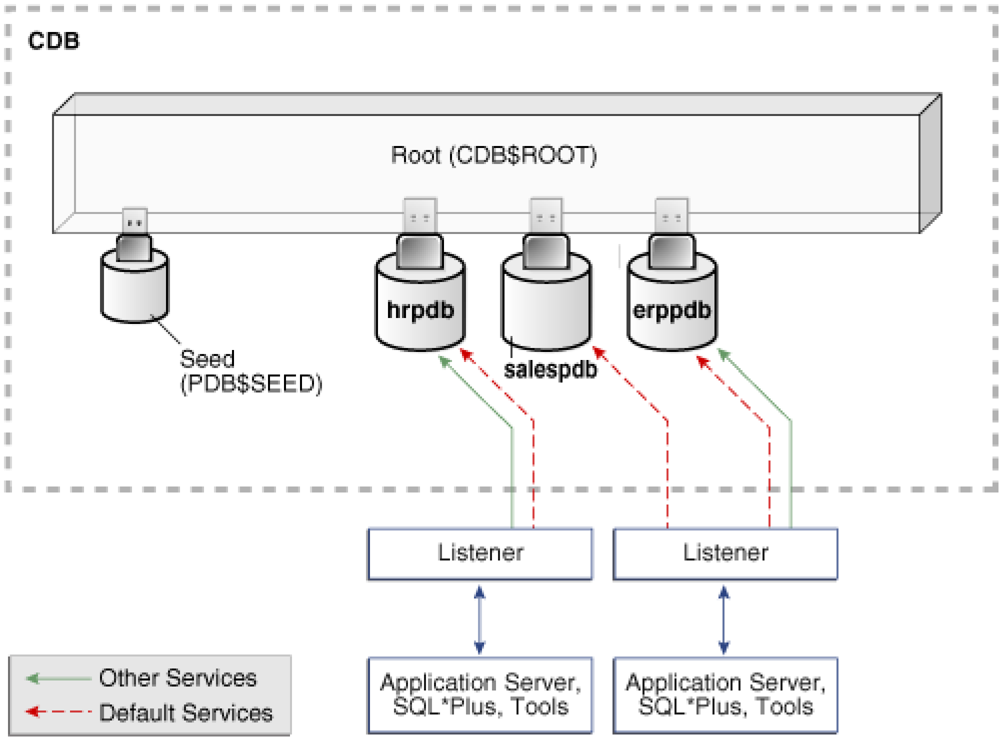
I. Overview of Containers in a CDB

A container is a collection of schemas, objects, and related structures in a multitenant container database (CDB) that appears logically to an application as a separate database. Within a CDB, each container has a unique ID and name.

The root and every **pluggable database (PDB)** is considered a container. PDBs isolate data and operations so that from the perspective of a user or application, each PDB appears as if it were a traditional non-CDB.

A **CDB** includes zero, one, or many customer-created pluggable databases (PDBs).

A **PDB** is a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a **non-CDB**. All Oracle databases before Oracle Database 12*c* were non-CDBs.



II. Connecting to a Container Database (CDB) from the command line

```
OS Authentication
Open the command prompt and then type:
$> sqlplus / as sysdba
```

```
--
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL>
```

You can connect to other common users in similar way.

```
Open the command prompt and then type:
$> sqlplus
SQL*Plus: Release 19.0.0.0.0 - Production on Thu Mar 26 10:08:48 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter user-name: sys/password as sysdba

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL>
```

SQL*Plus displays the SQL*Plus command prompt:

```
SQL>
```

Connect to another user:

```
SQL> CONN system/password
```

Or you can connect to DB:

```
SQL> -- EZCONNECT
SQL> CONN sys/password@localhost:1521/orcl as sysdba
Connected.
SQL> CONN sys/password@localhost/orcl as sysdba
Connected.


SQL> CONN system/password@localhost:1521/orcl
Connected.

SQL> -- tnsnames.ora
SQL> CONN sys/password@orcl as sysdba
Connected.

SQL> CONN system/password@orcl
Connected.
```

Displaying the Current Container

```
SQL> SHOW CON_NAME
```

III. PLUGGABLE DATABASE (PDB)

1. Create pluggable database

Use the `CREATE PLUGGABLE DATABASE` statement to create a pluggable database (PDB).

```
CREATE PLUGGABLE DATABASE orclpdb4 ADMIN USER adminpdb4 IDENTIFIED BY
pdbpassword ROLES=(DBA) file_name_convert=('pdbseed', 'orclpdb4');
```

2. Viewing Information about PDBs

```
SELECT PDB_NAME, STATUS FROM cdb_pdbs;
```

3. Viewing the Open Mode of Each PDB

```
SELECT NAME, OPEN_MODE, RESTRICTED, OPEN_TIME FROM V$PDBS;
```

4. Modifying the Open Mode of PDBs

```
ALTER PLUGGABLE DATABASE orclpdb4 OPEN;
```

5. Switching Between Containers

When logged in to the CDB as an appropriately privileged user, the `ALTER SESSION` command can be used to switch between containers within the container database.

```
ALTER SESSION SET CONTAINER=orclpdb4;
```

```
Session altered.
```

```
SQL> SHOW CON_NAME


SQL> ALTER SESSION SET CONTAINER=cdb$root;

Session altered.

SQL> SHOW CON_NAME
```

