Lab5: Creating and Managing Tables

Objectives:

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

- Use DDL : create table, alter table, drop table, truncate table.
- Creating and using Views.

Database Objects

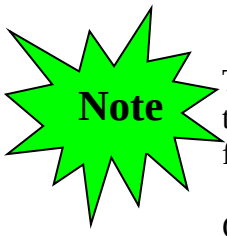
- **Table:** Stores data
- **View:** Subset of data from one or more tables
- **Sequence:** Generates primary key values
- **Index:** Improves the performance of some queries
- **Synonym:** Gives alternative names to objects

Creating Tables

```
CREATE TABLE dept2    (deptno         NUMBER(2), dname VARCHAR2(14),
      loc      VARCHAR2(13));
```

Creating a Table by Using a Subquery

```
CREATE TABLE   dept30 AS
            SELECT          empno, ename, sal*12 ANNSAL, hiredate
            FROM    emp  WHERE          deptno = 30;
```



To create a table with the same structure as an existing table, but without the data from the existing table, use a subquery with a WHERE clause, that will always evaluate as false. For example:

```
CREATE TABLE TEST AS
 (SELECT * FROM emp      WHERE 1 = 2);
```

Copying Rows from Another Table

Create a new table called managers that has the same structure as EMP

```
CREAT TABLE managers(id,name,salary,hiredate) as
(select empno,ename,sal,hiredate from emp where 1=2);
```

Copy data of managers to the new table as following:

```
INSERT INTO managers(id, name, salary, hiredate)
            SELECT        empno, ename, sal, hiredate  FROM   emp
      WHERE        job = 'MANAGER';
```


Prepared by : Dr. Saleh Alhazbi

Exercise : Create table Tax which has the following structure

Tax	
Empno	Number(5)
Tax	Number(10,2)

Fill the table with employees data and the tax of their salaries, where tax is 5%.

Solution:

```
CREATE TABLE TAX ( EMPNO NUMBER(5),
TAX NUMBER (10,2));

INSERT INTO TAX SELECT EMPNO, SAL*0.05 FROM EMP;
```

It can be done in one command as follows :

```
CREATE TABLE TAX AS SELECT EMPNO,SAL *0.05
  AS TAX FROM EMP ;
```

RENAME A TABLE:

You can rename any database object using the command rename:

Example :

```
RENAME DEPT2 TO DEPARTMENT2.
```

The ALTER TABLE Statement

1) Adding a Column

```
ALTER TABLE dept30
 2  ADD                (PHONE NUMBER (6));
```

You can add more than one column in a single ALTER command:

```
ALTER TABLE DEPT30 ADD (ADDRESS VARCHAR2(20), EMAIL VARCHAR2(10));
```

2) Modifying a Column

```
SQL> ALTER TABLE       dept30
 2  MODIFY          (ename VARCHAR2(15));
```

Try to modify the size of column ENAME in dept30 table to 5 characters.

To change the name of a specific field (column):

```
ALTER TABLE DEPT30 RENAME COLUMN LOC TO LOCATION;
```

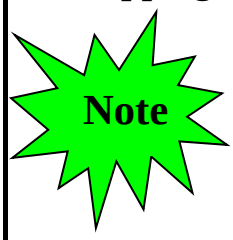
3) Dropping a Column

```
ALTER TABLE       dept30
 DROP  COLUMN   phone ;
```

Try to drop the field *deptno* in table dept.

To drop more than one column :
ALTER TABLE dept30
DROP (address,email) ;

Dropping a Table



- **All data and structure in the table is deleted.**
- **You *cannot* roll back this statement.**

```
SQL> DROP TABLE dept30;
```

Try to drop the table dept.

TRUNCATE a TABLE

TRUNCATE TABLE command is used to delete complete data from an existing table.

- It removes the data in the table
- It is considered a DDL command.
- It cannot be rolled back

Example:

```
Truncate Table Tax;
```

Difference between TRUNCATE, DELETE:

- Truncate will delete all data, while delete will delete records based on a condition.
- Delete is DML, truncate is DDL.
- Delete needs comment, truncate does not.
- Delete can be rolled back and restore the data, while truncate cannot .

Practice:

1. Create the EMPLOYEE2 table based on the structure of the EMP table. Include only the EMPNO, ENAME, and DEPTNO columns. Name the columns in your new table ID,LAST_NAME, and DEPT_ID, respectively.

2. Modify the EMPLOYEE2 table to allow for longer employee last names. Confirm your modification.

3. Drop the LAST_NAME column from the EMPLOYEE2 table. Confirm your modification by checking the description of the table.

4. Drop the EMPLOYEE2 table.

Views

A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed.

Simple Views

Example

```
CREATE VIEW     v1
    AS SELECT   empno, ename, job
    FROM        emp
    WHERE       deptno = 10;

SQL> DESCRIBE v1
SQL>  SELECT *
      FROM v1;
```

Exercise:

1) Insert a new employee to emp table :
   Empno: 456, name : Khaled, deptno: 10.
   Query the view v1
2) Create a view contains empno, ename, dname based on the tables emp and dept.

Complex View:

Example

```
SQL> CREATE VIEW      dept_sum
                      (name, minsal, maxsal, avgsal)
  AS SELECT       d.dname, MIN(e.sal), MAX(e.sal), AVG(e.sal)
  FROM            emp e, dept d
  WHERE           e.deptno = d.deptno
  GROUP BY        d.dname;
```

Exercise:

- Increase the salaries of employees in dept 10 by 1000.
- Display the data in dept_sum.

Dropping a VIEW

To remove a view

```
DROP VIEW V1
```

Practice

1. Create a view called EMP_VU based on the employee number, employee name, and department number from the EMP table. Change the heading for the employee name to EMPLOYEE

2. Using your view EMP_VU, enter a query to display all employee names and department numbers.

Difference between simple views and complex views

Simple View	Complex View
Created from only one table.	Created from more than one tables.
We cannot use group functions like MAX(), COUNT(), etc.	We can use group functions.
Does not contain groups of data.	It can contain groups of data.
DML operations could be performed through a simple view.	DML operations could not always be performed through a complex view.
INSERT, DELETE and UPDATE are directly possible on a simple view.	We cannot apply INSERT, DELETE and UPDATE on complex view directly.

Creating Sequences

CREATE SEQUENCE sequence_name
[INCREMENT BY interval]
[START WITH first_number]
[MAXVALUE max_value | NOMAXVALUE]
[MINVALUE min_value | NOMINVALUE]

CREATE SEQUENCE id_seq increment by 10 start with 10;
SELECT id_seq.nextval FROM dual;
SELECT id_seq.currval FROM dual;

Use Sequence

CREATE TABLE tasks(id number, title varchar2(20));
INSERT INTO tasks VALUES (id_seq.nextval,'DATABASE');

Dropping Sequences

DROP SEQUENCE id_seq;

Practice

1. Create a sequence called TEST_SEQ that start with 5, increment by 1, maximum value is 100. Display the next and current value.

2. Drop TEST_SEQ sequence.