Lab 08: Constraints

Objective(s):

1. To learn an application of Constraints on a Table & its columns.

1: Introduction of 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

NOT NULL constraint restricts a column from having a `NULL` value. Once **NOT NULL** constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value.

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.

`UNIQUE` Constraint

UNIQUE constraint ensures that a field or column will only have unique values. A **UNIQUE** constraint field will not have duplicate data. This constraint can be applied at column level or table level.

Using `UNIQUE` constraint when creating a Table (Table Level)

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 UNIQUE, Name varchar(60), Age int);
```

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

Using `UNIQUE` constraint after Table is created (Column Level)

```
ALTER TABLE Student ADD UNIQUE(s_id);
```

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.

Using 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`.

Using 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`.

For multiple columns:

```
PRIMARY KEY (s_id, home_city)
```

Foreign Key Constraint

FOREIGN KEY is used to relate two tables. FOREIGN KEY constraint is also used to restrict actions that would destroy links between tables. To understand FOREIGN KEY, let's see its use, with help of the below tables:

Customer_Detail Table

cid	CustomerName	address
101	Nida	Sukkur
102	Rameez	Karachi
103	Fabiha	Lahore

Order Table

Orderid	OrderName	cid
10	Order1	101
11	Order2	103
12	Order3	102

In **Customer_Detail** table, **cid** is the primary key which is set as foreign key in **Order** table. The value that is entered in **cid** which is set as foreign key in **Order** table must be present in

Customer_Detail table where it is set as primary key. This prevents invalid data to be inserted into **cid** column of **Order** table.

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

Using FOREIGN KEY constraint at Table Level

```
CREATE table Order(
    Orderid int PRIMARY KEY,
    OrderName varchar(60) NOT NULL,
    cid int(11),
    FOREIGN KEY(cid) REFERENCES Customer_Detail(cid)
);
```

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

Using FOREIGN KEY constraint at Column Level

```
ALTER table Order_Detail ADD FOREIGN KEY (cid) REFERENCES
Customer_Detail(cid);
```

For multiple columns:

```
FOREIGN KEY(ord_no,book_id) REFERENCES  neworder(ord_no,book_id),
```

Behavior of Foreign Key Column on Delete/Update

When two tables are connected with Foreign key, and certain data in the main table is deleted/updated, for which a record exists in the child table, then we must have some mechanism to save the integrity of data in the child table.

MySQL allows creating a table with CASCADE, SET NULL and RESTRICT options.

CASCADE option deletes or updates the row from the parent table (containing PRIMARY KEYs), and automatically delete or update the matching rows in the child table (containing FOREIGN KEYs).

RESTRICT option bars the removal (i.e. using delete) or modification (i..e using an update) of rows from the parent table.

SET NULL option will delete or update the row from the parent table, and set the foreign key column or columns in the child table to NULL.

You can use SET NULL for DELETE as well as UPDATE.

If SET NULL is used, you should not set NOT NULL options to the columns of the child table (containing FOREIGN KEYS).

```
CREATE table Order_Details(
    Orderid int PRIMARY KEY,
    OrderName varchar(60) NOT NULL,
    id int(11),
    FOREIGN KEY(id)  REFERENCES customer(id)
    ON UPDATE CASCADE ON DELETE RESTRICT
);

CREATE table Order_Details(
    Orderid int PRIMARY KEY,
    OrderName varchar(60) NOT NULL,
    id int(11),
    FOREIGN KEY(id)  REFERENCES customer(id)
    ON UPDATE CASCADE ON DELETE SET NULL
);
```

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. It's 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);
```

CHECK CONSTRAINT using IN operator

MySQL CHECK CONSTRAINT can be applied to a column of a table, to set a limit for storing values within a range, along with IN operator.

```
CREATE TABLE customer(
        id INT(11) NOT NULL PRIMARY KEY,
        name VARCHAR(30),
        phone INT(15),
        country varchar(25) NOT NULL CHECK(country IN('UK', 'USA'))
);
```

You can use CHECK CONSTRAINT with LIKE, OR & AND operator and also with CASE statement.

AUTO_INCREMENT Constraint

MySQL allows you to set AUTO_INCREMENT to a column. Doing so will increase the value of that column by 1 automatically, each time a new record is added.

```
ALTER TABLE customer MODIFY id int AUTO_INCREMENT;
```

The above command will modify the id column of the customer table, to an auto increment.

Lab Task(s):

Exercise

1. Create a table DEPARTMENT with the following attributes:

   DEPTNO number, DNAME varchar(10), LOC varchar(10).

   PRIMARY KEY constraint on DEPTNO.

2. Create a table EMPLOYEE with the following attributes:

   EMPNO number, ENAME varchar(10),SAL number, DEPTNO number.

   Apply FOREIGN KEY constraint on DEPTNO referencing the DEPARTMENT table created in question 1 and PRIMARY KEY constraint on EMPNO and DEPTNO.

3. ALTER table EMPLOYEE created in question 2 and apply the constraint CHECK on

   ENAME attribute such that ENAME should always be inserted in capital letters.

4. ALTER table DEPARTMENT created in question 1 and apply constraint on DNAME such that DNAME should not be entered empty.

5. ALTER table EMPLOYEE created in question 2 and apply the constraint on SAL attribute such that no two salaries of the employees should be similar.

6. ALTER table EMPLOYEE created in question 2 and apply the constraint on DEPTNO attribute such that on update, update a child value and on delete set null value to a child.

END