DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2003 Database Systems
Spring 2023

LAB 04 Constraints

OBJECTIVE(S)

- Learn about constraints

CONSTRAINTS

Constraints are rules to which data must conform. They are used to ensure database accuracy and reliability. There are two types of constraints:

- **Column level** constraints apply to a single column (usually the one which they follow).
- **Table level** constraints apply to the whole table. They specify the names of tables to which they apply.

Adding Constraints

The general syntax for applying a constraint (unless otherwise specified) is:

- **CREATE TABLE** tb_name(
  col1_name *datatype* ***constraint***,
  col2_name *datatype* ***constraint***,
  col3_name *datatype* ***constraint***);

In the case where we want to add a constraint to an already created table, we use the following syntaxes:

- **ALTER TABLE** tb_name
  ADD col_name *datatype* ***constraint***;

- **ALTER TABLE** tb_name
  ADD ***constraint***(col_name(s));

- **ALTER TABLE** tb_name
  ADD CONSTRAINT constraint_name ***constraint***(col_name(s));

- **ALTER TABLE** tb_name
  CHANGE old_col_name new_col_name *datatype* ***constraint***;

- **ALTER TABLE** tb_name
  MODIFY col_name *datatype* ***constraint***;

- **ALTER TABLE** tb_name
  ADD CHECK condition;

Some common SQL constraints are listed henceforth.

FUNCTION NAME	DESCRIPTION
NOT NULL	Ensures that a column cannot have a NULL value. This can only be added *after* the datatype of the column. **To add this constraint to an existing column, first ensure that the column has no NULL values.**
UNIQUE	Ensures that all values in a column are different. It can be either a column level or a table-level constraint.
PRIMARY KEY	Used to uniquely identify a record, it is a combination of **NOT NULL** and **UNIQUE**. Primary keys may contain single or multiple fields. *A table can have only one primary key.*
COMPOSITE KEY	It is a type of **key** which is formed by more than one column. It is a combination of two or more two columns in a table that allows us to identify each row of the table uniquely. MySQL guaranteed the uniqueness of the column only when they are combined. If they have taken individually, the uniqueness cannot maintain.
FOREIGN KEY	Used to uniquely identify a record in another table. The foreign key, like the primary key, may contain single or multiple fields which form the primary key of another table. When defining a foreign key, we also have to include the field it references in the other table. The table containing the foreign key is called the child table. The table containing the respective primary key is called the parent/referenced table.
CHECK	It is used to limit the value range that can be placed in a column. If applied to a single column, it allows only certain values for that column. If applied to a table, it can limit the values in certain columns based on the values in another column for a particular record.
DEFAULT	Sets default values for a column *if* no other value is specified. The syntax is: • **CREATE TABLE** tb_name(col1_name **DEFAULT** value, col2_name **DEFAULT** value col3_name); • **ALTER TABLE** tb_name **ALTER** col_name **SET DEFAULT** value;

Removing Constraints

- To remove any named constraint, we use the following syntax:
  ALTER TABLE tb_name **DROP CONSTRAINT** constraint_name;

- To remove an unnamed **UNIQUE** constraint, the following syntaxes can be used:
  DROP INDEX col_name **ON** tb_name;
  ALTER TABLE tb_name **DROP INDEX** col_name;

- The syntax for removing the primary key is:
  ALTER TABLE tb_name **DROP PRIMARY KEY**;

- The syntax for removing default is:
  ALTER TABLE tb_name **ALTER** col_name **DROP DEFAULT**;

- The syntax for allowing **NULL** values is:
  ALTER TABLE tb_name **MODIFY** col_name *datatype*;

TASK

- Set the student ID and name as the composite primary key. Show the structure of the table.
- Remove the primary key. Show the structure of the table.
- Set the student ID to be the primary key.
- Change the student name and semester fields to ensure that their values are never **NULL**.
- Set any default value for the department field. Show the structure of the table.
- Show the structure of the table.

AUTO INCREMENT

Auto-increment is used to automatically generate a unique number every time a new record is inserted into a table. Usually, we set the primary key to auto increment. Once a field has been set to increment automatically, we no longer have to provide a value for it.

- **CREATE TABLE** tb_name(
  col_name **AUTO_INCREMENT**);

- **ALTER TABLE** tb_name **MODIFY** col_name *datatype* **AUTO_INCREMENT**;

By default, the value for **AUTO_INCREMENT** is 1 but we can change it so that it starts at an arbitrary value.

- **ALTER TABLE** tb_name **AUTO_INCREMENT** = value;