6. Connecting to a Pluggable Database (PDB)

```
SQL> -- EZCONNECT
SQL> connect adminpdb4/pdbpassword@localhost:1521/orclpdb4
```

```
SQL> CONN system/password@localhost:1521/orclpdb4
Connected.
SQL>
```

```
SQL> -- tnsnames.ora
SQL> CONN system/password@orclpdb4 --you have to create a service naming in Oracle
Net Manager
Connected.
SQL>
```

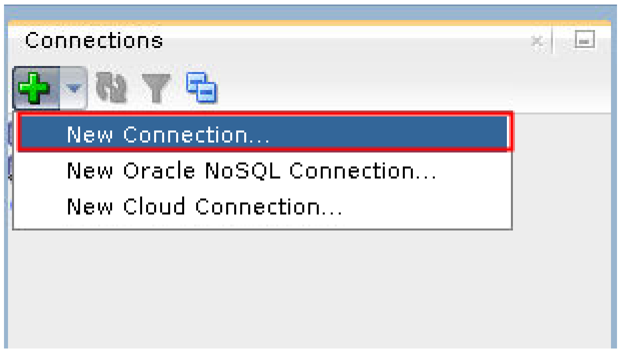
IV. Create a Database Connection Using SQL Developer

What Do You Need?

- Oracle Database 19c
- SQL Developer 19.2

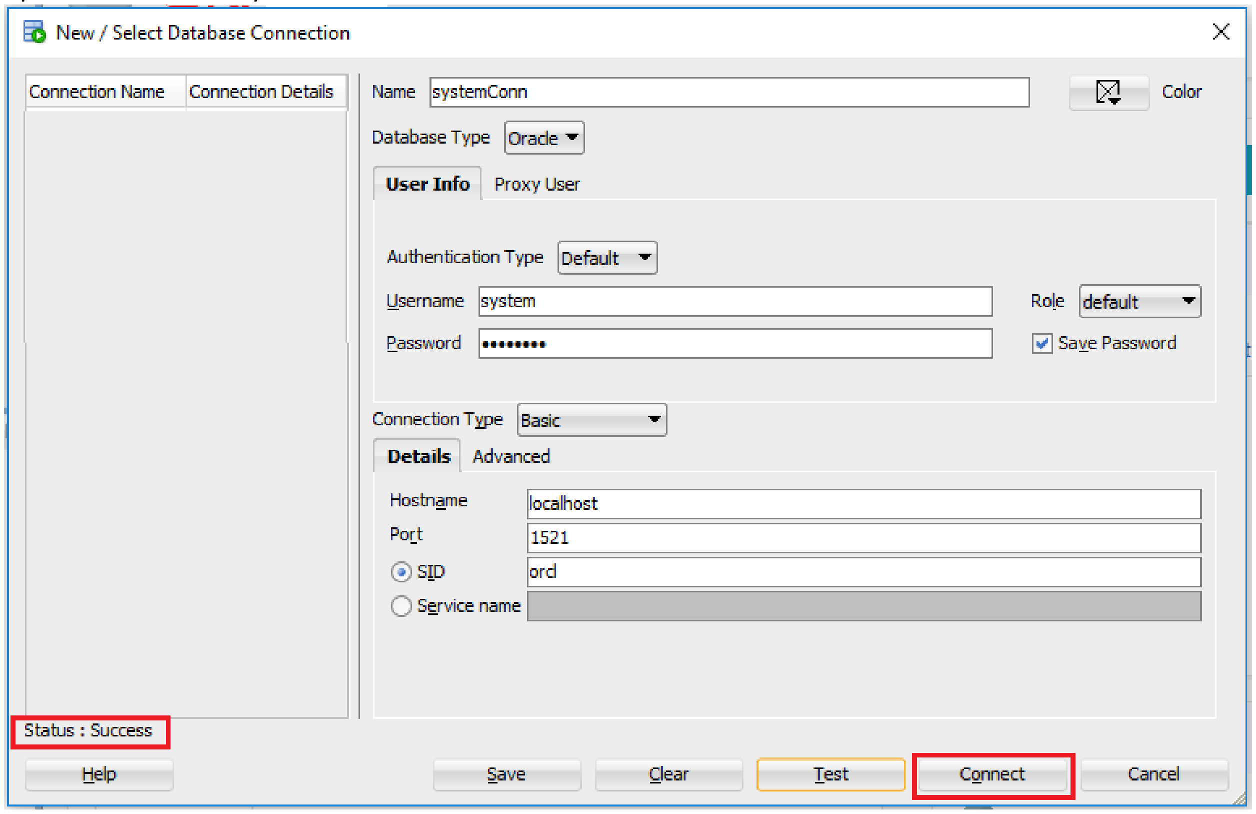
Create a Database Connection

1. Run sqldeveloper.exe
2. In the Connections navigator, right-click the Connections node and select **New Connection**.



3. Enter a connection name of your choice, username of system and password for the SYSTEM user. Select "**Save Password**" if you want to save your password for future connections as this user. Accept the default connection type and role. Enter the hostname, port, and SID. You can click **Test** to ensure that the connection works correctly. Click **Connect**.

