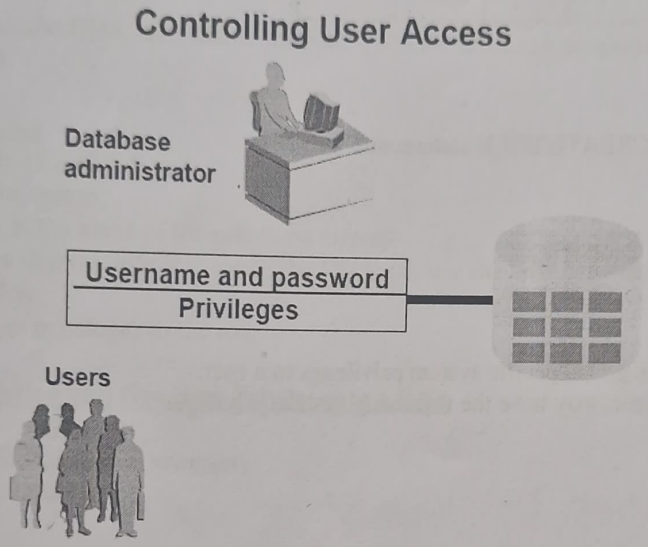
EXERCISE-15

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.

Object Privileges

Object Privilege	Table	View	Sequence	Procedure
ALTER	√		√	
DELETE	√	√		
EXECUTE				√
INDEX	√			
INSERT	√	√		
REFERENCES	√	√		
SELECT	√	√	√	
UPDATE	√	√		

Object Privileges

- Object privileges vary from object to object.

- An owner has all the privileges on the object.

- An owner can give specific privileges on that owner's object.

GRANT *object_priv* [(*columns*)]
ON *object*
TO {*user*|*role*|PUBLIC}
[WITH GRANT OPTION];

In the syntax:

object_priv is an object privilege to be granted

ALL specifies all object privileges

columns specifies the column from a table or view on which privileges are granted

ON *object* is the object on which the privileges are granted

TO identifies to whom the privilege is granted

PUBLIC grants object privileges to all users

WITH GRANT OPTION allows the grantee to grant the object privileges to other users and roles

Granting Object Privileges

- Grant query privileges on the EMPLOYEES table.

- Grant privileges to update specific columns to users and roles.

GRANT select
ON employees
TO sue, rich;

```
GRANT update (department_name, location_id)
ON departments
TO scott, manager;
```

Using the WITH GRANT OPTION and PUBLIC Keywords

- Give a user authority to pass along privileges.

- Allow all users on the system to query data from Alice's DEPARTMENTS table.

```
GRANT select, insert
ON departments
TO scott
WITH GRANT OPTION;
```

.

```
GRANT select
ON alice.departments
TO PUBLIC;
```

How to Revoke Object Privileges

- You use the REVOKE statement to revoke privileges granted to other users.

- Privileges granted to others through the WITH GRANT OPTION clause are also revoked.

```
REVOKE {privilege [, privilege...]|ALL}
ON object
FROM {user[, user...]|role|PUBLIC}
[CASCADE CONSTRAINTS];
```

In the syntax:

CASCADE is required to remove any referential integrity constraints made to the CONSTRAINTS object by means of the REFERENCES privilege

Revoking Object Privileges

As user Alice, revoke the SELECT and INSERT privileges given to user Scott on the DEPARTMENTS
table.

```
REVOKE select, insert
ON departments
FROM scott;
```

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?

Privilege: CREATE SESSION, Type: system Priviledge

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

Privilege: CREATE TABLE

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

The table owner, any grantee who received the privilege WITH GRANT OPTION can also pass it on.

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?

use a ROLE

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

ALTER USER your_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 SELECT ON your_schema.DEPARTMENTS TO other_user;

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 DEPARTMENTS(dept_id, dept_name) VALUES(510, 'Human Resources'); COMMIT; SELECT * FROM other_schema.DEPARTMENTS;

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

SELECT * FROM USER_TABLES;

10. Revoke the SELECT privilege on your table from the other team. REVOKE SELECT ON your_schema.DEPARTMENTS FROM other_user;

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

```
DELETE FROM DEPARTMENTS WHERE
department_id = 500;
COMMIT;
```

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

PL/SQL

PL/SQL

Control Structures

In addition to SQL commands,PL/SQL can also process data usin flow of statements.the flow of control statements are classified into the following categories.

- Conditional control -Branching
- Iterative control - looping
- Sequential control

BRANCHING in PL/SQL:

Sequence of statements can be executed on satisfying certain condition .

If statements are being used and different forms of if are:

1.Simple IF

2.ELSIF

3.ELSE IF

SIMPLE IF:

Syntax:

```
IF condition THEN
   statement1;
   statement2;
END IF;
```

IF-THEN-ELSE STATEMENT:

Syntax:

```
IF condition THEN
   statement1;
ELSE
   statement2;
END IF;
```

ELSIF STATEMENTS:

Syntax:

```
IF condition1 THEN
  statement1;
```

```
ELSIF condition2 THEN
  statement2;
ELSIF condition3 THEN
  statement3;
ELSE
  statementn;
END IF;
```

NESTED IF :

Syntax:

```
IF condition THEN
   statement1;
ELSE
  IF condition THEN
    statement2;
  ELSE
    statement3;
  END IF;
END IF;
ELSE
  statement3;
END IF;
```

SELECTION IN PL/SQL(Sequential Controls)

SIMPLE CASE

Syntax:

```
CASE SELECTOR
   WHEN Expr1 THEN statement1;
   WHEN Expr2 THEN statement2;
:
```

ELSE

Statement n;

END CASE;

SEARCHED CASE:

CASE

WHEN searchcondition1 THEN statement1;

WHEN searchcondition2 THEN statement2;

:

:

ELSE

statementn;

END CASE;

ITERATIONS IN PL/SQL

Sequence of statements can be executed any number of times using loop construct.

