UNIT -II

RELATIONAL MODEL AND SQL

SQL Constraints

SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.

Constraints can be divided into the following two types,

1. **Column level constraints:** Limits only column data.
2. **Table level constraints:** Limits whole table data.

Constraints are used to make sure that the integrity of data is maintained in the database. Following are the most used constraints that can be applied to a table.

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

NOT NULL Constraint

By default, a column can hold NULL values. If you do not want a column to have a NULL value, use the NOT NULL constraint.

- It restricts a column from having a NULL value.
- We use ALTER statement and MODIFY statement to specify this constraint.

One important point to note about this constraint is that it cannot be defined at table level.

Example using NOT NULL constraint:

```
CREATE TABLE Student
```

```
(           s_id int NOT NULL,

            name varchar(60),

            age  int

);
```

The above query will declare that the **s_id** field of **Student** table will not take NULL value.

If you wish to alter the table after it has been created, then we can use the ALTER command for it:

```
ALTER TABLE Student

MODIFY s_id int NOT NULL;
```

UNIQUE Constraint

It ensures that a column will only have unique values. A UNIQUE constraint field cannot have any duplicate data.

- It prevents two records from having identical values in a column
- We use ALTER statement and MODIFY statement to specify this constraint.

Example of UNIQUE Constraint:

Here we have a simple CREATE query to create a table, which will have a column **s_id** with unique values.

```
CREATE TABLE Student

(           s_id int  NOT NULL,

            name varchar(60),

            age int  NOT NULL UNIQUE

);
```

The above query will declare that the **s_id** field of **Student** table will only have unique values and wont take NULL value.

If you wish to alter the table after it has been created, then we can use the ALTER command for it:

```
ALTER TABLE Student

MODIFY age INT NOT NULL UNIQUE;
```

The above query specifies that **s_id** field of **Student** table will only have unique value.

Primary Key Constraint

Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table.

PRIMARY KEY constraint at Table Level

```
CREATE table Student

(           s_id int PRIMARY KEY,

            Name varchar(60) NOT NULL,

            Age int);
```

The above command will creates a PRIMARY KEY on the s_id.

PRIMARY KEY constraint at Column Level

```
ALTER table Student

ADD PRIMARY KEY (s_id);
```

The above command will creates a PRIMARY KEY on the s_id.

Foreign Key Constraint

Foreign Key is used to relate two tables. The relationship between the two tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

- This is also called a referencing key.
- We use ALTER statement and ADD statement to specify this constraint.

To understand FOREIGN KEY, let's see its use, with help of the below tables:

c_id	Customer_Name	address
101	Adam	Noida
102	Alex	Delhi
103	Stuart	Rohtak

Customer_Detail Table

Order_Detail Table

Order_id	Order_Name	c_id
10	Order1	101
11	Order2	103
12	Order3	102

In **Customer_Detail** table, **c_id** is the primary key which is set as foreign key in **Order_Detail** table. The value that is entered in **c_id** which is set as foreign key in **Order_Detail** table must be present in **Customer_Detail** table where it is set as primary key. This prevents invalid data to be inserted into **c_id** column of **Order_Detail** table.

If you try to insert any incorrect data, DBMS will return error and will not allow you to insert the data.

FOREIGN KEY constraint at Table Level

CREATE table Order_Detail(order_id int PRIMARY KEY,   order_name varchar(60) NOT NULL,   c_id int FOREIGN KEY REFERENCES Customer_Detail(c_id));

In this query, **c_id** in table Order_Detail is made as foriegn key, which is a reference of **c_id** column in Customer_Detail table.

FOREIGN KEY constraint at Column Level

```
ALTER table Order_Detail

ADD FOREIGN KEY (c_id) REFERENCES Customer_Detail(c_id);
```

Behaviour of Foriegn Key Column on Delete

There are two ways to maintin the integrity of data in Child table, when a particular record is deleted in the main table. When two tables are connected with Foriegn key, and certain data in the main table is deleted, for which a record exits in the child table, then we must have some mechanism to save the integrity of data in the child table.

1. **On Delete Cascade :** This will remove the record from child table, if that value of foriegn key is deleted from the main table.
2. **On Delete Null :** This will set all the values in that record of child table as NULL, for which the value of foriegn key is deleted from the main table.
3. If we don't use any of the above, then we cannot delete data from the main table for which data in child table exists. We will get an error if we try to do so.

```
ERROR : Record in child table exist
```

CHECK Constraint

CHECK constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. Its like condition checking before saving data into a column.

Using CHECK constraint at Table Level

```
CREATE table Student(

   s_id int NOT NULL CHECK(s_id > 0),

   Name varchar(60) NOT NULL,

   Age int

);
```

The above query will restrict the **s_id** value to be greater than zero.

Using CHECK constraint at Column Level

```
ALTER table Student ADD CHECK(s_id > 0);
```