ASSIGNMENT NO.02

TITLE: Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym

PROBLEM STATEMENT: To design and develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence and Synonym

OBJECTIVES:

- To list and understand DDL commands and SQL Objects
- Use database techniques such as SQL DDL statements

OUTCOMES:

- Students will able to create SQL objects using different DDL commands
- Students will be able to write queries for given requirements using different DDL commands

SOFTWARE & HARDWARE REQUIREMENTS:

1. 64-bit Open source Linux or its derivative
2. MySQL Server

THEORY:

SQL (Structured Query Language) is a standardized programming language used for managing relational databases and performing various operations on the data in them.

All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as −

- MS SQL Server using T-SQL,
- Oracle using PL/SQL,
- MS Access version of SQL is called JET SQL (native format) etc.

The uses of SQL include modifying database table and index structures; adding, updating and deleting rows of data; and retrieving subsets of information from within a database for transaction processing and analytics applications.

SQL language is divided into four types of primary language statements: DML, DDL, DCL and TCL. Using these statements, we can define the structure of a database by creating and altering database objects, and we can manipulate data in a table through updates or deletions. We also can control which user can read/write data or manage transactions to create a single unit of work.

The four main categories of SQL statements are as follows:

1. DML (Data Manipulation Language)
2. DDL (Data Definition Language)
3. DCL (Data Control Language)
4. TCL (Transaction Control Language)

DDL (Data Definition Language)

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

CREATE – Create database and its objects like table, index, views, stored procedures, functions and triggers
ALTER – Alter structure of existing database, table and column description
DROP – Delete existing objects from database
TRUNCATE – Remove all records from table including all spaces allocated for the records are removed
RENAME – Rename an object

DML (Data Manipulation Language)
DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

DML statements include the following:

SELECT – Select records from a table
INSERT – Insert new records
UPDATE – Update/Modify existing records
DELETE – Delete existing records

DCL (Data Control Language)

DCL statements control the level of access that users have on database objects.
GRANT – Allows users to read/write on certain database objects
REVOKE – Keeps users from read/write permission on database objects

TCL (Transaction Control Language)
TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.
BEGIN– Opens a transaction

COMMIT– Commits a transaction
ROLLBACK– ROLLBACK a transaction in case of any error
SAVEPOINT – To rollback the transaction to a certain point in transaction

DDL COMMANDS

CREATE COMMAND

Creating a Database: To create a database in RDBMS, create command is used. Following is the Syntax,

```
create database database-name;
```

Example for Creating Database
mysql>create database Test;
The above command will create a database named Test.

Creating a Table: Create command is also used to create a table. We can specify names and datatypes of various columns along. Following is the Syntax

```
create table table-name
{
 column-name1 datatype1,
 column-name2 datatype2,
 column-name3 datatype3,
 column-name4 datatype4
};
```

Create table command will tell the database system to create a new table with given table name and column information.
Example for creating Table
mysql> create table Student(id int, name varchar, age int);
The above command will create a new table Student in database system with 3 columns, namely id, name and age.

ALTER COMMAND

Alter command is used for alteration of table structures. There are various uses of alter command, such as,

- to add a column to existing table
- to rename any existing column
- to change datatype of any column or to modify its size.
- alter is also used to drop a column.

To Add Column to existing Table

Using alter command we can add a column to an existing table. Following is the Syntax,

alter table ***table-name*** add(***column-name datatype***);

Here is an Example for this,

mysql> alter table Student add(address char);

The above command will add a new column address to the Student table

To Modify an existing Column

Alter command is used to modify data type of an existing column . Following is the Syntax,

alter table ***table-name*** modify(***column-name datatype***);

Here is an Example for this,

mysql> alter table Student modify(address varchar(30));

The above command will modify address column of the Student table

To Rename a column

Using alter command you can rename an existing column. Following is the Syntax,

alter table ***table-name*** rename ***old-column-name*** to ***column-name***;

Here is an Example for this,

mysql>alter table Student rename address to Location;

The above command will rename address column to Location.