It is broadly classified into:

- Simple Loop
- For Loop
- While Loop

SIMPLE LOOP

Syntax:

LOOP

statement1;

EXIT [WHEN Condition];

END LOOP;

WHILE LOOP

Syntax:

WHILE condition LOOP

statement1;

statement2;

END LOOP;

FOR LOOP

Syntax:

```
FOR counter IN [REVERSE]
   LowerBound..UpperBound
LOOP
statement1;
statement2;
END LOOP;
```

PROGRAM 1

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

```
DECLARE

    v_emp_id employees.employee_id%TYPE := 110;
    v_salary employees.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_incentive);

EXCPEPTION:

    WHEN NO_DATA FOUND THEN

DBMS_OUTPUT.PUT_LINE ('employee not found.');
WHEN OTERS THEN

    DBMS_OUTPUT.PUT_LINE ('Error:' || SQL ERROR);

END:

/
```

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 := 100;
BEGIN
      DBMS_OUTPUT.PUT_LINE (MyVar);
   DBMS_OUTPUT.PUT_LINE ("Myvar");
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 employees.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 successfully
        for employee ID: '||v_emp_id): EXCEPTION
WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Employee not found.');
WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: '||SQLERRM);
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 checknulls IS
  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.PUTLINE ('At least one value is NULL');
 END IF;
END;
/
BEGIN:
    Check_Nulls;
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
    IF v_name LIKE 'RAY.' THEN
  DBMS_OUTPUT.PUT_LINE ('Name starts with RA');
  END IF;
  IF v_name LIKE 'R_JESH ' THEN
  DBMS_OUTPUT.PUT_LINE ('second character is any single letter');
  END IF;
  IF 'A#_B' LIKE 'A#_Y' ESCAPE '#' THEN
      DBMS_OUTPUT.PUT_LINE ('escape character used
                              correctly');
    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.

```
DECLARE
        a NUMBER := 50;
        b NUMBER := 30;
        NUM_Small 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.PUTLINE ('Small Number : ' || num_small);
DBMS_OUTPUT.PUTLINE ('Large Number : ' || NUM_large);
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.

```
BEGIN
    UPDATE employees
  SET salary = salary + (salary * 0.05)
 WHERE employee_id = 110 AND sales >= target;
IF SQL %ROWCOUNT > 0 THEN
   DBMS_OUTPUT.PUT_LINE (' Incentive added. ');
 ELSE
DBMS_OUTPUT.PUT_LINE ('Record not updated');
ENDIF;
END;
/
```

PROGRAM 8

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

```
CREATE OR REPLACE PROCEDURE cale_incentive IS
  sales NUMBER := 80000;
  incentive NUMBER;
BEGIN
   IF sales >= 100000 THEN
     incentive := sales * 0.10;
   ELSEIF sales >= 50000 THEN
      incentive := sales * 0.05
   ELSE
        incentive := 0;
      ENDIF;
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 available: '|| (v_vacancies-
  ELSE
DBMS_OUTPUT.PUT_LINE('No vacancies in department 50');
     END IF;
END;
```

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_positions - v_emp_count))
  ELSE
    DBMS_OUTPUT.PUTLINE('No vacancies in department:' ||
                v_dept_id);
    END IF;
    END;
    /
```

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_date, 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.employee_id ||
                      'Name: ' || v_emp.first_name ||
                      'Job: ' || v_emp.job_id ||
                      'Hire Date: ' || v_emp.hire_date ||
                      'Salary: ' || v_emp.salary);
    END LOOP;
    CLOSE emp_cur;
    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 department 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.PUT_LINE ('ID: ' || v_rec.employee_id ||
                              'Name: ' || v_rec.first_name ||
                              'Department: ' || v_rec.department_
                                                name);
    END LOOP;
    CLOSE emp_cur;
END;
/
```

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%NOT FOUND;

    DBMS_OUTPUT.PUT_LINE ('JobID: ' || v_job.job_id ||
                          'Title: ' || v_job.job_title ||
                          'Minsalary: ' || v_job.Minsalary);
    END LOOP;
    CLOSE job_cur;
```

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%NOT FOUND;
  DBMS_OUTPUT.PUT_LINE('ID: ' || v_rec.employee_id ||
                       ',Name: ' || v_rec.first_name ||
                       ',StartDate: ' || v_rec.start_date);
    END LOOP;
    CLOSE emp_cur;
  END;
  /
```

PROGRAM 15

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

```
BEGIN
    FOR REC IN (
    SELECT e.employee_id, e.first_name, jh.end_date
    FROM employees e JOIN job_history jh
    ON e.employees_id = jh.employee_id
    ) LOOP
      DBMS_OUTPUT.PUT_LINE ('ID: ' || rec.employee_id ||,
                            'Name: ' || rec.first_name ||,
                            'End Date: ' || rec.end_date );
    END LOOP;
);
```

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