Experiment No. : 2

Title: Implementation of DDL statement to create table, alter table and Drop Table.

Objectives:

1. To learn the basics of Structured Query Language (SQL).
2. To understand and use data definition language (DDL) to write query for a database.

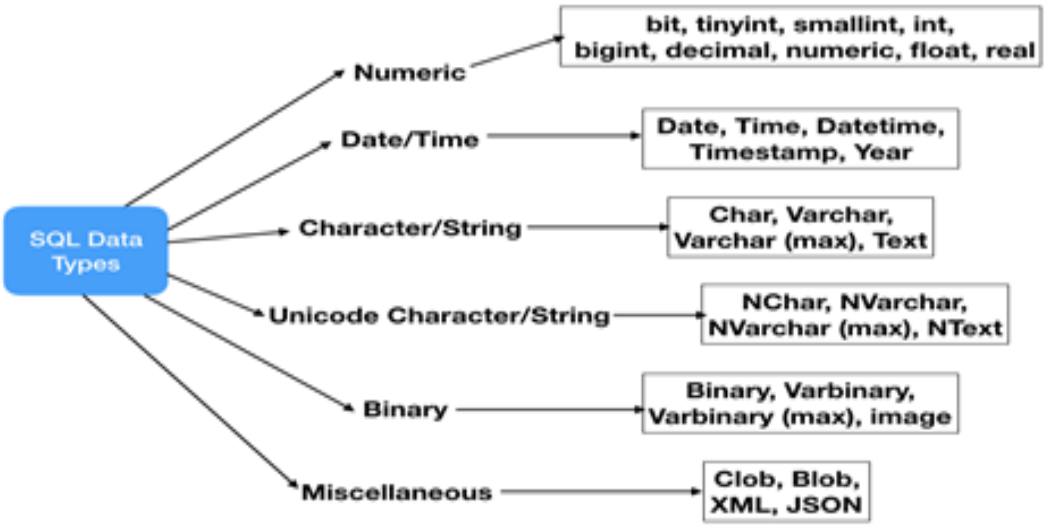
Key Concepts: SQL Data Definition Language (DDL),

Theory:

Structured Query Language (SQL) is a common query language through which we can interact with the database. SQL Statements can be divided into four main categories

1. **Data definition Language (DDL) Statements -** used to define the database structure or schema.
2. **Data manipulation Language (DML) Statements -** used for managing data within schema objects
3. **Data Control Language (DCL) statements –** used for Security and authorization
4. **Transaction Control (TCL) statements** - used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

DATA TYPES:



Data definition Language (DDL):

The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project. Let's take a look at the structure and usage of basic DDL commands:

1. **CREATE -** This is used to create a new relation (table)
   Syntax: CREATE TABLE <relation_name/table_name > (field_1 data_type(size),field_2 data_type(size), .. .);
   Example:
   CREATE TABLE Student (sno NUMBER (3), sname CHAR (10), class CHAR (5));

2. **ALTER** - alters the structure of the database
   (a)ALTER TABLE ...ADD...: This is used to add some extra fields into existing relation.
   Syntax: ALTER TABLE relation_name ADD (new field_1 data_type(size), new field_2 data_type(size),..);
   Example: ALTER TABLE std ADD (Address CHAR(10));

   (b)ALTER TABLE...MODIFY...: This is used to change the width as well as data type of fields of existing relations.
   Syntax: ALTER TABLE relation_name MODIFY (field_1 newdata_type(Size), field_2 newdata_type(Size),....field_newdata_type(Size));
   Example:
   ALTER TABLE student MODIFY(sname VARCHAR(10),class VARCHAR(5));

   c) ALTER TABLE..DROP...: This is used to remove any field of existing relations.
   Syntax: ALTER TABLE relation_name DROP COLUMN (field_name);
   Example: ALTER TABLE student DROP column (sname);

   d)ALTER TABLE..RENAME...: This is used to change the name of fields in existing relations.
   Syntax: ALTER TABLE relation_name RENAME COLUMN (OLD field_name) to (NEW field_name);
   Example: ALTER TABLE student RENAME COLUMN sname to stu_name;

3. **DROP** - This is used to delete the structure of a relation. It permanently deletes the records in the table.
   Syntax: DROP TABLE relation_name;
   Example: DROP TABLE std;

4. **TRUNCATE -** remove all records from a table, including all spaces allocated for the records are removed
   Syntax: TRUNCATE TABLE <table name>;
   Example: TRUNCATE TABLE Employee;

5. **RENAME** - It is used to modify the name of the existing database object.
   Syntax: RENAME TABLE old_relation_name TO new_relation_name;
   Example: RENAME TABLE std TO std1;

Constraints:

Constraints are used to specify rules for the data in a table. If there is any violation between the constraint and the data action, the action is aborted by the constraint. It can be specified when the table is created (using CREATE TABLE statement) or after the table is created (using ALTER TABLE statement).