4. Your connection is displayed in the Connections tab on the left side and a SQL worksheet is opened automatically.

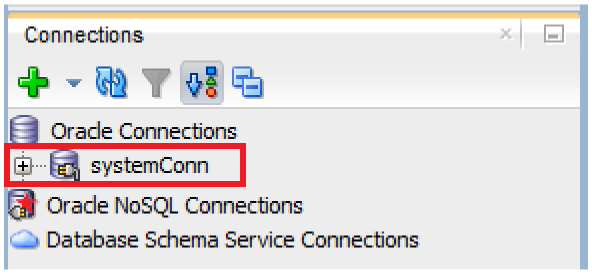

New / Select Database Connection

Connection Name | Connection Details

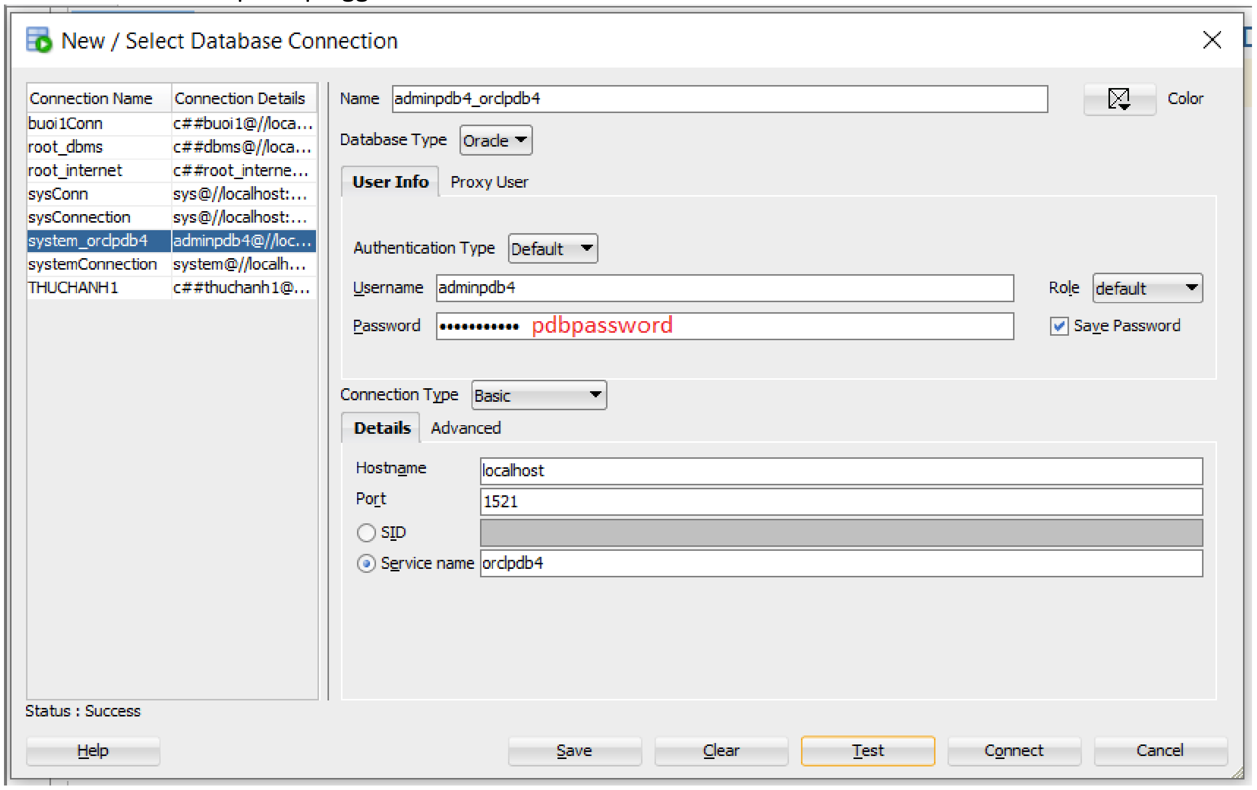
Name: systemConn
Database Type: Oracle
User Info | Proxy User
Authentication Type: Default
Username: system
Role: default
Password: ••••••••
[✓] Save Password
Connection Type: Basic
Details | Advanced
Hostname: localhost
Port: 1521
(●) SID: orcl
( ) Service name

Status : Success

5.



