LAB OBJECTIVE

Upon successful completion of this Lab the student will be able to:

- Creating database objects
- Modifying database objects
- Manipulating the data
- Retrieving the data from the database server
- Performing database operations in a procedural manner using pl/sql
- Performing database operations (create, update, modify, retrieve, etc..) using front-end tools l
- Design and Develop applications like banking, reservation system, etc..

AFTER COMPLETION OF THIS LAB WE WILL BE ABLE TO CREATE SOME PROJECTS TO ILLUSTRATE THE WORK OF DBMS

Introduction

A **database-management system (DBMS)** is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the **database**, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*.

SQL is structure query language that enables to create and operate on relational database, which are sets of information stored in a table.

The SQL language has several parts:

•**Data-definition language** (DDL). The SQL DDL provides commands for definingrelation schemas, deleting relations, and modifying relation schemas.

•**Data-manipulation language** (DML). The SQL DML provides the ability toquery information from the database and to insert tuples into, delete tuplesfrom, and modify tuples in the database.

•**Integrity** .The SQL DDL includes commands for specifying integrity constraintsthat the data stored in the database must satisfy. Updates that violateintegrity constraints are disallowed.

•**View definition**. The SQL DDL includes commands for defining views.

•**Transaction control**. SQL includes commands for specifying the beginningand ending of transactions.

•**Embedded SQL** and **dynamic SQL**. Embedded and dynamic SQL define howSQL statements can be embedded within general-purpose programming languages,such as C, C++, and Java.

•**Authorization**. The SQL DDL includes commands for specifying access rightsto relations and views.

SQL contains different data types those are

•**char** (*n*): A fixed-length character string with user-specified length *n*. The fullform, **character**, can be used instead.

•**varchar**(*n*): A variable-length character string with user-specified maximumlength *n*. The full form, **character varying, is** equivalent.

•**int**: An integer (a finite subset of the integers that is machine dependent). Thefull form, **integer**, is equivalent.

•**smallint**: A small integer (a machine-dependent subset of the integer type).

·**numeric**(p, d): A fixed-point number with user-specified precision. The numberconsists of p digits (plus a sign), and d of the p digits are to the right ofthe decimal point. Thus, **numeric**(3,1) allows 44.5tobestoredexactly,butneither 444.5or0.32 can be stored exactly in a field of this type.

·**real, double precision**: Floating-point and double-precision floating-pointnumbers with machine-dependent precision.

·**float**(n): A floating-point number, with precision of at least n digits.

Different types of commands in SQL:

A).**DDL commands:** - To create a database objects

B).**DML commands:** - To manipulate data of a database objects

C).**DQL command:** - To retrieve the data from a database.

D).**DCL/DTL commands:** - To control the data of a database…

In SQL, to answer the every query, we need three types of clauses, the **select** clause, the **from**clause, and the **where** clause. The role of each clause is as follows:

- The **select** clause is used to list the attributes desired in the result of a query.
- The **from**clause is a list of the relations to be accessed in the evaluation ofthe query.
- The **where** clause is a predicate involving attributes of the relation in the**from**clause.

1. DATA DEFINITION LANGUAGE (DDL) COMMANDS IN DBMS

AIM:

To execute and verify the Data Definition Language commands and constraints

DDL (DATA DEFINITION LANGUAGE)

- CREATE
- ALTER
- DROP
- TRUNCATE
- RENAME
- DESC

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Execute different Commands and extract information from the table.

STEP 4: Stop

SQL COMMANDS

1. COMMAND NAME: **CREATE**
   COMMAND DESCRIPTION: **CREATE** command is used to create objects in the database.
2. COMMAND NAME: **ALTER**
   COMMAND DESCRIPTION: **ALTER** command is used to alter the structure of database.
3. COMMAND NAME: **DROP**
   COMMAND DESCRIPTION: **DROP** command is used to delete the object from the database.
4. COMMAND NAME: **TRUNCATE**
   COMMAND DESCRIPTION: **TRUNCATE** command is used to remove all the records from thetable.

5. COMMAND NAME: **RENAME**
COMMAND DESCRIPTION: **RENAME** command is used to rename the objects.

6. COMMAND NAME: **DESC**
COMMAND DESCRIPTION: **DESC** is used to view the structure of the table.

QUERY: 01

Q1. Write a query to create a table employee with empno, ename, designation, and salary.

Syntax for creating a table:

SQL: CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1<DATATYPE> (SIZE), COLUMN NAME.1 <DATATYPE> (SIZE) .....);

QUERY: 01

SQL>CREATE TABLE EMP (EMPNO NUMBER (4), ENAME VARCHAR2 (10), DESIGNATIN VARCHAR2 (10), SALARY NUMBER (8,2));

Table created.

QUERY: 02

Q2. Write a query to display the column name and datatype of the table employee.

Syntax for describe the table:

SQL: DESC <TABLE NAME>;

QUERY: 02

SQL> DESC EMP;

Name	Null?	Type
EMPNO		NUMBER (4)
ENAME		VARCHAR2 (10)
DESIGNATIN		VARCHAR2 (10)
SALARY		NUMBER (8,2)

QUERY: 03

Q3. Write a query for create a from an existing table with all the fields

Syntax For Create A from an Existing Table With All Fields

SQL> CREATE TABLE <TRAGET TABLE NAME> SELECT * FROM<SOURCE TABLE NAME>;

QUERY: 03

SQL> CREATE TABLE EMP1 AS SELECT * FROM EMP;

Table created.

