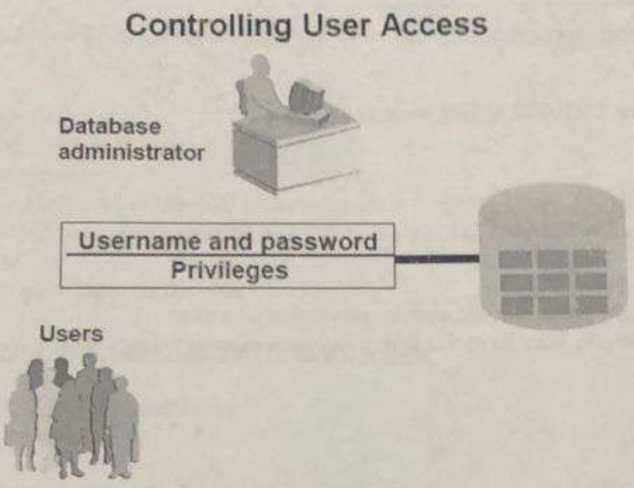
EXERCISE-15

Controlling User Access

Objectives
After the completion of this exercise, the students will be able to do the following:

• Create users
• Create roles to ease setup and maintenance of the security model
• Use the GRANT and REVOKE statements to grant and revoke object privileges
• Create and access database links



Controlling User Access

In a multiple-user environment, you want to maintain security of the database access and use. With Oracle server database security, you can do the following:
• Control database access
• Give access to specific objects in the database
• Confirm given and received *privileges* with the Oracle data dictionary
• Create synonyms for database objects

Privileges

• Database security:
– System security
– Data security
• System privileges: Gaining access to the database
• Object privileges: Manipulating the content of the database objects
• Schemas: Collections of objects, such as tables, views, and sequences

System Privileges

• More than 100 privileges are available.
• The database administrator has high-level system privileges for tasks such as:
– Creating new users

– Removing users
– Removing tables
– Backing up tables

Typical DBA Privileges

System Privilege	Operations Authorized
CREATE USER	Grantee can create other Oracle users (a privilege required for a DBA role).
DROP USER	Grantee can drop another user.
DROP ANY TABLE	Grantee can drop a table in any schema.
BACKUP ANY TABLE	Grantee can back up any table in any schema with the export utility.
SELECT ANY TABLE	Grantee can query tables, views, or snapshots in any schema.
CREATE ANY TABLE	Grantee can create tables in any schema.

Creating Users

The DBA creates users by using the CREATE USER statement.
EXAMPLE:

```
CREATE USER scott IDENTIFIED BY tiger;
```

User System Privileges

• Once a user is created, the DBA can grant specific system privileges to a user.
• An application developer, for example, may have the following system privileges:
– CREATE SESSION
– CREATE TABLE
– CREATE SEQUENCE
– CREATE VIEW
– CREATE PROCEDURE

GRANT *privilege* [, *privilege...*]
TO *user* [, *user*| *role, PUBLIC...*];

Typical User Privileges

System Privilege	Operations Authorized
CREATE SESSION	Connect to the database
CREATE TABLE	Create tables in the user's schema
CREATE SEQUENCE	Create a sequence in the user's schema
CREATE VIEW	Create a view in the user's schema
CREATE PROCEDURE	Create a stored procedure, function, or package in the user's schema

In the syntax:

privilege is the system privilege to be granted

user |role|PUBLIC is the name of the user, the name of the role, or PUBLIC designates that every user is granted the privilege

Note: Current system privileges can be found in the dictionary view SESSION_PRIVS.

Granting System Privileges

The DBA can grant a user specific system privileges.

GRANT create session, create table, create sequence, create view TO scott;

What is a Role?

A role is a named group of related privileges that can be granted to the user. This method makes it easier to revoke and maintain privileges.

A user can have access to several roles, and several users can be assigned the same role. Roles are typically created for a database application.

Creating and Assigning a Role

First, the DBA must create the role. Then the DBA can assign privileges to the role and users to the role.

Syntax
CREATE ROLE *role*;
In the syntax:
role is the name of the role to be created
Now that the role is created, the DBA can use the GRANT statement to assign users to the role as well as
assign privileges to the role.

Creating and Granting Privileges to a Role

```
CREATE ROLE manager;
Role created.

GRANT create table, create view TO manager;
Grant succeeded.

GRANT manager TO DEHAAN, KOCHHAR;
Grant succeeded.
```

- Create a role
- Grant privileges to a role
- Grant a role to users

Changing Your Password

- The DBA creates your user account and initializes your password.