6. Connect to the orclpdb4 pluggable database:



New / Select Database Connection

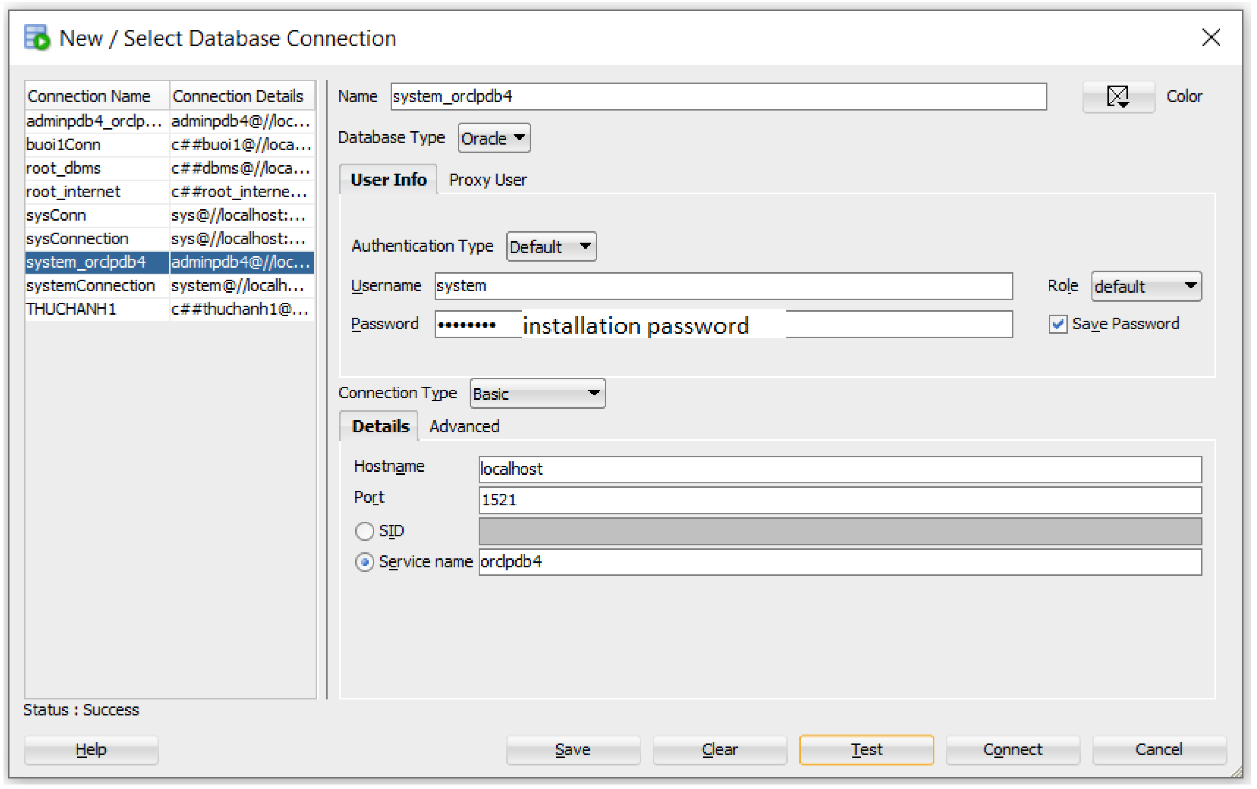
Connection Name	Connection Details
buoi1Conn	c##buoi1@//loca…
root_dbms	c##dbms@//loca…
root_internet	c##root_interne…
sysConn	sys@//localhost:…
sysConnection	sys@//localhost:…
system_orclpdb4	adminpdb4@//loc…
systemConnection	system@//localh…
THUCHANH1	c##thuchanh1@…

Name: adminpdb4_orclpdb4
Database Type: Oracle
User Info | Proxy User
Authentication Type: Default
Username: adminpdb4
Role: default
Password: ••••••••••• pdbpassword
Save Password ✓
Connection Type: Basic
Details | Advanced
Hostname: localhost
Port: 1521
SID
Service name: orclpdb4