SQL> DESC EMP1

Name	Null?	Type
EMPNO		**NUMBER (4)**
ENAME		**VARCHAR2 (10)**
DESIGNATION		**VARCHAR2 (10)**
SALARY		**NUMBER (8,2)**

QUERY: 04

Q4. Write a query for create a from an existing table with selected fields

Syntax For Create A from an Existing Table With Selected Fields

SQL> CREATE TABLE <TRAGET TABLE NAME> SELECT EMPNO, ENAME FROM <SOURCE TABLE NAME>;

QUERY: 04

SQL> CREATE TABLE EMP2 AS SELECT EMPNO, ENAME FROM EMP;

Table created.

SQL> DESC EMP2

Name	Null?	Type
EMPNO		**NUMBER (4)**
ENAME		**VARCHAR2 (10)**

QUERY: 05

Q5. Write a query for create a new table from an existing table without any record:

Syntax for create a new table from an existing table without any record:

SQL> CREATE TABLE <TRAGET TABLE NAME> AS SELECT * FROM<SOURCE TABLE NAME> WHERE<FALSE CONDITION>;

QUERY: 05

SQL> CREATE TABLE EMP3 AS SELECT * FROM EMP WHERE 1>2;

Table created.

SQL> DESC EMP3;

Name	Null?	Type
EMPNO		**NUMBER (4)**
ENAME		**VARCHAR2 (10)**
DESIGNATION		**VARCHAR2 (10)**
SALARY		**NUMBER (8,2);**

ALTER & MODIFICATION ON TABLE

QUERY: 06

Q6. Write a Query to Alter the column EMPNO NUMBER (4) TO EMPNO NUMBER (6).

Syntax for Alter & Modify on a Single Column:

SQL > ALTER <TABLE NAME> MODIFY <COLUMN NAME><DATATYPE> (SIZE);

QUERY: 06

SQL>ALTER TABLE EMP MODIFY EMPNO NUMBER (6);

Table altered.

SQL> DESC EMP;

Name	Null?	Type
EMPNO		**NUMBER (6)**
ENAME		**VARCHAR2 (10)**
DESIGNATION		**VARCHAR2 (10)**
SALARY		**NUMBER(8,2)**

QUERY: 07

Q7. Write a Query to Alter the table employee with multiple columns (EMPNO, ENAME.)

Syntax for alter table with multiple column:

SQL > ALTER <TABLE NAME> MODIFY <COLUMN NAME1><DATATYPE>(SIZE), MODIFY <COLUMN NAME2><DATATYPE>(SIZE).....;

QUERY: 07

SQL>ALTER TABLE EMP MODIFY (EMPNO NUMBER (7), ENAME VARCHAR2(12));

Table altered.

SQL> DESC EMP;

Name	Null?	Type
EMPNO		**NUMBER (7)**
ENAME		**VARCHAR2 (12)**
DESIGNATION		**VARCHAR2 (10)**
SALARY		**NUMBER (8, 2);**

QUERY: 08

Q8. Write a query to add a new column in to employee

Syntax for add a new column:

SQL> ALTER TABLE <TABLE NAME> ADD (<COLUMN NAME><DATA TYPE><SIZE>);

QUERY: 08

SQL> ALTER TABLE EMP ADD QUALIFICATION VARCHAR2 (6);

Table altered.

SQL> DESC EMP;

Name	Null?	Type
EMPNO		**NUMBER (7)**
ENAME		**VARCHAR2 (12)**
DESIGNATION		**VARCHAR2 (10)**
SALARY		**NUMBER (8, 2)**
QUALIFICATION		**VARCHAR2 (6)**

QUERY: 09

Q9. Write a query to add multiple columns in to employee

Syntax for add a new column:

SQL> ALTER TABLE <TABLE NAME> ADD (<COLUMN NAME1><DATA TYPE><SIZE>,(<COLUMN NAME2><DATA TYPE><SIZE>.....);

QUERY: 09

SQL>ALTER TABLE EMP ADD (DOB DATE, DOJ DATE);

Table altered.

SQL> DESC EMP;

Name	Null?	Type
EMPNO		**NUMBER(7)**
ENAME		**VARCHAR2(12)**
DESIGNATION		**VARCHAR2(10)**
SALARY		**NUMBER(8,2)**
QUALIFICATION		**VARCHAR2(6)**
DOB		**DATE**
DOJ		**DATE**

REMOVE / DROP

QUERY: 10

Q10. Write a query to drop a column from an existing table employee

Syntax for add a new column:

SQL> ALTER TABLE <TABLE NAME> DROP COLUMN <COLUMN NAME>;

QUERY: 10

SQL> ALTER TABLE EMP DROP COLUMN DOJ;

Table altered.

