ASSIGNMENT NO. 6

Aim

Create table with primary key and foreign key constraint.
a Alter table with add and modify b. Drop Table

Objective

Understand and implement DDL statements.

Theory

MySQL uses many different data types broken into three categories: numeric, date and time, and string types.

Numeric Data Types:

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you. The following list shows the common numeric data types and their descriptions:

- **INT** - A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- **TINYINT** - A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT** - A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT** - A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to

8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.

- **BIGINT** - A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
- **FLOAT(M,D)** - A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D)** - A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
- **DECIMAL(M,D)** - An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types:

The MySQL date and time datatypes are:

- **DATE** - A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** - A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** - A timestamp between midnight, January 1, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
- **TIME** - Stores the time in HH:MM:SS format.
- **YEAR(M)** - Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069

(70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

String Types:

Although numeric and date types are fun, most data you'll store will be in string format. This list describes the common string datatypes in MySQL.

- **CHAR(M)** - A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** - A variable-length string between 1 and 255 characters in length; for example VARCHAR(25). You must define a length when creating a VARCHAR field.
- **BLOB or TEXT** - A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.
- **TINYBLOB or TINYTEXT** - A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT** - A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** - A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

Table

CREATE TABLE table_name (column_name column_type);

drop table table_name;

alter table table_name [add|modify|drop]

Index

A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.

While creating index, it should be considered that what are the columns which will be used to make SQL queries and create one or more indexes on those columns.

Practically, indexes are also type of tables, which keep primary key or index field and a pointer to each record into the actual table.

The users cannot see the indexes, they are just used to speed up queries and will be used by Database Search Engine to locate records very fast.

INSERT and UPDATE statements take more time on tables having indexes where as SELECT statements become fast on those tables. The reason is that while doing insert or update, database need to insert or update index values as well.

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX *index_name*
[*index_type*]
ON *tbl_name* (*index_col_name*,...)
[*index_type*]

index_col_name:
col_name [(*length*)] [ASC | DESC]

index_type:
USING {BTREE | HASH}

Sequence

In MySQL, a sequence is a list of integers generated in the ascending order i.e., 1,2,3… Many applications need sequences to generate unique numbers mainly for identification e.g., customer ID in CRM, employee number in HR, equipment number in services management system, etc.

To create a sequence in MySQL automatically, you set the AUTO_INCREMENT attribute to a column, which typically is a primary key column. The following rules you must follow when you use AUTO_INCREMENT attribute:

- Each table has only one AUTO_INCREMENT column whose data type is typically integer
- The AUTO_INCREMENT column must be indexed, which means it can be either PRIMARY KEY or UNIQUE index.
- The AUTO_INCREMENT column must have NOT NULL constraint. When you set AUTO_INCREMENT attribute to a column, MySQL will make it NOT NULL for you in case you don't define it explicitly.

Start MySQL

mysql -u root -p

DCL

create database mydb;

use mydb;

create user user1@localhost identified by 'test123';

grant all on mydb to user1@localhost;

revoke all on mydb from user1@localhost;

DDL

Multinational Company Manages information of Employees.Each Employee has unique ID.

Each Employee works on project in some department.

Design ER Diagram for Employee Management System

Create tables for above.

Employee(EmpID, EName, Address, Contactno, DOB, DOJ, Salary, DID, PId)

Department(DID, DName)

Project(PId, PName, LocationId)

Location(LocationId, LocationCity)

1 use mydb;

2 create table Department(did int primary key, dname varchar(20));

3 create table Employee(empid int primary key auto_increment, ename varchar(20) not null, contactno int unique, salary int, did int, check (salary>15000), foreign key(did) references Department(did));

4. drop table employee;

5. create table Project(Pid int primary key, Pname varchar(20), LocationId int);

6. alter table Employee add Pid int;

7. desc Employee;

8. alter table Employee add foreign key(Pid) references Project(Pid);

9. alter table Project drop LocationId;

10 alter table Project modify Pname varchar(50);

Example of composite key

11. create table Emp(empid int, name varchar(20), primary key(empid, name));