Status : Success

Help | Save | Clear | Test | Connect | Cancel

7. Connect to the orclpdb4 pluggable database using system or sys user (of orclpdb4)



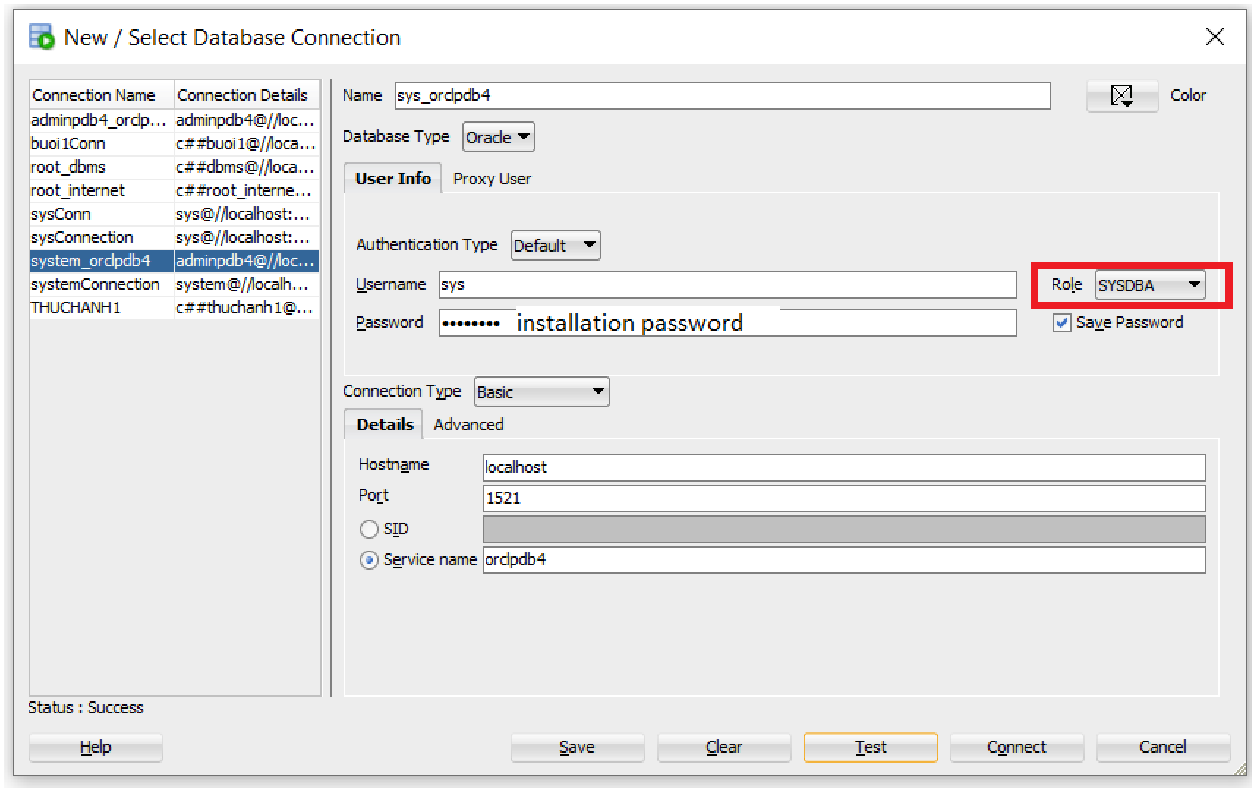
New / Select Database Connection

Connection Name	Connection Details
adminpdb4_orclp…	adminpdb4@//loc…
buoi1Conn	c##buoi1@//loca…
root_dbms	c##dbms@//loca…
root_internet	c##root_interne…
sysConn	sys@//localhost:…
sysConnection	sys@//localhost:…
system_orclpdb4	adminpdb4@//loc…
systemConnection	system@//localh…
THUCHANH1	c##thuchanh1@…

Name: system_orclpdb4
Database Type: Oracle
User Info | Proxy User
Authentication Type: Default
Username: system
Role: default
Password: •••••••• installation password
Save Password ✓
Connection Type: Basic
Details | Advanced
Hostname: localhost
Port: 1521
SID
Service name: orclpdb4

Status : Success

Help | Save | Clear | Test | Connect | Cancel


New / Select Database Connection

Connection Name	Connection Details
adminpdb4_orclp...	adminpdb4@//loc...
buoi1Conn	c##buoi1@//loca...
root_dbms	c##dbms@//loca...
root_internet	c##root_interne...
sysConn	sys@//localhost:...
sysConnection	sys@//localhost:...
system_orclpdb4	adminpdb4@//loc...
systemConnection	system@//localh...
THUCHANH1	c##thuchanh1@...

Name: sys_orclpdb4
Database Type: Oracle
User Info | Proxy User
Authentication Type: Default
Username: sys
Role: SYSDBA
Password: •••••••• installation password
Save Password [✓]
Connection Type: Basic
Details | Advanced
Hostname: localhost
Port: 1521
SID
Service name [●]: orclpdb4

Status : Success

Creating a Database User:

username password

```
CREATE USER sidney IDENTIFIED BY out_standing1;
```

Creating a Common User in a CDB

```
CREATE USER c##comm_user IDENTIFIED BY comm_pwd
```

Note: viewing container name: `SHOW CON_NAME`

Creating a Database User: default tablespace

This statement creates tablespace `tbs_03` with one datafile

```
CREATE TABLESPACE tbs_03

   DATAFILE 'C:\Oracle\oradata\ORCL\orclpdb4\tbs_f03_1.dbf' SIZE 20M;



ALTER TABLESPACE tbs_03 ADD DATAFILE
```

```
    'C:\Oracle\oradata\ORCL\orclpdb4\tbs_f03_2.dbf' SIZE 1M;


CREATE USER user1 identified by password

default tablespace tbs_03

QUOTA unlimited on tbs_03;
```

DROP a pluggable database:

```
--connect to sys of Database

ALTER PLUGGABLE  DATABASE orclpdb4 CLOSE IMMEDIATE;

drop pluggable database orclpdb4 including datafiles;
```

Do exercise 1.