- You can change your password by using the

ALTER USER statement.

```
ALTER USER scott
IDENTIFIED BY lion;
User altered.
```

Find the Solution for the following:

1. What privilege should a user be given to log on to the Oracle Server? Is this a system or an object privilege?

Create session

2. What privilege should a user be given to create tables?

Create table

3. If you create a table, who can pass along privileges to other users on your table?

GRANT SELECT ON emp To user 1 with grant option

4. You are the DBA. You are creating many users who require the same system privileges. What should you use to make your job easier?

GRANT Create TABLE Create view to manager role

5. What command do you use to change your password?

ALTER USER username Identified by new password,

6. Grant another user access to your DEPARTMENTS table. Have the user grant you query access to his or her DEPARTMENTS table.

GRANT selection ON department into user 2

7. Query all the rows in your DEPARTMENTS table.

Select * from departments

8. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources department number 510. Query the other team's table. INSERT into department values (500, Education),

INSERT into department values (510, Human Resource)

9. Query the USER_TABLES data dictionary to see information about the tables that you own.

Select table_name from user_table

10. Revoke the SELECT privilege on your table from the other team.

Revoke select on department from user 2,

11. Remove the row you inserted into the DEPARTMENTS table in step 8 and save the changes.

DELET from departments where department ID (500, 510) ; Commit),'

(4)

Evaluation Procedure	Marks awarded
Practice Evaluation (5)	5
Viva(5)	5
Total (10)	10
Faculty Signature	[illegible]

150

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

```
    DECLARE
      v-emp-id employee employee - id%Type := 110;
      v - salary employee . salary %Type;
      V - incentive Number;

    BEGIN
        Select salary INTO v-salary
          from employees
          where employee_id = v-emp-id

          v- incentive = v_salary * 0.10;
     DBMS.Output .put - Line ('employee ID:'||v-emp
                                                  -id)
    DBMS_output .Put - Line ('Salary', ||v-salary);
    DBMS_ output .Put - line ('Incentive (10%)'||v-incent

Exception:
             when No_Data found then
  DBMS - Output .Put _line ('Employee not found');
   when others then
        DBMS. output . put . line ('Error:'||SQLE

     END;

     /
```

155

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
DECLARE

    "My Var" NUMBER: =1001

  BEGIN
      DBMS - OUTPut. Put_Line (myVar),

      DBMS - OUTPut. Put - LINE ("My Var")

    END.
```

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.

Sample table: employees

```
DECLARE
    v_emp_id employee.employee_id%type = 122.
BEGIN
        update employees
        SET Salary = Salary + (Salary * 0.10)
    where employee_id = v_emp_id;
    DBMS_OUTPUT.PUT_LINE ('Salary updated succe
            for employee ID: '||v_emp_id); Exception

    when    NO_DATA_found Then
        DBMS_output.PUT.LINE ('Employee not
                                    found.')

    when other then
        DBMS_output.Put_line ('Error:"
                        SQL ERRM)

    END.
```

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

```
Create OR REPLACE procedure check_nullis
    a number := 10;
    b Number := Null;
    BEGIN
  If a IS NOT NULL AND b is NOT NULL then
 DBMS_output.Put_line ('Both Values are not
                                        Null');
    ELSE
    DBMS_output.Put_line ('At least one value
                                   is Null');
    END If;
 END;
/
BEGIN
 check_nullis;
END;
/
```

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
DECLARE
  V- name  VARCHAR2(20) = Rajesh
  BEGIN
   -- using % wildcard (matches any sequences of charact
If  V_name  Like `RA%.' THEN
    DBMS_output Put_Line ('Name Starts with
                                  RA');
    ENDIF
  -- using_ wildcard (matches ~~any~~ single character)
  If  v_name Like `R_JESH . THEN
     DBMS_OUTPUT.PUT_LINE ('Secondcharacter
                             is any single
                                   letter);
  ENDIF,
  -- using EScape character
   If 'A# B' Like 'A ~~#~~ %. ESCAPE '#' THEN
   DBMS -output . Put_Line (Escape characters
                             usedcorrectly');
    END If
   END;
```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```
a Number := 50;
b Number:= 30,
num _Smal NumBer
num _large Number

BeGin
    If    a < b Then
        num _small:=a;
        num _ large:=b;
    Else
        Num _Small:=b
        num - large:=a;
    END if;
    DBms _Output.putt _liner ('Small Number'|| num_small
D Bms_ output.Put _ line'('large Number'.|| num_larg
END;
/
```

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

```
Create or Replace PROCEDURE calc_incentive S
    V_emp id employee . employee _id%.type:= 110;
   V_ target Number = 800000;
    V_ sales Number = 90000;
     V_ incentive Number;
 BEGIN
  If  v_sals >= V_ target then
        v_ incentive := V_sale * 0.05; --5% incenti
      update employes
            SET Salary = Salary + V_incentive
         Where  employee _ id = v_emp_id;
   If SQL%. Row Count >0 Then
         DBms_Output .Put_Line ('Record updated
                                    Incentive added.'
    Else
      DBms_output .Put_line ("Employe not Fo
    END If

