNIQUE constraints.
A table can have only one field as primary key.

Example :

```
CREATE TABLE students (
  ID int NOT NULL UNIQUE,
  EMAIL varchar(60) NOT NULL UNIQUE,
  NAME varchar(10) NOT NULL,
  PRIMARY KEY (ID)
);
```

4. FOREIGN KEY :

Foreign Key is a field in a table which uniquely identifies each row of another table. That is, this field points to the primary key of another table. This usually creates a kind of link between the tables.

Example :

```
CREATE TABLE orders (
  order_id int NOT NULL,
  order_no int NOT NULL,
  customer_id int NOT NULL,
  PRIMARY KEY (order_id)
  FOREIGN KEY (cutomer_id) REFERENCE customer(customer_id)
 );
```

5. CHECK :

Using the CHECK constraint we can specify a condition for a field, which should be satisfied at the time of entering values for this field.

Example :

```
CREATE TABLE students (
  ID int NOT NULL,
  NAME varchar(10) NOT NULL,
  AGE int NOT NULL CHECK (AGE >=18),
);
```

6. DEFAULT :

This constraint is used to provide a default value for the fields. That is, if at the time of entering new records in the table if the user does not specify any value for these fields then the default value will be assigned to them.

Example :

```
CREATE TABLE students (
  ID int NOT NULL,
  NAME varchar(10) NOT NULL,
  AGE int DEFAULT 18,
);
```

7. INDEX

The INDEX constraints are created to speed up the data retrieval from the database. An Index can be created by using a single or group of columns in a table. A table can have a single PRIMARY KEY but can have multiple INDEX. An Index can be Unique or Non Unique based on requirements.

Example :

```
CREATE INDEX idx_age ON customer (AGE) ;
```