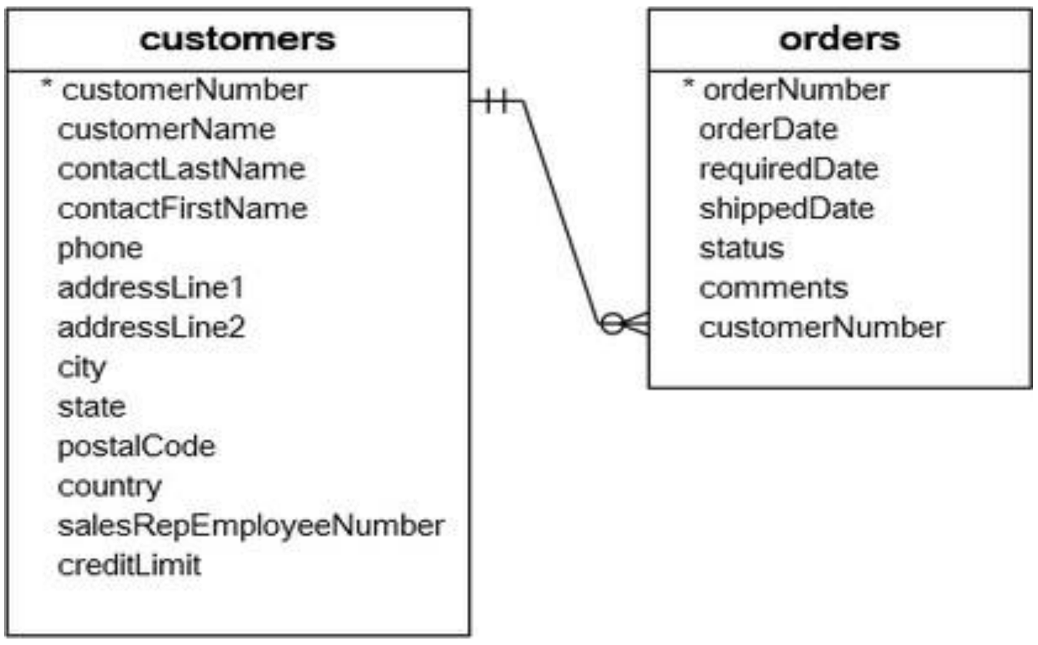
- The syntax for removing auto-increment is:
  ALTER TABLE tb_name **MODIFY** col_name *datatype*;

TASK

- Set the student ID to auto increment. Display the structure of the table.
- Create a new table with at least two columns. Select one of the keys to be the primary key and set it to auto increment.
- Insert three records. Display them.
- **DELETE** records from the table.
- Insert three more records. Display them.
- **TRUNCATE** the table.
- Insert three more records. Display them.
- Remove auto-increment from the table.

FOREIGN KEY

A foreign key is a field(s) in the table which is the primary key in another table. Foreign keys are used to link tables together. For instance, in the tables below, customerNumber is the *primary key* in table **customers** and *foreign key* in table **orders**.



To add a foreign key, we use,

- **CREATE TABLE** tb_name(

    …
    FOREIGN KEY (col_name) **REFERENCES** tb2_name(col_name)
        [ON UPDATE *option***]**
        [ON DELETE *option***]**
    );

- **ALTER TABLE** tb_name
  ADD FOREIGN KEY (col_name) **REFERENCES** tb2_name(col_name)
  [ON UPDATE *option***]**
  [ON DELETE *option***]**

The following deletes a foreign key constraint.

- **ALTER TABLE** tb_name **DROP FOREIGN KEY** constraint_name;

Note: To find the name of a foreign key constraint (in case we have not named it), use the following query:

```
SELECT Column_Name, Constraint_Name
    FROM Information_Schema.Key_Column_Usage
    WHERE Table_Name = “tb_name”;
```

Options with Foreign Keys

Since foreign keys are primary keys in another table, they need to be updated/deleted each time the primary key is updated/deleted in the parent table. In order to do that, we can set the following option on a foreign key.

- **ON DELETE** *option*
- **ON UPDATE** *option*

OPTION	DESCRIPTION
CASCADE	The values of the foreign key in the child table will be updated or the referenced record will be deleted automatically.
RESTRICT	This rejects any **DELETE** or **UPDATE** operation on the primary key of the parent table. This is the default option.
SET NULL	Sets NULL value of the foreign key of the referenced record in the child table.
NO ACTION	No action is taken when any **DELETE** or **UPDATE** operation is performed on the primary key of the parent table.

TASK

- Create a new table ***Courses*** having course ID (AB123) and course name.
- Create a new table ***Department*** having department ID and department name.
- Insert two new fields in ***Students*** course ID and department ID. Set as foreign keys without **ON DELETE/UPDATE**
- Insert relevant values in both tables.
- Delete any course from the ***Courses***.
- Delete the foreign key on course ID.
- Reset the foreign key on course ID. Set the option to **SET NULL**.
- Delete a course from ***Courses***.
- Delete any department from ***Department***.
- Delete the foreign key on course ID.
- Reset the foreign key on course ID. Set the option to **CASCADE**.
- Delete a department from ***Department***.

LAB ASSIGNMENT

1. Create the following tables and show their structures:
   Employee(employeeID, firstName, lastName, hiringDate, deptID, jobID),
   Department(deptID, deptName, locationID),
   Job(jobID, jobTitle, salary),
   Location(locationID, city, country).
   Job IDs should have the format "ENG123" for an engineer, "MAN456" for a manager etc.
   It is not necessary that all employees may be assigned a department or that a department may have employees.
   It is also not necessary that each designation may be filled.
2. Ensure that all numerical IDs increment automatically. Show the relevant table structures.
3. Set the location field to hold the value "Karachi, Pakistan" unless otherwise specified. Show the relevant table structures.
4. Insert at least 4 relevant records in each table. Display all the data.
5. Ensure that employee names, department names and job titles are never empty. Show the relevant table structures.
6. Delete any location and ensure that all departments at that location have their locations set to null. Show all relevant queries, table structures, and data.

SUBMISSION GUIDELINES

- Take a screenshot of each task. Ensure that all screenshots have a white background and black text. You can alter the background and text colors through the properties of the MySQL command line client.
- Place all the screenshots in a single word file labeled with Roll No and Lab No. e.g. **'cs181xxx_Lab04'**
- Convert the file into PDF.
- Submit the PDF file at LMS
- **-100%** policies for plagiarism.