Lab 6

Objectives:

At the end of this lab, you should be able to:

- Use Oracle constraints.
- Use set operators to combine two or more queries.
- Use oracle data dictionary.

Constraints

The Oracle Server uses *constraints* to prevent invalid data entry into tables.
You can use constraints to do the following:

- Enforce rules at the table level whenever a row is inserted, updated, or deleted from that table. The constraint must be satisfied for the operation to succeed.
- Prevent the deletion of a table if there are dependencies from other tables.

Constraint	Description
NOT NULL	Specifies that this column may not contain a null value
UNIQUE	Specifies a column or combination of columns whose values must be unique for all rows in the table
PRIMARY KEY	Uniquely identifies each row of the table
FOREIGN KEY	Establishes and enforces a foreign key relationship between the column and a column of the referenced table
CHECK	Specifies a condition that must be true

Create a constraint:

- At the same time as the table is created
- After the table has been created

1- The NOT NULL Constraint

Example

```
CREATE TABLE emp5(
        empno NUMBER(4),
        ename  VARCHAR2(10) NOT NULL,
        deptno NUMBER(2) NOT NULL);
```

Try to insert a new record to the table emp5 and make valued of *ename* null.
NOTE: because no constraint name is provided, the constraint is named automatically by Oracle server.

2- The UNIQUE Key Constraint

Example

```
CREATE TABLE   dept5(
        deptno   NUMBER(2),
        dname    VARCHAR2(14),
        loc      VARCHAR2(13),
        CONSTRAINT dept5_dname_uk UNIQUE(dname));
```

Try to insert two records to the table dept5 with the same values for dname field

---

3- The PRIMARY KEY Constraint

Example

```
CREATE TABLE   dept6(
        deptno   NUMBER(2),
        dname    VARCHAR2(14),
        loc      VARCHAR2(13),
        CONSTRAINT dept6_dname_uk2 UNIQUE (dname),
        CONSTRAINT dept6_deptno_pk PRIMARY KEY(deptno));
```

Try to insert two records with the same primary key.

Difference between primary key and unique constraints:

1- Primary key field cannot have null values, while unique fileds can have null.
2- Only one primary key constraint in a table, but you can have multiple unique constraints for a table.

Composite primary key:

It is a combination of two or more columns

Example:

```
CREATE TABLE CUSTOMER( YEAR NUMBER(4),
ID NUMBER (4),
NAME VARCHAR2(20),
PHONE NUMBER (7),
CONSTRAINT CUSTOMER_YEAR_ID_PK PRIMARY KEY (YEAR,ID) );
```

4- The FOREIGN KEY Constraint

Example

```
CREATE TABLE emp6(
 empno          NUMBER(4),
 ename VARCHAR2(10) NOT NULL,
deptno number(4),
 CONSTRAINT emp6_deptno_fk FOREIGN KEY (deptno)
REFERENCES dept (deptno));
```

The foreign key is defined in the child table, and the table containing the referenced column is the parent table. The foreign key is defined using a combination of the following keywords:

- FOREIGN KEY is used to define the column in the child table at the table constraint level.
- REFERENCES identifies the table and column in the parent table.
- ON DELETE CASCADE indicates that when the row in the parent table is deleted, the dependent rows in the child table will also be deleted.
- Without the ON DELETE CASCADE option, the row in the parent table cannot be deleted if it is referenced in the child table.

Example

```
CREATE TABLE CUSTOMER2 (ID NUMBER(4) PRIMARY KEY,
NAME VARCHAR2(20),
DEPTNO NUMBER (3),
CONSTRAINT CUSTOMER2_DEPTNO_FK FOREIGN KEY(DEPTNO)
REFERENCES DEPT(DEPTNO) ON DELETE CASCADE );
```

- Insert a new record to dept table as follows
  Deptno: 88 , Dname: HR, Loc: Doha.
- Insert a new record to table customer2 as follows
  Id: 123, name : Ahmed , deptno :88.
- Now delete the dept 88.
- Check if the customer 123 was deleted.