To Drop a Column

Alter command is also used to drop columns also. Following is the Syntax,

alter table ***table-name*** drop(***column-name***);

Here is an Example for this,

mysql> alter table Student drop(address);

The above command will drop address column from the Student table

TRUNCATE COMMAND

Truncate command removes all records from a table. But this command will not destroy the table's structure. When we apply truncate command on a table its Primary key is initialized. Following is its Syntax,

truncate table ***table-name;***

Here is an Example explaining it.

mysql> truncate table Student;

The above query will delete all the records of Student table.

Truncate command is different from delete command. delete command will delete all the rows from a table whereas truncate command re-initializes a table(like a newly created table).

For eg. If you have a table with 10 rows and an auto_increment primary key, if you use delete command to delete all the rows, it will delete all the rows, but will not initialize the primary

key, hence if you will insert any row after using delete command, the auto_increment primary key will start from 11. But in case of truncate command, primary key is re-initialized.

DROP COMMAND

Drop query completely removes a table from database. This command will also destroy the table structure. Following is its Syntax

```
drop table table-name;
```

Here is an Example explaining it.

mysql> drop table Student;

The above query will delete the Student table completely. It can also be used on Databases. For Example, to drop a database,

mysql> drop database Test;

The above query will drop a database named Test from the system.

RENAME COMMAND

Rename command is used to rename a table. Following is its Syntax,

```
rename table old-table-name to new-table-name
```

Here is an Example explaining it.

mysql> rename table Student to Student-record;

The above query will rename Student table to Student-record.

SQL VIEWS

A view in SQL is a logical subset of data from one or more tables. View is used to restrict data access.

Syntax for creating a View,

```
CREATE or REPLACE view view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

Consider following Sale table,

oid	order_name	previous_balance	customer
11	ord1	2000	Alex
12	ord2	1000	Abhi
13	Ord3	3000	Adam
14	Ord4	2500	Alex
15	Ord5	3000	Abhi

SQL Query to Create View

mysql> CREATE or REPLACE view sale_view as select * from Sale where customer = 'Alex';

The data fetched from select statement will be stored in another object called sale_view. We can use create seperately and replace too but using both together works better.

Example of Displaying a View

Syntax of displaying a view is similar to fetching data from table using Select statement.

mysql> SELECT * from sale_view;

UPDATE A VIEW

Update command for view is same as for tables.

Syntax to Update a View is,

```
UPDATE view-name
set value
WHERE condition;
```

If we update a view it also updates base table data automatically.

TYPES OF VIEW

There are two types of view,

- Simple View
- Complex View
-

Simple View	Complex View
Created from one table	Created from one or more table
Does not contain functions	Contain functions
Does not contain groups of data	Contains groups of data

SQL INDEX

An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view. An index contains keys built from one or more columns in the table or view. These keys are stored in a structure (B-tree) that enables SQL Server to find the row or rows associated with the key values quickly and efficiently.

When there are thousands of records in a table, retrieving information will take a long time. Therefore indexes are created on columns which are accessed frequently, so that the information can be retrieved quickly. Indexes can be created on a single column or a group of columns. When a index is created, it first sorts the data and then it assigns a ROWID for each row.

Syntax to create Index:

```
CREATE INDEX index_name ON table_name (column_name1,column_name2...);
```

Syntax to create SQL unique Index:

```
CREATE UNIQUE INDEX index_name ON table_name (column_name1,column_name2...);
```

If you want to view the indexes defined on a particular table, you can use the following command.

```
SHOW INDEXES FROM table_name;
```

Drop index

The drop command is used to remove already defined indexes on a table.

```
DROP INDEX `index_id` ON `table_name`;
```

SQL SEQUENCE

Sequence is a feature supported by some database systems to produce unique values on demand. Some DBMS like MySQL supports **AUTO_INCREMENT** in place of Sequence. AUTO_INCREMENT is applied on columns, it automatically increments the column value by 1 each time a new record is entered into the table. Sequence is also somewhat similar to AUTO_INCREMENT but its has some extra features.

Creating Sequence

Syntax to create sequences is,

```
CREATE Sequence sequence-name
start with initial-value
increment by increment-value
maxvalue maximum-value
cycle|nocycle
```

initial-value specifies the starting value of the Sequence, increment-value is the value by which sequence will be incremented and **maxvalue** specifies the maximum value until which sequence will increment itself. **cycle** specifies that if the maximum value exceeds the set limit, sequence will restart its cycle from the beginning. **No cycle** specifies that if sequence exceeds **maxvalue** an error will be thrown.

Example to create Sequence