SQL> DESC EMP;

Name	Null?	Type
EMPNO		**NUMBER (7)**
ENAME		**VARCHAR2 (12)**
DESIGNATION		**VARCHAR2 (10)**
SALARY		**NUMBER (8, 2)**
QUALIFICATION		**VARCHAR2 (6)**
DOB		**DATE**

QUERY: 11

Q10. Write a query to drop multiple columns from employee

Syntax for add a new column:

SQL> ALTER TABLE <TABLE NAME> DROP <COLUMN NAME1>,<COLUMN NAME2>,…… ;

QUERY: 11

SQL> ALTER TABLE EMP DROP (DOB, QUALIFICATION);

Table altered.

SQL> DESC EMP;

Name	Null?	Type
EMPNO		NUMBER(7)
ENAME		VARCHAR2(12)
DESIGNATION		VARCHAR2(10)
SALARY		NUMBER(8,2)

RENAME

QUERY: 12

Q12. Write a query to rename table emp to employee

Syntax for add a new column:

SQL> ALTER TABLE RENAME <OLD NAME> TO <NEW NAME>

QUERY: 12

SQL> ALTER TABLE EMP RENAME EMP TO EMPLOYEE;

Rename emp to emp33;

Table altered.

SQL> DESC EMPLOYEE;

Name	Null?	Type
EMPNO		NUMBER(7)
ENAME		VARCHAR2(12)
DESIGNATION		VARCHAR2(10)
SALARY		NUMBER(8,2)

CONSTRAINTS

Constraints are part of the table definition that limits and restriction on the value entered into its columns.

TYPES OF CONSTRAINTS:
1) Primary key
2) Foreign key/references
3) Check
4) Unique
5) Not null
6) Null
7) Default

CONSTRAINTS CAN BE CREATED IN THREE WAYS:
1) Column level constraints
2) Table level constraints
3) Using DDL statements-alter table command

OPERATION ON CONSTRAINT:
i) ENABLE
ii) DISABLE
iii) DROP

A **superkey** of a relation is a set of one or more attributes whose values are guaranteed to identify tuples in the relation uniquely. A candidate key is a minimal superkey, that is, a set of attributes that forms a superkey, but noneof whose subsets is a superkey. One of the candidate keys of a relation is chosen as its **primary key**.

A **foreign key** is a set of attributes in a referencing relation, such that for each tuple in the referencing relation, the values of the foreign key attributes are guaranteed to occur as the primary key value of a tuple in the referenced relation.

Column level constraints Using Primary key

Q13. Write a query to create primary constraints with column level

Primary key

Syntax for Column level constraints Using Primary key:

SQL:>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE)<TYPE OF CONSTRAINTS> , COLUMN NAME.1 <DATATYPE>(SIZE).....);

QUERY: 13

SQL>CREATE TABLE EMPLOYEE(EMPNO NUMBER(4) **PRIMARYKEY,** ENAME VARCHAR2(10),JOB VARCHAR2(6),SAL NUMBER(5),DEPTNO NUMBER(7)):

Column level constraints Using Primary key with naming convention

Q14. Write a query to create primary constraints with column level with naming convention

Syntax for Column level constraints Using Primary key:

SQL: >CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE> (SIZE) CONSTRAINTS <NAME OF THE CONSTRAINTS><TYPE OF THE CONSTRAINTS>, COLUMN NAME.1 <DATATYPE> (SIZE).....);

QUERY: 14

SQL>CREATE TABLE EMPLOYEE (EMPNO NUMBER (4) **CONSTRAINT EMP_EMPNO_PK PRIMARY KEY,** ENAME VARCHAR2(10), JOB VARCHAR2(6), SAL NUMBER(5),DEPTNO NUMBER(7));

Table Level Primary Key Constraints

Q15. Write a query to create primary constraints with table level with naming convention

Syntax for Table level constraints Using Primary key:

SQL: >CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE) , COLUMN NAME.1 <DATATYPE> (SIZE), CONSTRAINTS <NAME OF THE CONSTRAINTS><TYPE OF THE CONSTRAINTS>);

QUERY: 15

SQL>CREATE TABLE EMPLOYEE (EMPNO NUMBER(6),ENAME VARCHAR2(20), JOB VARCHAR2(6), SAL NUMBER(7), DEPTNO NUMBER(5),**CONSTRAINT EMP_EMPNO_PK PRIMARYKEY(EMPNO));**

Table level constraint with alter command (primary key):

Q16. Write a query to create primary constraints with alter command

Syntax for Column level constraints Using Primary key:

SQL:>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE> (SIZE), COLUMN NAME.1 <DATATYPE> (SIZE));

SQL> ALTER TABLE <TABLE NAME> ADDCONSTRAINTS <NAME OF THE CONSTRAINTS><TYPE OF THE CONSTRAINTS><COLUMN NAME>);

QUERY: 16

SQL>CREATE TABLE EMPLOYEE(EMPNO NUMBER(5),ENAME VARCHAR2(6), JOB VARCHAR2(6), SAL NUMBER(6);

SQL>ALTER TABLE EMP3 ADD CONSTRAINT **EMP3_EMPNO_PK PRIMARY KEY (EMPNO);**

Reference /foreign key constraint

Column level foreign key constraint:

Q.17. Write a query to create foreign key constraints with column level

Parent Table:

Syntax for Column level constraints Using Primary key:

SQL>CREATE<OBJ.TYPE><OBJ.NAME>(COLUMNNAME.1 <DATATYPE>(SIZE)<TYPE OF CONSTRAINTS> , COLUMN NAME.1 <DATATYPE>(SIZE).....);

Child Table:

Syntax for Column level constraints using foreign key:

SQL>CREATE<OBJ.TYPE><OBJ.NAME>(COLUMN NAME.1 <DATATYPE>(SIZE), COLUMN NAME2 <DATATYPE> (SIZE) REFERENCES <TABLE NAME>(COLUMN NAME> ……..);

QUERY: 17

SQL>CREATE TABLE DEPT (DEPTNO NUMBER(2) PRIMARYKEY,DNAME VARCHAR2(20),LOCATION VARCHAR2(15));

SQL>CREATE TABLE EMP4 (EMPNO NUMBER(3), DEPTNO NUMBER(2) **REFERENCES DEPT(DEPTNO),**DESIGN VARCHAR2(10));

Column level foreign key constraint with naming conversions:

Parent Table:

Syntax for Column level constraints Using Primary key:

Q.18. Write a query to create foreign key constraints with column level

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE> (SIZE)<TYPE OF CONSTRAINTS> , COLUMN NAME.1 <DATATYPE> (SIZE).....);

Child Table:

Syntax for Column level constraints using foreign key:

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE) , COLUMN NAME2 <DATATYPE> (SIZE) CONSTRAINT <CONST.NAME>REFERENCES <TABLE NAME> (COLUMN NAME>.....);

QUERY: 18

SQL>CREATE TABLE DEPT (DEPTNO NUMBER (2) PRIMARY KEY, DNAME VARCHAR2(20), LOCATION VARCHAR2(15));

SQL>CREATE TABLE EMP4A (EMPNO NUMBER (3), DEPTNO NUMBER(2)**CONSTRAINT EMP4A_DEPTNO_FK REFERENCES DEPT(DEPTNO),** DESIGN VARCHAR2(10));

Table Level Foreign Key Constraints

Q.19. Write a query to create foreign key constraints with Table level

Parent Table:

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE> (SIZE)<TYPE OF CONSTRAINTS> , COLUMN NAME.1 <DATATYPE> (SIZE).....);

Child Table:

Syntax for Table level constraints using foreign key:

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE), COLUMN NAME2 <DATATYPE> (SIZE), CONSTRAINT <CONST.NAME>REFERENCES <TABLE NAME> (COLUMN NAME>);

QUERY: 19

SQL>CREATE TABLE DEPT(DEPTNO NUMBER(2) PRIMARY KEY, DNAME VARCHAR2(20), LOCATION VARCHAR2(15));

SQL>CREATE TABLE EMP5 (EMPNO NUMBER(3), DEPTNO NUMBER(2),DESIGN VARCHAR2(10)**CONSTRAINT ENP2_DEPTNO_FK FOREIGN KEY(DEPT NO)REFERENCESDEPT(DEPTNO));**

Table Level Foreign Key Constraints with Alter command

Q.20. Write a query to create foreign key constraints with Table level with alter command.

Parent Table:

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE> (SIZE) <TYPE OF CONSTRAINTS>, COLUMN NAME.1 <DATATYPE>(SIZE).....);

Child Table:

Syntax for Table level constraints using foreign key:

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE), COLUMN NAME2 <DATATYPE> (SIZE));

SQL> ALTER TABLE <TABLE NAME>ADDCONSTRAINT<CONST.NAME> REFERENCES <TABLE NAME> (COLUMN NAME>);

QUERY: 20

SQL>CREATE TABLE DEPT (DEPTNO NUMBER (2) PRIMARY KEY, DNAME VARCHAR2 (20), LOCATION VARCHAR2(15));

SQL>CREATE TABLE EMP5(EMPNO NUMBER(3),DEPTNO NUMBER(2), DESIGN VARCHAR2(10));

SQL>ALTER TABLE EMP6 ADD CONSTRAINT EMP6_DEPTNO_FK FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO);

ALTER TABLE OrdersDROP CONSTRAINT FK_PersonOrder;

Check constraint

Column Level Check Constraint

Q.21. Write a query to create Check constraints with column level

Syntax for column level constraints using Check:

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE) CONSTRAINT <CONSTRAINTS NAME><TYPE OF CONSTRAINTS> (CONSTRAITNS CRITERIA) , COLUMN NAME2 <DATATYPE> (SIZE));

QUERY: 21

SQL>CREATE TABLE EMP7(EMPNO NUMBER(3),ENAME VARCHAR2(20),DESIGN VARCHAR2(15),SAL NUMBER(5)CONSTRAINT EMP7_SAL_CK CHECK(SAL>500 AND SAL<10001),DEPTNO NUMBER(2));

Table Level Check Constraint:

Q.22. Write a query to create Check constraints with table level

Syntax for Table level constraints using Check:

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE), (COLUMN NAME2 <DATATYPE> (SIZE), CONSTRAINT<CONSTRAINTS NAME><TYPE OFCONSTRAINTS> (CONSTRAITNS CRITERIA)) ;

QUERY: 22

SQL>CREATE TABLE EMP8 (EMPNO NUMBER (3),ENAME VARCHAR2(20), DESIGN VARCHAR2(15), SAL NUMBER(5),DEPTNO NUMBER(2),CONSTRAINTS EMP8_SAL_CK CHECK(SAL>500 AND SAL<10001));

Check Constraint with Alter Command

Q.23. Write a query to create Check constraints with table level using alter command.