V. About Privileges and Roles

Authorization permits only certain users to access, process, or alter data; it also creates limitations on user access or actions.
The limitations placed on (or removed from) users can apply to objects such as schemas, entire tables, or table rows.
A user **privilege** is the right to run a particular type of SQL statement, or the right to access an object that belongs to another user, run a PL/SQL package, and so on. The types of privileges are defined by Oracle Database.
Roles are created by users (usually administrators) to group together privileges or other roles. They are a way to facilitate the granting of multiple privileges or roles to users.

- **System privileges.** These privileges allow the grantee to perform standard administrator tasks in the database. Restrict them only to trusted users
- **Object privileges.** Each type of object has privileges associated with it.
- **User roles.** A **role** groups several privileges and roles, so that they can be granted to and revoked from users simultaneously.

1. System privileges

A system privilege is the right to perform a particular action or to perform an action on **any object** of a particular type. Objects include tables, views, materialized views, synonyms, indexes, sequences, cache groups, replication schemes and PL/SQL functions, procedures and packages.

Some system privileges:

Privilege	Description

`ALTER ANY PROCEDURE`	Enables a user to alter any PL/SQL procedure, function or package in the database.
`ALTER ANY SEQUENCE`	Enables a user to alter any sequence in the database. **Note**: There is no `ALTER SEQUENCE` statement.
`ALTER ANY TABLE`	Enables a user to alter any table in the database.
`ALTER ANY VIEW`	Enables a user to alter any view in the database. **Note**: There is no `ALTER VIEW` statement.
`CREATE ANY PROCEDURE`	Enables a user to create a PL/SQL procedure, function or package owned by any user in the database.
`CREATE ANY SEQUENCE`	Enables a user to create a sequence owned by any user in the database.
`CREATE ANY TABLE`	Enables a user to create a table owned by any user in the database.
`CREATE ANY VIEW`	Enables a user to create a view owned by any user in the database.
`CREATE PROCEDURE`	Enables a user to create a PL/SQL procedure, function or package owned by that user.
`CREATE SEQUENCE`	Enables a user to create a sequence owned by that user.
`CREATE TABLE`	Enables a user to create a table owned by that user.
`CREATE VIEW`	Enables a user to create a view owned by that user.
`DELETE ANY TABLE`	Enables a user to delete from any table in the database.
`DROP ANY PROCEDURE`	Enables a user to drop any PL/SQL procedure, function or package in the database.
`DROP ANY SEQUENCE`	Enables a user to drop any sequence in the database.
`DROP ANY TABLE`	Enables a user to drop any table in the database.
`DROP ANY VIEW`	Enables a user to drop any view in the database.
`DROP PUBLIC SYNONYM`	Enables a user to drop a public synonym.
`EXECUTE ANY PROCEDURE`	Enables a user to execute any PL/SQL procedure, function or package in the database.
`INSERT ANY TABLE`	Enables a user to insert into any table in the database. It also enables the user to insert into any table using the synonym, public or private, to that table.
`SELECT ANY TABLE`	Enables a user to select from any table, view, materialized view, or synonym in the database.
`UPDATE ANY TABLE`	Enables a user to update any table or synonym in the database.
……………….. .	………………….

2. Object privileges

An object privilege is the right to perform a particular action on an object or to access another user's object. Objects include tables, views, materialized views, indexes, synonyms, sequences, cache groups, replication schemes and PL/SQL functions, procedures and packages.

Privilege	Object type	Description
`DELETE`	Table	Enables a user to delete from a table.
`EXECUTE`	PL/SQL package, procedure or function	Enables a user to execute a PL/SQL package, procedure or function directly.
`FLUSH`	Cache group	Enables a user to flush a cache group.
`INDEX`	Table or materialized view	Enables a user to create an index on a table or materialized view.
`INSERT`	Table or synonym	Enables a user to insert into a table or into the table through a synonym.

`LOAD`	Cache group	Enables a user to load a cache group.
`REFERENCES`	Table or materialized view	Enables a user to create a foreign key dependency on a table or materialized view. The `REFERENCES` privilege `on` a parent table implicitly grants `SELECT` privilege on the parent table.
`REFRESH`	Cache group	Enables a user to refresh a cache group.
`SELECT`	Table, sequence, view, materialized view, or synonym	Enables a user to select from a table, sequence, view, materialized view, or synonym. The `SELECT` privilege enables a user to perform all operations on a sequence. A user can be granted the `SELECT` privilege on a synonym or a view without being explicitly granted the `SELECT` privilege on the originating table.
`UNLOAD`	Cache group	Enables a user to unload a cache group.
`UPDATE`	Table	Enables a user to update a table.