1. NOT NULL

When a column is defined as NOTNULL, then that column becomes a mandatory column. It implies that a value must be entered into the column if the record is to be accepted for storage in the table.

Syntax: CREATE TABLE Table_Name (column_name data_type (size) NOT NULL,);

Example: CREATE TABLE student (sno NUMBER(3)NOT NULL, name CHAR(10));

2. UNIQUE

The purpose of a unique key is to ensure that information in the column(s) is unique i.e. a value entered in column(s) defined in the unique constraint must not be repeated across the column(s). A table may have many unique keys

Syntax: CREATE TABLE Table_Name(column_name data_type(size) UNIQUE, ….);

Example: CREATE TABLE student (sno NUMBER(3) UNIQUE, name CHAR(10));

3. CHECK

Specifies a condition that each row in the table must satisfy. To satisfy the constraint, each row in the table must make the condition either TRUE or unknown (due to a null).

Syntax: CREATE TABLE Table_Name(column_name data_type(size) CHECK(logical expression), ….);

Example: CREATE TABLE student (sno NUMBER (3), name CHAR(10),class CHAR(5),CHECK(class IN('CSE','CAD','VLSI'));

4. PRIMARY KEY

A field which is used to identify a record uniquely. A column or combination of columns can be created as primary key, which can be used as a reference from other tables. A table contains primary key is known as Master Table.

- It must uniquely identify each record in a table.
- It must contain unique values.
- It cannot be a null field.
- It should contain a minimum no. of fields necessary to be called unique

Syntax: CREATE TABLE Table_Name(column_name data_type(size) PRIMARY KEY, ….);

Example: CREATE TABLE faculty (fcode NUMBER(3) PRIMARY KEY, fname CHAR(10));

5. FOREIGN KEY

Columns defined as foreign key refer the primary key of other tables. The foreign key points to a primary key of another table, guaranteeing that, you can't enter data into a table unless the referenced table has the data already present. It enforces the referential integrity.

Syntax: CREATE TABLE Table_Name(column_name data_type(size) 39 FOREIGN KEY(column_name) REFERENCES table_name);

Example: CREATE TABLE subject (scode NUMBER (3) PRIMARY KEY, subname CHAR(10),fcode NUMBER(3), FOREIGN KEY(fcode) REFERENCE faculty);

6. DEFAULT

The DEFAULT constraint is used to insert a default value into a column. The default value will be added to all new records, if no other value is specified.

Syntax: CREATE TABLE Table_Name(col_name1,col_name2,col_name3 DEFAULT ‘’);
Example: CREATE TABLE student (sno NUMBER(3) UNIQUE, name CHAR(10),address VARCHAR(20) DEFAULT ‘Aurangabad’);

Defining integrity constraints in the alter table command:
Syntax: ALTER TABLE Table_Name ADD constraint_name;
Example: ALTER TABLE student ADD PRIMARY KEY (sno);
Dropping integrity constraints in the alter table command:
Syntax: ALTER TABLE Table_Name DROP constraint_name;
Example: ALTER TABLE student DROP PRIMARY KEY;

LAB WORK:

Q1. Create following tables for bank database with specific constraint

Branch (branchname, branchcity, asset)
- city and asset should not null, asset greater than 0

Customer (custname, custstreet, custcity)
- city not null

Account (accnum, branchname, balance)
- balance greater than 500

Loan (loanno, branchname, amount)
- amount greater than 0

Create a relation borrower which is between customer and loan
Create a relation Depositor which is between customer and accnum

Perform following:

1. Add Constraint street not null
2. Add a column phoneNo to customer table.
3. Add column custage to customer table
4. Add Constraint age greater than 18
5. Change the size of the phoneNo to varchar(10).
6. Drop the column phoneNo from customer table.
7. Remove the constraint on age.

LAB PRACTICE ASSIGNMENT:

1. Create a table EMPLOYEE with following schema: (Emp_no, E_name, E_address, E_ph_no, Dept_no, Dept_name,Job_id , Salary)
2. Add a new column; HIREDATE to the existing relation.
3. Change the datatype of JOB_ID from char to varchar2.
4. Change the name of column/field Emp_no to E_no.
5. Modify the column width of the job field of emp table

2. Create a table called EMP with the following structure. Name Type ---------- ---------- -- EMPNO NUMBER (6) ENAME VARCHAR2 (20) JOB VARCHAR2 (10) DEPTNO NUMBER (3) SAL NUMBER (7,2) Allow NULL for all columns except ename and job.
2. Add constraints to check, while entering the empno value (i.e) empno > 100.
3. Define the field DEPTNO as unique.
4. Create a primary key constraint for the table(EMPNO).
5. Write queries to implement and practice constraints.