Syntax for Table level constraints using Check:

SQL>CREATE <OBJ.TYPE><OBJ.NAME> (COLUMN NAME.1 <DATATYPE>(SIZE), (COLUMN NAME2 <DATATYPE> (SIZE), CONSTRAINT<CONSTRAINTS NAME><TYPE OF CONSTRAINTS> (CONSTRAITNS CRITERIA)) ;

QUERY:23

SQL>CREATE TABLE EMP9 (EMPNO NUMBER,ENAME VARCHAR2(20),DESIGN VARCHAR 2(15),SAL NUMBER(5));

SQL>ALTER TABLE EMP9 ADD CONSTRAINTS EMP9_SAL_CK CHECK(SAL>500 AND SAL<10001);

Unique Constraint

Column Level Constraint

Q.24. Write a query to create unique constraints with column level

Syntax for Column level constraints with Unique:

SQL :> CREATE <OBJ.TYPE><OBJ.NAME> (<COLUMN NAME.1><DATATYPE> (SIZE) CONSTRAINT <NAME OF CONSTRAINTS><CONSTRAINT TYPE>, (COLUMN NAME2 <DATATYPE> (SIZE)) ;

QUERY: 24

SQL>CREATE TABLE EMP10 (EMPNO NUMBER(3),ENAME VARCHAR2(20),DESGIN VARCHAR2(15)CONSTRAINT EMP10_DESIGN_UK UNIQUE, SAL NUMBER(5));

Table Level Constraint

Q.25. Write a query to create unique constraints with table level

Syntax for Table level constraints with Unique:

SQL :> CREATE <OBJ.TYPE><OBJ.NAME> (<COLUMN NAME.1><DATATYPE> (SIZE), (COLUMN NAME2 <DATATYPE> (SIZE), CONSTRAINT<NAME OF CONSTRAINTS><CONSTRAINT TYPE>(COLUMN NAME);) ;

QUERY: 25

SQL>CREATE TABLE EMP11(EMPNO NUMBER(3),ENAME VARCHAR2(20),DESIGN VARCHAR2(15),SAL NUMBER(5),CONSTRAINT EMP11_DESIGN_UK UNIGUE(DESIGN));

Table Level Constraint Alter Command

Q.26. Write a query to create unique constraints with table level

Syntax for Table level constraints with Check Using Alter

SQL :> CREATE <OBJ.TYPE><OBJ.NAME> (<COLUMN NAME.1><DATATYPE> (SIZE), (COLUMN NAME2 <DATATYPE> (SIZE)) ;

SQL> ALTER TABLE ADD <CONSTRAINTS><CONSTRAINTS NAME><CONSTRAINTS TYPE>(COLUMN NAME);

QUERY: 26

SQL>CREATE TABLE EMP12 (EMPNO NUMBER (3),ENAME VARCHAR2(20), DESIGN VARCHAR2 (15), SAL NUMBER(5));

SQL>ALTER TABLE EMP12 ADD CONSTRAINT EMP12_DESIGN_UK UNIQUE(DESING);

Not Null

Column Level Constraint

Q.27. Write a query to create Not Null constraints with column level

Syntax for Column level constraints with Not Null:

SQL :> CREATE <OBJ.TYPE><OBJ.NAME> (<COLUMN NAME.1><DATATYPE> (SIZE) CONSTRAINT <NAME OF CONSTRAINTS><CONSTRAINT TYPE>, (COLUMN NAME2 <DATATYPE> (SIZE)) ;

QUERY: 27

SQL>CREATE TABLE EMP13(EMPNO NUMBER(4),ENAME VARCHAR2(20) CONSTRAINT EMP13_ENAME_NN NOT NULL,DESIGN VARCHAR2(20),SAL NUMBER(3));

Null

Column Level Constraint

Q.28. Write a query to create Null constraints with column level

Syntax for Column level constraints with Null:

SQL > CREATE <OBJ.TYPE><OBJ.NAME> (<COLUMN NAME.1><DATATYPE> (SIZE) CONSTRAINT <NAME OF CONSTRAINTS><CONSTRAINT TYPE>, (COLUMN NAME2 <DATATYPE> (SIZE)) ;

QUERY:28

SQL>CREATE TABLE EMP13(EMPNO NUMBER(4),ENAME VARCHAR2(20) CONSTRAINT EMP13_ENAME_NN NULL,DESIGN VARCHAR2(20),SAL NUMBER(3));

2. DATA MANIPULATION LANGUAGE (DML) COMMANDS IN DBMS

AIM:

To execute and verify the DML

DML (DATA MANIPULATION LANGUAGE)

- SELECT
- INSERT
- DELETE
- UPDATE

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Insert the record into table

STEP 4: Update the existing records into the table

STEP 5: Delete the records in to the table

SQL COMMANDS

1. COMMAND NAME: **INSERT**
   COMMAND DESCRIPTION: **INSERT** command is used to Insert objects in the database.

2. COMMAND NAME: **SELECT**
   COMMAND DESCRIPTION: **SELECT** command is used to SELECT the object fromthe database.

3. COMMAND NAME: **UPDATE**
   COMMAND DESCRIPTION: **UPDATE** command is used to UPDATE the records from the table.

4. COMMAND NAME: **DELETE**
   COMMAND DESCRIPTION: **DELETE** command is used to DELETE the Records form the table.

QUERY: 01

Q1. Write a query to insert the records in to employee.