 Else
 DBMS. Output Put_Line ('Target not achieved. 
                                   incentive)
  END If,
  END;
  /
    BEGIN
```

162

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
Create or replac procedur calc_incentive IS

        Sales Number := 80000;
            Incentive number

    BEGIN
        If sales >= 100000 Then
            Incentive := sales * 0.10;
        Else if sales >= 50000 Then
            Incentive := Sales * 0.05;

      ELSE
        incentive := 0

      END if.
   DBMS_Output.Put_Line(sales||(sales);
  DBMS_Output Put_Line(incentive:||incentive
   END;

  /

  BEGIN
        calc_incentive;

        END;

    /
```

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
DECLARE
    V_Count Number;
    V_vacancies Number:=45;
      BEGIN
      Select Count (*) INTO V_Count
           from employees.
             where department_id=50;
     If V_Count <V_vacancies Then
  DBMS_output.Put_Line ('Vacancies availab
                                  (V_Vacanies-V_C
  Else.
     DBMS_output Put_Line ('NO Vacancies in d
                                 ment 50');
     END IF;
     END;
```

164

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
DECLARE
  v_dept_id Number:=60,
  v_total_positions Number =50,
  v_emp_count Number;
BEGIN.
 Select Count (*) into v_emp_count
   from employees
   where department_id = v_dept_id
IF   v_emp_count < v_total_positions then
      DBMS_output.Put_line('Vacancies available:'||(v_total_posi
      v_emp_count));
  ELse,
  DBMS_OutPut.Put_LINE ('NO Vacancies in Departme
                        || V_dept_id);
     END if
   END;

                       115
```

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

```
DECLARE
  Cursor emp_cur IS
        SELECT employee_id, first_name, job_id, hire_dat
 , Salary
        from employees;
          v_emp emp_cur %. Rowtype;
   BEGIN
        open emp_cur.
        LOOP
          FETCH emp_cur INTO v_emp;
            Exit when emp_cur%.NOT found;
  DBMS. output. Put_Line ('ID:|| v_emp.emplo
                  ; Name: '|| v_emp. first name||.
                  ; Job: '|| v_emp..job_id||
                  ; 'Hire Date : '|| v_emp.hire d
                  ; Salary : || v_emp.salary)

        END LOOP;
       close emp_cu
       END.;
```

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

```
DECLARE
    CURSOR emp_cur IS
        SELECT e.employee_id , e.first_name ,d.department_
                                           name.
          From employees e
          JOIN departments d
          ON e.department_id = d.department_id ;
    V_rec emp_cur%.ROWTYPE

BEGIN
    Open emp_cur

    Loop
        FETCH emp_cur INTO V_rec
        Exit when emp_cur%.NOT Found ;
        DBms_output_line('ID : '|| V_rec
            ; Name || V_rec first_name||
                    ; Department : || V_rec depa

    END LOOP ;
    Close emp_cur ;
END ;
```

[illegible]

PROGRAM 13

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
DECLARE
 Cursor job_cur is
   Select job_id, job_title, min salary
    From jobs;
    v_job job_cur%Rowtype;
    BEGIN
        Open job_cur;
        Loop
        FETCH job_cur INTO v_job;
         Exit when job_cur%NOTFOUND;
    DBMS_output.Put_Line('JobID:'||v_j
                             ; title || v-job job_
                             ; Min Salary:'||'V-job.m

    END loop;
    close job_cur
    END;
    /
```

PROGRAM 14

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
DECLARE
cursor emp_cur IS
    Select e.employee_id , e.first_name , jh.start_date
    From employees e
     Join job_history jh
      ON e.employee_id = jh.employee_id;

     V_rec emp_cur%.ROWTYPE;

      BEGIN
       OPEN emp_cur;
      Loop
     FETCH emp_cur INTO V_rec;
      Exit when emp_cur%.NOTFOUND;
       DBMS_output.put_line ('ID '|| V_rec.employee_id
                              ; Name '|| V_rec.first_name
                              ; Start Date: '|| V_rec.start
                                                    _date);
    END Loop;
    close emp_cur;
    END
```

169

PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

```
DECLARE
cursor emp_cu IS
      select e.employee_id, e.firstname, jh.end_date
       From employee e
       join job_history jh
        on e.employee_id = jh.employee_id;
         v_rec emp_cu%Rowtype
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	
Program/Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

```
BEGIN
      open emp_cu
      LOOP
     fetch emp_cu INTO v_rec;
      Exit when emp_cu%Notfound;
       DBMS_Output_Line ('ID:'||v_rec
                    ; Name:'||v_rec.firstname
   END LOOP;           ) End Date ||v_rec.end
END; Close emp_cu;
```