3. Roles

A **role** groups several privileges and roles, so that they can be granted to and revoked from users simultaneously. A role must be enabled for a user before it can be used by the user.

a. Creating a Role

```
CREATE ROLE salesclerk;
```

b. Dropping a Role

```
DROP ROLE clerk;
```

VI. GRANT and REVOKE

1. GRANT

Use the `GRANT` statement to grant:

- System privileges to users and roles.
- Roles to users, roles.
- Object privileges for a particular object to users and roles.

a. Grant system privileges to users and roles

```
-- Granting a System Privilege to a User

GRANT CREATE SESSION TO hr; --connect to database
```

```
-- Granting System Privileges to a Role

GRANT CREATE SESSION, SELECT ANY TABLE TO salesclerk;
```

b. Grant Roles to users, roles.

```
-- Granting a Role to a user

GRANT salesclerk TO smith;
```

```
-- Granting a Role to a Role

GRANT salesclerk TO dw_manager;
```

c. Granting a System Privilege and a Role to a User

```
GRANT CREATE SESSION, accts_pay TO jward;
```

Use of the **ADMIN Option** to Enable Grantee Users to Grant the Privilege

The **WITH ADMIN OPTION** clause can be used to expand the capabilities of a privilege grant.

These capabilities are as follows:

- The grantee can grant or revoke the system privilege or role to or from any other user or role in the database. Users cannot revoke a role from themselves.
- The grantee can grant the system privilege or role with the ADMIN option.
- The grantee of a role can alter or drop the role.

```
GRANT new_dba TO michael WITH ADMIN OPTION;
```

User **michael** is able to not only use all of the privileges implicit in the new_dba role, but he can also grant, revoke, and drop the new_dba role as deemed necessary.

d. Granting Object Privileges to Users and Roles

You can grant object privileges to users and roles, and enable the grantee to grant the privilege to other users.

The following example grants the READ, INSERT, and DELETE object privileges for all columns of the emp table to the users jfee and tsmith.

```
GRANT READ, INSERT, DELETE ON emp TO jfee, tsmith;
```

To grant all object privileges on the salary view to user jfee, use the ALL keyword as shown in the following example:

```
GRANT ALL ON salary TO jfee;
```

➢ **WITH GRANT OPTION**

Specify WITH GRANT OPTION to enable the grantee to grant the **object privileges** to other users and roles.

The WITH GRANT OPTION clause with the GRANT statement can enable a grantee to grant **object privileges** to other users.

User adams possesses the GRANT ANY OBJECT PRIVILEGE system privilege. He does not possess any other grant privileges. He issues the following statement:

```
GRANT SELECT ON HR.EMPLOYEES TO blake WITH GRANT OPTION;
```

If you examine the DBA_TAB_PRIVS view, then you will see that hr is shown as the grantor of the privilege:

```
SELECT GRANTEE, GRANTOR, PRIVILEGE, GRANTABLE
  FROM DBA_TAB_PRIVS
  WHERE TABLE_NAME = 'EMPLOYEES' and OWNER = 'HR';

GRANTEE  GRANTOR PRIVILEGE    GRANTABLE
-------- ------- -----------  ----------
BLAKE    HR       SELECT      YES
```

Now assume that user blake also has the GRANT ANY OBJECT PRIVILEGE system. He issues the following statement:

```
GRANT SELECT ON HR.EMPLOYEES TO clark;
```

In this case, when you query the DBA_TAB_PRIVS view again, you see that blake is shown as being the grantor of the privilege:

```
GRANTEE  GRANTOR  PRIVILEGE  GRANTABLE
-------- -------- ---------  ----------
BLAKE    HR       SELECT     YES
CLARK    BLAKE    SELECT     NO
```

The following statement grants the INSERT privilege on the acct_no column of the accounts table to user psmith:

```
GRANT INSERT (acct_no) ON accounts TO psmith;
```

In the following example, object privilege for the `ename` and `job` columns of the `emp` table are granted to the users `jfee` and `tsmith`:

```
GRANT INSERT(ename, job) ON emp TO jfee, tsmith;
```

2. REVOKE

a. Revokes of System Privileges and Roles

Any user with the `ADMIN` option for a system privilege or role can revoke the privilege or role from any other database user or role. The revoker does not have to be the user that originally granted the privilege or role. Users with `GRANT ANY ROLE` can revoke *any* role.

Revoking a System Privilege and a Role from a User

```
REVOKE CREATE TABLE, accts_rec FROM psmith;
```

b. Revokes of Object Privileges

Assuming you are the original grantor of the privilege, the following statement revokes the `SELECT` and `INSERT` privileges on the `emp` table from users `jfee` and `psmith`:

Revoking object Privileges from two User

```
REVOKE SELECT, INSERT ON emp FROM jfee, psmith;
```

The following statement revokes all object privileges for the `dept` table that you originally granted to the `human_resource` role:

```
REVOKE ALL ON dept FROM human_resources;
```

Revokes of Column-Selective Object Privileges

```
REVOKE UPDATE ON dept FROM human_resources;

GRANT UPDATE (dname) ON dept TO human_resources;
```

VII. User Privilege and Role Data Dictionary Views

A database administrator (DBA) for Oracle can simply execute a query to view the rows in **DBA_SYS_PRIVS**, **DBA_TAB_PRIVS**, and **DBA_ROLE_PRIVS** to retrieve information about user privileges related to the system, tables, and roles, respectively.