Syntax for Insert Records in to a table:

SQL :> INSERT INTO <TABLE NAME> VALUES< VAL1, 'VAL2',.....);

QUERY: 01

INSERT A RECORD FROM AN EXISTING TABLE:

SQL>INSERT INTO EMP VALUES (101,'NAGARAJAN','LECTURER',15000);

1 row created.

QUERY: 02

Q3. Write a query to display the records from employee.

Syntax for select Records from the table:

SQL> SELECT * FROM <TABLE NAME>;

QUERY: 02

DISPLAY THE EMP TABLE:

SQL> SELECT * FROM EMP;

EMPNO	ENAME	DESIGNATION	SALARY
101	NAGARAJAN	LECTURER	15000

INSERT A RECORD USING SUBSITUTION METHOD

Q3. Write a query to insert the records in to employee using substitution method.

Syntax for Insert Records into the table:

SQL > INSERT INTO <TABLE NAME> VALUES< '&column name', '&column name 2',.....);

QUERY: 03

```
SQL> INSERT INTO EMP VALUES(&EMPNO,'&ENAME','&DESIGNATIN','&SALARY');
Enter value for empno: 102
Enter value for ename: SARAVANAN
Enter value for designatin: LECTURER
Enter value for salary: 15000

old   1: INSERT INTO EMP VALUES(&EMPNO,'&ENAME','&DESIGNATIN','&SALARY')
new   1: INSERT INTO EMP VALUES(102,'SARAVANAN','LECTURER','15000')

1 row created.

SQL> /
Enter value for empno: 103
Enter value for ename: PANNERSELVAM
Enter value for designatin: ASST. PROF
Enter value for salary: 20000

old   1: INSERT INTO EMP VALUES(&EMPNO,'&ENAME','&DESIGNATIN','&SALARY')
new   1: INSERT INTO EMP VALUES(103,'PANNERSELVAM','ASST.PROF','20000')

1 row created.

SQL> /
Enter value for empno: 104
Enter value for ename: CHINNI
Enter value for designatin: HOD, PROF
Enter value for salary: 45000

old   1: INSERT INTO EMP VALUES(&EMPNO,'&ENAME','&DESIGNATIN','&SALARY')
new   1: INSERT INTO EMP VALUES(104,'CHINNI','HOD, PROF','45000')

1 row created.

SQL> SELECT * FROM EMP;

EMPNO      ENAME          DESIGNATION    SALARY
---------- -------------- -------------- ----------
  101      NAGARAJAN      LECTURER       15000
  102      SARAVANAN      LECTURER       15000
  103      PANNERSELVAM   ASST.PROF      20000
```

104 CHINNI HOD, PROF 45000

QUERY: 04

Q4. Write a query to update the records from employee.

Syntax for update Records from the table:

SQL> UPDATE <<TABLE NAME> SET <COLUMNANE>=<VALUE> WHERE<COLUMN NAME=<VALUE>;

QUERY: 04

SQL> UPDATE EMP SET SALARY=16000 WHERE EMPNO=101;

1 row updated.

SQL> SELECT * FROM EMP;

EMPNO	ENAME	DESIGNATION	SALARY
101	NAGARAJAN	LECTURER	16000
102	SARAVANAN	LECTURER	15000
103	PANNERSELVAM	ASST.PROF	20000
104	CHINNI	HOD, PROF	45000

QUERY: 05

Q5. Write a query to update multiple records from employee.

Syntax for update multiple Records from the table:

SQL> UPDATE <<TABLE NAME> SET <COLUMNANE>=<VALUE> WHERE<COLUMN NAME=<VALUE>;

QUERY: 05

SQL>UPDATE EMP SET SALARY = 16000, DESIGNATIN='ASST. PROF' WHERE EMPNO=102;

1 row updated.

SQL> SELECT * FROM EMP;

EMPNO	ENAME	DESIGNATION	SALARY
101	NAGARAJAN	LECTURER	16000
102	SARAVANAN	ASST. PROF	16000
103	PANNERSELVAM	ASST. PROF	20000
104	CHINNI	HOD, PROF	45000

QUERY: 06

Q6. Write a query to delete records from employee.

Syntax for delete Records from the table:

SQL> DELETE <TABLE NAME> WHERE <COLUMN NAME>=<VALUE>;

QUERY: 06

SQL> DELETE EMP WHERE EMPNO=103;

1 row deleted.

SQL> SELECT * FROM EMP;

EMPNO	ENAME	DESIGNATION	SALARY
101	NAGARAJAN	LECTURER	16000
102	SARAVANAN	ASST. PROF	16000
104	CHINNI	HOD, PROF	45000

3. DATA CONTROL LANGUAGE (DCL) & TRANSACTION CONTROL LANGUAGE (TCL)COMMANDS IN DBMS

AIM:

To execute and verify the DCL and TCLLanguage commands

DCL (DATA CONTROL LANGUAGE)

- GRANT
- REVOKE

TCL (TRANSACTION CONTROL LANGUAGE)

- COMMIT
- ROLL BACK
- SAVE POINT

PROCEDURE

STEP 1: Start

STEP 2: Create the table with its essential attributes.