```
mysql> CREATE Sequence seq_1
        start with 1
        increment by 1
        maxvalue 999
        cycle ;
```

Example to use Sequence

The class table

ID	Name
1	Abhi
2	Adam
4	Alex

The sql query will be,

mysql> INSERT into class value(seq_1.nextval,'anu');

Result table will look like,

ID	Name
1	Abhi
2	Adam
4	Alex
1	Anu

Accessing Sequence Numbers

To generate Sequence Numbers you can use NEXTVAL and CURRVAL for example to get the next sequence number of bills sequence type the following command.

mysql> Select bills.nextval from dual;

BILLS
1

NEXTVAL gives the next number in sequence. Whereas, CURRVAL returns the current number of the sequence. This is very handy in situations where you have insert records in Master Detail tables. For example to insert a record in SALES master table and SALES_DETAILS detail table.

```
mysql> insert into sales (billno,custname,amt)
        values (bills.nextval,'Sami',2300);

mysql> insert into sales_details (billno,itemname,qty,rate) values
        (bills.currval,'Onida',10,13400);
```

ALTERING SEQUENCES

To alter sequences use ALTER SEQUENCE statement. For example to alter the bill sequence MAXVALUE give the following command.

```
mysql> ALTER SEQUENCE BILLS
        MAXVALUE 200;
```

Except Starting Value, you can alter any other parameter of a sequence. To change START WITH parameter you have to drop and recreate the sequence.

DROPPING SEQUENCES

To drop sequences use DROP SEQUENCE command. For example to drop bills sequence give the following statement

mysql> drop sequence bills;

SQL SYNONYMS:

A synonym is an alias for a table, view, snapshot, sequence, procedure, function, or package.

There are two types to SYNONYMS they are

- PUBLIC SYNONYM
- PRIVATE SYNONYM

If you a create a synonym as public then it can be accessed by any other user with qualifying the synonym name i.e. the user doesn't have to mention the owner name while accessing the synonym. Nevertheless the other user should have proper privilege to access the synonym. Private synonyms needs to be qualified with owner names.

CREATING SYNONYMS

To create a private synonym in your own schema, you must have the CREATE SYNONYM privilege; to create a private synonym in another user's schema, you must have the CREATE ANY SYNONYM privilege. To create a public synonym, you must have the CREATE PUBLIC SYNONYM system privilege.

Create a synonym using the SQL command CREATE SYNONYM. For example, the following statement creates a public synonym named PUBLIC_EMP on the EMP table contained in the schema of I2IT:

sql>CREATE PUBLIC SYNONYM public_emp FOR I2IT.emp;

DROPPING SYNONYMS

You can drop any private synonym in your own schema. To drop a private synonym in another user's schema, you must have the DROP ANY SYNONYM system privilege. To drop a public synonym, you must have the DROP PUBLIC SYNONYM system privilege.

Drop a synonym that is no longer required using the SQL command DROP SYNONYM. To drop a private synonym, omit the PUBLIC keyword; to drop a public synonym, include the PUBLIC keyword.

For example, the following statement drops the private synonym named EMP:

sql>DROP SYNONYM emp;

The following statement drops the public synonym named PUBLIC_EMP:

sql>DROP PUBLIC SYNONYM public_emp;

CONCLUSION:

QUESTIONS:

1. Create following tables using given schema and insert appropriate data into these tables.
Student(StudID, Name, Address, Marks)
Employee(EmplyeeID, Name, Address, Salary, DateOfJoining ,Department)
Weather(CityID, CityName, MinTemp, MaxTemp)

2. Alter Student and Employee table to add Not Null constraint on all columns.

3. Alter the Student table to add Primary key constraint on StudID column.

4. Create a view JoiningInfo on Employee table displaying Employee ID, Name and DateOfJoining of employees.

5. Crete index on primary key columns of all the tables.

6. Crate view MarksInfo on Student table displaying StuID and Marks.

7. Change the name of Weather table to WeatherData.

8. Drop column CityName from WeatherData table.

9. Add column Grade to Student table.

10. Crate a view "DistinctionStudents" on student table displaying data of students having Distinction as Grade.

11. Create a sequence on StudID in student table.

12. Create a synonym 'Emp_Info' for Employee table.