# Department of Computer Science

**CS302: Database Systems**

**Class: BSCS**

**Lab 04: DDL and Constraints**

# Instructor: Mr. Hamza Javed

# CLO-1: Create a database schema that incorporates keys and integrity constraints.

# 

**Lab 04: DDL and Constraints**

**Introduction**

This lab will focus on the DDL (Data Definition Language) part of SQL. It discusses how to create relations and discusses different types of constraints defined in SQL. Constraints are specific set of rules defined for SQL objects. In this lab you will implement constraints and then observe how a constraint ensures that valid data is inserted into the relations.

**Objectives**

After completing this lab, you should be able to do the following:

* Create relations in a database
* Create constraints including NOT NULL, Unique Key Constraint, Primary Key, Foreign Key
* Insert valid data into tables
* Update & delete data from table
* Check constraint violations

**Tools/Software Requirement**

* MySQL Workbench 6.1 or Latest

**Description**

**Creating Tables**

1. **The CREATE TABLE Statement**

First of all, create a new schema/database or use an existing one to create tables in it. Create tables/relations to store data by executing the SQL CREATE TABLE statement. This statement is one of the data definition languages (DDL) statements. DDL statements are a subset of SQL statements used to create, modify, or remove database structures. These statements have an immediate effect on the database, and they also record information in the data dictionary.

To create a table, a user must have the CREATE TABLE privilege and a storage area in which to create objects. The syntax to create a table is as follows:

**CREATE TABLE [*schema*.]*table***

**(*column* *datatype* [DEFAULT *expr*]**

**[*column\_constraint*],**

**...**

**[*table\_constraint*][,...]);**

In the syntax:

schema is the database in which to create table

table is the name of the table

DEFAULT expr specifies a default value if a value is omitted in the INSERT statement

Column is the name of the column

datatype is the column’s data type and length

column constraint is the constraint to apply on table

**Example:**

**Create a table employees and departments.**

CREATE TABLE employees(

employee\_id INT,

first\_name VARCHAR(20),

job\_id INT NOT NULL,

dept\_id INT,

CONSTRAINT emp\_emp\_id\_pk

PRIMARY KEY (EMPLOYEE\_ID),

CONSTRAINT emp\_dept\_id\_fk

FOREIGN KEY (dept\_id) references departments (department\_id) );

CREATE TABLE departments(

department\_id INT,

department\_name VARCHAR(30),

location VARCHAR(30),

job\_id INT,

CONSTRAINT dept\_dept\_id\_pk

PRIMARY KEY (department\_ID));

Notice the sequence of creating the tables. Due to the referential integrity constraint, the sequence of execution of table creation queries of employees and departments is important.

1. **AUTO-INCREMENT values**

In case you want a column to auto\_increment (mostly ID columns), you can use the following syntax:

CREATE TABLE departments(

department\_id INT AUTO\_INCREMENT,

department\_name VARCHAR(30),

location VARCHAR(30),

job\_id INT,

CONSTRAINT dept\_dept\_id\_pk

PRIMARY KEY (department\_ID));

1. **DEFAULT value**

For a value of column which is not explicitly supplied while inserting values into a table, default value can be used. The default value for a column is specified at the time of creation of table.

CREATE TABLE departments(

department\_id INT,

department\_name VARCHAR(30),

location VARCHAR(30),

job\_id INT default 10,

CONSTRAINT dept\_dept\_id\_pk

PRIMARY KEY (department\_ID));

1. **Actions taken when DELETE or UPDATE operation is performed**

When a referential integrity constraint violation occurs in case of delete or update operation, you can specify one the operations as a response.

* Restrict
* Set NULL, SET DEFAULT
* Cascade (Propagate the change)

CREATE TABLE employees(

employee\_id INT,

first\_name VARCHAR(20),

job\_id INT NOT NULL,

dept\_id INT,

CONSTRAINT emp\_emp\_id\_pk

PRIMARY KEY (EMPLOYEE\_ID),

CONSTRAINT emp\_dept\_id\_fk

FOREIGN KEY (dept\_id) references departments (department\_id)

ON DELETE SET NULL

ON UPDATE CASCADE);

**Inserting Values**

The syntax to insert values into a table is as follows:

INSERT INTO employees

Values(2,'John',10,2);

There is an implicit and explicit method to add values to a table.

* Implicit method: Omit the column from the column list.

INSERT INTO employees (employee\_id, first\_name)

Values(2,'John');

* Specify the NULL keyword in the VALUES clause.

INSERT INTO employees

Values(2,'John',NULL, NULL);

* Adding multiple rows at a time:

INSERT INTO employees

Values(2,'John',10, NULL),(3,'Smith',12, NULL);

**Deleting Values**

The syntax to delete values from a table is as follows:

DELETE FROM table\_name

[WHERE condition];

**Updating Values**

The syntax to insert values into a table is as follows:

UPDATE *table*

SET *column* = *value* [, *column* = *value, ...*]

[WHERE *condition*];

**Constraints Table**

All constraints are stored in relation known as table\_constraint. This relation is stored in the system schema “Information\_schema” and can be extracted using the following query.

select \*

from information\_schema.table\_constraints;

**Dropping a Constraint**

To drop a constraint, you can identify the constraint name from the TABLE\_CONSTRAINTS. Then use the ALTER TABLE statement with the DROP clause.

**Syntax**

ALTER TABLE *table*

DROP CONSTRAINT *constraint*;

In the syntax:

*table* is the name of the table

*constraint*  is the name of the constraint

Example:

**ALTER TABLE employees**

**DROP CONSTRAINT emp\_manager\_fk;**