STEP 3: Insert the record into table

STEP 4: Use Grant to grant access to the database

STEP 5: Use Revoke to remove the grant

STEP 6: Use save point if any changes occur in any portion of the record to undo its original state.

STEP 7: Use rollback for completely undo the records

STEP 6: Use commit for permanently save the records.

SQL COMMANDS

1. COMMAND NAME: **GRANT**
   COMMAND DESCRIPTION: It is used to create users and grant access to the database.

2. COMMAND NAME:**REVOKE**
   COMMAND DESCRIPTION: Using **REVOKE**, the DBA can revoke the granted database privileges from the user

3. COMMAND NAME: **DELETE**
   COMMAND DESCRIPTION: **DELETE** command is used to **DELETE** the Records form the table

4. COMMAND NAME: **COMMIT**
   COMMAND DESCRIPTION: **COMMIT** command is used to save the Records.

5. COMMAND NAME: **ROLLBACK**
   COMMAND DESCRIPTION: **ROLL BACK** command is used to undo the Records.

6. COMMAND NAME: **SAVE POINT**
   COMMAND DESCRIPTION: **SAVE POINT** command is used to undo the Records in a particular transaction.

DCL (DATA CONTROL LANGUAGE):

CREATING A USER

SQL>CONNECT SYSTEM/MANAGER;

SQL>CREATE USER "USERNAME" IDENTIFIED BY "PASSWORD"

SQL>GRANT DBA TO "USERNAME"

SQL>CONNECT "USERNAME"/"PASSWORD";

EXAMPLE

CREATING A USER

SQL>CONNECT SYSTEM/MANAGER;

SQL>CREATE USER CSE2 IDENTIFIED BY CSECSE;

SQL>GRANT DBA TO CSE2;

SQL>CONNECT CSE2/CSECSE;

SQL>REVOKE DBA FROM CSE2;

TCL (TRANSACTION CONTROL LANGUAGE)

QUERY: 01

Q1. Write a query to implement the save point.

Syntax for save point: ——————> Commit.

SQL> SAVEPOINT <SAVE POINT NAME>;

QUERY: 01

SQL> SAVEPOINT S1;

Savepoint created.

SQL> SELECT * FROM EMP;

EMPNO	ENAME	DESIGNATION	SALARY
101	NAGARAJAN	LECTURER	16000
102	SARAVANAN	ASST. PROF	16000
104	CHINNI	HOD, PROF	45000

SQL> INSERT INTO EMP VALUES(105,'PARTHASAR','STUDENT',100);

1 row created.

SQL> SELECT * FROM EMP;

EMPNO	ENAME	DESIGNATION	SALARY
105	PARTHASAR	STUDENT	100
101	NAGARAJAN	LECTURER	16000
102	SARAVANAN	ASST. PROF	16000
104	CHINNI	HOD, PROF	45000

QUERY: 02

Q2. Write a query to implement the Rollback.

Syntax for Rollback:

SQL> ROLL BACK <SAVE POINT NAME>;

QUERY: 02

SQL> ROLL BACK S1;

Rollback complete.

SQL> SELECT * FROM EMP;

EMPNO	ENAME	DESIGNATION	SALARY
101	NAGARAJAN	LECTURER	16000
102	SARAVANAN	ASST. PROF	16000
103	PANNERSELVAM	ASST. PROF	20000
104	CHINNI	HOD, PROF	45000

QUERY: 03

Q3. Write a query to implement the Rollback.

Syntax for commit:

SQL> COMMIT;

QUERY: 04

SQL> COMMIT;

Commit complete.

4. IN BUILT FUNCTIONS

AIM :

Implementation of different types of inbuilt functions with suitable examples.

Objective:

- Number Function
- Aggregate Function
- Character Function
- Conversion Function
- Date Function

NUMBER FUNCTION:

Abs(n) :Select abs(-15) from dual;

Exp(n): Select exp(4) from dual;

Power(m,n): Select power(4,2) from dual;

Mod(m,n): Select mod(10,3) from dual;

Round(m,n): Select round(100.256,2) from dual;

Trunc(m,n): ;Select trunc(100.256,2) from dual;

Sqrt(m,n);Select sqrt(16) from dual;

Develop aggregate plan strategies to assist with summarization of several data entries.

Aggregative operators: In addition to simply retrieving data, we often want to perform some computation or summarization. SQL allows the use of arithmetic expressions. We nowconsider a powerful class of constructs for computing aggregate values such as MIN andSUM.

1. Count: COUNT following by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (*) indicates all the tuples of the column.