Query to List All System Privilege Grants

The `DBA_SYS_PRIVS` data dictionary view returns all system privilege grants made to roles and users.

```
SELECT GRANTEE, PRIVILEGE, ADMIN_OPTION FROM DBA_SYS_PRIVS;
```

Query to List All Role Grants

The `DBA_ROLE_PRIVS` query returns all the roles granted to users and other roles.

```
SELECT * FROM DBA_ROLE_PRIVS;
```

Query to List Object Privileges Granted to a User

The `DBA_TAB_PRIVS` data dictionary view returns all object privileges (not including column-specific privileges) granted to the specified user.

```
SELECT TABLE_NAME, PRIVILEGE, GRANTABLE FROM DBA_TAB_PRIVS

WHERE GRANTEE = 'JWARD';
```

To list all the column-specific privileges that have been granted, you can use the following query:

```
SELECT GRANTEE, TABLE_NAME, COLUMN_NAME, PRIVILEGE

    FROM DBA_COL_PRIVS;
```

QUERYING THE CURRENT USER'S PRIVILEGES

If DBA access isn't possible or necessary, it is also possible to slightly modify the above queries to view the privileges solely for the **current user**.

This is done by alternatively querying USER_ versions of the above DBA_ views. Thus, instead of looking at DBA_SYS_PRIVS we'd query **USER_SYS_PRIVS**, like so:

SELECT *** FROM** USER_SYS_PRIVS;

Query to List the Current Privilege Domain of Your Session

The `SESSION_ROLES` and `SESSION_PRIVS` data dictionary views list the current privilege domain of a database session.

```
SELECT * FROM SESSION_ROLES;
```

```
SELECT * FROM SESSION_PRIVS;
```

Query to List Roles of the Database

```
SELECT * FROM DBA_ROLES;
```

EXERCISE

I. **Exercise 1: Do the following things using sqlplus and write the results into your answer sheet.**

1. Open sqlplus and connect to sys user.
2. Show the connection name of current container.
3. Create a pluggable database (PDB) and open this PDB.
4. Connect to the sys user (or ADMIN user of this PDB) in the above PDB.
5. Show the connection name of current container.
6. In this PDB, create 3 user: user1, user2, user3.

II. **Exercise 2: Do the following things using sqlplus and write the results into your answer sheet.**

7. Connect to user1.
8. Connect to sys user (or ADMIN user) of PDB above.
9. Create a role named manager.
10. Grant CREATE SESSION, CREATE TABLE to manager role WITH ADMIN OPTION.
11. Grant manager role to user1.

12.Connect to user1.
13.Create a TEST table (ID NUMBER, NAME VARCHAR2(100))
14. Grant manager role to user2.
15. Grant CREATE SESSION privilege to user2;
16. Connect to user 2, create a table.
17. From user2, grant CREATE SESSION privilege to user3.
18. Connect to SYS user of this PDB.
19. Grant manager role to user3 WITH ADMIN OPTION.
20. Connect to user3.
21.Grant manager role to user2.
22.From user3, how can you create a table in user2 schema?
23.From user3, query the roles and privilege of current user.
24.Use SQLDeveloper to connect to ADMIN USER of this PDB.

III. Exercise 3: Using sqlplus to do the following things and write out the results:

1. Start SQL*Plus and connect to the Database.
2. CREATE a PLUGGABLE DATABASE (PDB) and open this PDB.
3. Using sys user to connect to the above PDB.
4. Show connection name.
5. Creating a tablespace.
6. CREATE 4 USERs: user1, user2, user3, user4, default tablespace and quota 1M for each user on the tablespace above
7. Grant privilege to user1 and user4 so that these users can connect to the database.
8. Show privilege of user1.
9. Create a programing role and grant privilege CREATE SESSION, CREATE TABLE to this role.
10. Grant programing role to user2 with admin option.
11. Show the privileges of user2.
12. Connect to the database using user2.
13. From user2, create the employees table: ID, NAME, SALARY, DESCRIPTION.
14. From user2, insert 2 rows to employees table.
15. From user2, grant programing role to user3 and grant update (name, salary) on employees table to user3.

16. From user3, grant programing role to user1.
17. From user3, show privilege of user3.
18. From user3, query data from employees table.
19. From sys, grant select on employees to user1.
20. Show the privileges of user1.
21. From sys user, grant all on employee to user1.
22. From sys user, grant create sequence to user3
23. From user3, create table students (id, fullname, birthday)
24. From user3, Create sequence named student_seq
25. From user3, insert data into students table with ID is generated from the sequence.
26. How can user4 insert data into students table of user3?
27. How can user4 delete data from students table of user3?