5- The CHECK Constraint

Example

```
CREATE TABLE emp7(
        empno NUMBER(4),
        ename  VARCHAR2(10) NOT NULL,
        deptno NUMBER(2) NOT NULL)
        gender CHAR(1),
        CONSTRAINT emp7_gender_ck  CHECK (gender='M' or gender='F'));
```

Try to insert a new record with the gender ‘m’.

Note: This type of constraint can be used to add null constraint with a name.

Example

```
CREATE TABLE CUSTOMER3 (
ID NUMBER (3),
NAME VARCHAR2(20),
CONSTRAINT NAME_NN CHECK (NAME IS NOT NULL));
```

Adding a Constraint after creating the table:

Example

```
CREATE TABLE EMP8 AS SELECT * FROM EMP;

ALTER TABLE     emp8
ADD CONSTRAINT  EMP8_EMPNO_PK primary key (empno);
```

Note : a constraint cannot be added if the data already in that table violates that constraint.

Dropping a Constraint

Example

```
ALTER TABLE        emp6
DROP CONSTRAINT  emp_mgr_fk;
```

Disabling and Enabling Constraint

You can disable a constraint without dropping it ore recreating it.

Example

```
ALTER TABLE EMP8 DISABLE CONSTRAINT EMP8_EMPNO_PK;

ALTER TABLE EMP8 ENABLE CONSTRAINT EMP8_EMPNO_PK;
```

Set Operators:

Create a new table called job_history to keep track of all different positions held by an employee since his/her hire date, the table job_history contains the following fields

Name	Not null?	Type
Emp_id	Not null	Number (4)
Job	Not null	Varchar2(10)
Start_date	Not null	date
End_date	Not null	Date
Deptno		Number (2)

Insert the following records to that table

Emp_id	Job	Start_date	End-date	Deptno
7369	SALESMAN	17-DEC-80	25-JAN-82	30
7369	CLERK	26-JAN-82	30-MAY-85	10
7839	ANALYST	17-NOV-81	12-JUN-83	20
7839	PRESIDENT	13-JUN-83	01-MAR-84	30
7654	CLERK	28-SEP-81	02-FEB-84	10

To display the current and previous job for the employees :

```
Select empno, job from emp Union
Select emp_id, job from job_history;
```

To display all data (include duplicated ones)

```
Select empno, job from emp Union  all
Select emp_id, job from job_history;
```

To display data for those who currently hold same position as before

```
Select empno, job from emp intersect
Select emp_id, job from job_history;
```

To display employee ID's for those who never change their jobs

```
Select empno  from emp minus
Select emp_id  from job_history;
```

Oracle Data Dictionary :

The *data dictionary* is a **read-only** set of tables that provides information about its associated database. For example the names of Oracle users, privileges and roles each user has been granted …etc.

The data dictionary has two primary uses:

- Oracle accesses the data dictionary every time that a DDL statement is issued.
- Any Oracle user can use the data dictionary as a read-only reference for information about the database.

Oracle divides data dictionary views into the three families, as indicated by the following prefixes:

- **USER**:

USER views return information about objects owned by the currently-logged-on database user. Here some of those views:

a. **USER_TABLES** all tables with their name, number of columns, storage
b. **USER_CONSTRAINTS** constraint definitions for tables
c. **USER_INDEXES** all information about indexes created for tables (IND)
d. **USER_OBJECTS** all database objects owned by the user (OBJ)
e. **USER_TRIGGERS** triggers defined by the user
f. **USER_USERS** information about the current user
g. **USER_VIEWS** views defined by the user

For example, a query to USER_TABLES returns a list of all of the relational tables that you own.

```
select table_name, tablespace_name from user_tables
```

```
SELECT constraint_name, constraint_type, status, table_name  FROM user_constraints
```

- **ALL:**

ALL views return information about all objects to which you have access, regardless of who owns them. For example, a query to ALL_TABLES returns a list not only of all of the relational tables that you own, but also of all relational tables to which their owners have specifically granted you access (using the GRANT command).

```
select table_name, tablespace_name from all_tables
```

- **DBA:**

DBA views are generally accessible only to database administrators, and return information about all objects in the database, regardless of ownership or access privileges. For example, a query to DBA_TABLES will return a list of all relational tables in the database, whether or not you own them or have been granted access to them.