Syntax: COUNT (Column name)
Example:SELECT COUNT (Sal) FROM emp;

2. SUM: SUM followed by a column name returns the sum of all the values in that column.
Syntax: SUM (Column name)
Example: SELECT SUM (Sal) From emp;

3. AVG: AVG followed by a column name returns the average value of that column values.
Syntax: AVG (n1, n2...)
Example: Select AVG (10, 15, 30) FROM DUAL;

4. MAX: MAX followed by a column name returns the maximum value of that column.
Syntax: MAX (Column name)
Example: SELECT MAX (Sal) FROM emp;

SQL> select deptno, max(sal) from emp group by deptno;

```
DEPTNO MAX (SAL)
---------- ----------
10       5000
20       3000
30       2850
```

SQL> select deptno, max (sal) from emp group by deptno having max(sal)<3000;

```
   DEPTNO    MAX(SAL)
   ----------  ----------
      30          2850
```

5. MIN: MIN followed by column name returns the minimum value of that column.
Syntax: MIN (Column name)
Example:SELECT MIN (Sal) FROM emp;

SQL>select deptno,min(sal) from emp group by deptno having min(sal)>1000;

```
   DEPTNO  MIN (SAL)
   ---------- ----------
     10         1300
```

CHARACTER FUNCTION:

initcap(char) : select initcap('hello') from dual;

lower (char): select lower ('HELLO') from dual;

upper (char) :select upper ('hello') from dual;

ltrim (char,[set]): select ltrim ('cseit', 'cse') from dual;

rtrim (char,[set]): select rtrim ('cseit', 'it') from dual;

replace (char,search): select replace('jack and jue','j','bl') from dual;

CONVERSION FUNCTIONS:

To_char: TO_CHAR (number) converts n to a value of VARCHAR2 data type. using the optional number format fmt. The value n can be of type NUMBER, BINARY_FLOAT, or BINARY_DOUBLE.

SQL>select to_char(65,'RN')from dual;

LXV

To_number :TO_NUMBER converts expr to a value of NUMBER data type.

SQL>Select to_number ('1234.64') from Dual;

1234.64

To_date:TO_DATE converts char of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of DATE data type.

SQL>SELECT TO_DATE('January 15, 1989, 11:00 A.M.')FROM DUAL;

TO_DATE

15-JAN-89

STRING FUNCTIONS:

Concat: CONCAT returns char1 concatenated with char2. Both char1 and char2 can be any of the datatypes

SQL>SELECT CONCAT('ORACLE','CORPORATION')FROM DUAL;

ORACLECORPORATION

Lpad: LPAD returns expr1, left-padded to length n characters with the sequence of characters in expr2.

SQL>SELECT LPAD('ORACLE',15,'*')FROM DUAL;

*********ORACLE

Rpad:RPAD returns expr1, right-padded to length n characters with expr2, replicated as many times as necessary.

SQL>SELECT RPAD ('ORACLE',15,'*')FROM DUAL;

ORACLE*********

Ltrim: Returns a character expression after removing leading blanks.

SQL>SELECT LTRIM('SSMITHSS','S')FROM DUAL;

MITHSS

Rtrim: Returns a character string after truncating all trailing blanks

SQL>SELECT RTRIM('SSMITHSS','S')FROM DUAL;

SSMITH

Lower: Returns a character expression after converting uppercase character data to lowercase.

SQL>SELECT LOWER('DBMS')FROM DUAL;

dbms

Upper: Returns a character expression with lowercase character data converted to uppercase

SQL>SELECT UPPER('dbms')FROM DUAL;

DBMS

Length: Returns the number of characters, rather than the number of bytes, of the given string expression, excluding trailing blanks.

SQL>SELECT LENGTH('DATABASE')FROM DUAL;

8

Substr: Returns part of a character, binary, text, or image expression.

SQL>SELECT SUBSTR('ABCDEFGHIJ'3,4)FROM DUAL;

CDEF

Instr: The INSTR functions search string for substring. The function returns an integer indicating the position of the character in string that is the first character of this occurrence.

SQL>SELECT INSTR('CORPORATE FLOOR','OR',3,2)FROM DUAL;

14

DATE FUNCTIONS:

Sysdate:

SQL>SELECT SYSDATE FROM DUAL;

29-JULY-17

next_day:

SQL>SELECT NEXT_DAY(SYSDATE,'WED')FROM DUAL;

05-JAN-09

add_months:

SQL>SELECT ADD_MONTHS(SYSDATE,2)FROM DUAL;

28-FEB-09

last_day:

SQL>SELECT LAST_DAY(SYSDATE)FROM DUAL;

31-DEC-08

months_between:

SQL>SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE)FROM EMP;

4

Least:

SQL>SELECT LEAST('10-JAN-07','12-OCT-07')FROM DUAL;

10-JAN-07

Greatest:

SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;

10-JAN-07

Trunc:

SQL>SELECT TRUNC(SYSDATE,'DAY')FROM DUAL;

28-DEC-08

Round:

SQL>SELECT ROUND(SYSDATE,'DAY')FROM DUAL;

28-DEC-08

to_char:

SQL> select to_char(sysdate, "dd\mm\yy") from dual;

24-mar-05.

to_date:

SQL> select to date (sysdate, "dd\mm\yy") from dual;

24-mar-05.

GROUP BY CLAUSE

This allows us to use simultaneous column name and group functions.

Example: Select max(percentage), deptname from student group by deptname;

HAVING CLAUSE

This is used to specify conditions on rows retrieved by using group by clause.

Example: Select max(percentage), deptname from student group by deptname having count(*)>=50;