Practice

Managing Local and Common Users

Practice Target

In this practice you will examine the principles that govern managing the common and local users in CDB and PDBs.

Specifically, you will perform the following:

- Manage common users
- Manage local users
- Manage common and local roles
- Grant privileges as common or local
- Enable common users to view information about specific PDBs

Practice Assumptions

This practice assumes that you have the virtual machine `srv1` available with its **CDB** snapshot.

Managing Common Users

In the following steps, you will examine the principles of creating and managing common users.

1. Open a Putty session to `srv1` as `oracle`

2. Verify the CDB database is up and running.

```
sqlplus / as sysdba
SELECT CDB FROM V$DATABASE;
```

3. Create a new PDB from the seed container. Give it the name `PDB2`.

```
CREATE PLUGGABLE DATABASE PDB2
 ADMIN USER pdb2admin IDENTIFIED BY ABcd##1234
 STORAGE (MAXSIZE 2G);

ALTER PLUGGABLE DATABASE PDB2 OPEN;
```

4. View all common users in the entire database.

Observe that, in a freshly created database, all common users are Oracle maintained.

```
set pagesize 80
col username format a30
SELECT DISTINCT USERNAME, ORACLE_MAINTAINED
FROM CDB_USERS WHERE COMMON='YES'
ORDER BY 1;
```

5. Create a common user named as `C##USER1` and grant commonly the `CREATE SESSION` privilege to it.

```
CREATE USER C##USER1 IDENTIFIED BY ABcd##1234 CONTAINER=ALL;
GRANT CREATE SESSION TO C##USER1 CONTAINER=ALL;
```

6. Connect to the root container as `C##USER1` user and then to `PDB2` using the same credentials.

The common user can connect to the root as well as to the PDBs.

```
connect C##USER1/ABcd##1234
connect C##USER1/ABcd##1234@//srv1/pdb2.localdomain
```

7. Try creating a local user named as `LUSER1` in the root container.

An error should be returned as you cannot create a local user in the root container.

```
connect / as sysdba
CREATE USER LUSER1 IDENTIFIED BY ABcd##1234 CONTAINER=CURRENT;
```

8. Create another common user (`C##USER2`) and grant `CREATE SESSION` as a local privilege to the user.

Granting a local privilege to a common user is not a recommended practice. It is added in this practice only for demonstration purpose.

```
CREATE USER C##USER2 IDENTIFIED BY ABcd##1234 CONTAINER=ALL;

-- because CONTAINER was not set, its default value is CURRENT
GRANT CREATE SESSION TO C##USER2;
```

9. Try connecting to the root and a PDB using the new common user.

The `CREATE SESSION` privilege was locally granted to the user. It takes effect only on the local container (which is the root).

```
conn C##USER2/ABcd##1234@//srv1/pdb2.localdomain
conn C##USER2/ABcd##1234@//srv1/oradb.localdomain
sho con_name
```

10. Drop the common user `C##USER2`

```
conn / as sysdba
DROP USER C##USER2;
```

Managing Local Users

In the following steps, you will examine the principles of creating and managing local users.

11. View all local users in the CDB.

```
col pdb_name format a10
SELECT U.USERNAME, P.PDB_NAME
FROM   CDB_USERS U, CDB_PDBS P
WHERE  U.CON_ID = P.CON_ID AND COMMON='NO'
ORDER BY 2,1;
```

12. Create a local user, `LUSER2`, in `PDB2`, and grant `CREATE SESSION` privilege to it.

```
ALTER SESSION SET CONTAINER=PDB2;
CREATE USER LUSER2 IDENTIFIED BY ABcd##1234;
GRANT CREATE SESSION TO LUSER2;
```

13. Try creating a common user in `PDB2`.

The statement fails because common users can only be created in the root.

```
CREATE USER C##USER2 IDENTIFIED BY ABcd##1234 CONTAINER=ALL;
```

14. As `LUSER2`, try connecting to `PDB1`, `PDB2`, and the root

`LUSER2` can connect only to `PDB2`.

```
conn LUSER2/ABcd##1234@//srv1/pdb1.localdomain
conn LUSER2/ABcd##1234@//srv1/pdb2.localdomain
conn LUSER2/ABcd##1234
```

Managing Common and Local Roles

In the following steps, you will examine the principles of creating and managing common and local roles and assigning them to the users.

15. List all the predefined roles in the database (all common and local roles of the root and PDBs).

```
connect / as sysdba

col role format a30
SELECT ROLE, COMMON, CON_ID FROM CDB_ROLES ORDER BY ROLE, CON_ID;
```

16. List the roles defined in the root.

All the root roles are common. You cannot create local roles in the root.

```
SELECT ROLE, COMMON FROM DBA_ROLES ORDER BY ROLE;
```

17. Create a common role named `C##ROLE1`. This role will be referred to as "the common role" in the rest of the practice.

```
CREATE ROLE C##ROLE1 CONTAINER=ALL;
```

18. Try creating a local role, named as `LROLE`, in root.

The command should fail. Local roles cannot be created in the root.

```
CREATE ROLE LROLE CONTAINER=CURRENT;
```

19. List all the predefined roles in `PDB2`.

```
connect system/ABcd##1234@//srv1/pdb2.localdomain

col role format a30
SELECT ROLE, COMMON FROM DBA_ROLES ORDER BY ROLE;
```

20. Create a local role in `PDB2`. This role will be referred to as "the local role" in the rest of the practice.

```
ALTER SESSION SET CONTAINER=PDB2;
CREATE ROLE LROLE_PDB2 CONTAINER=CURRENT;
SELECT ROLE FROM DBA_ROLES WHERE COMMON ='NO' ORDER BY ROLE;
```

21. Grant the common role to the common user from the root.

Note that the common role is granted locally to the common user. The granted role is applicable only in the root.

```
connect / as sysdba

GRANT C##ROLE1 TO C##USER1;
```

```
-- the COMMON column in this query is not a property of the role. It is how
--  the privilege was granted
col grantee format A16
col granted_role format A16
SELECT GRANTEE, GRANTED_ROLE, COMMON, CON_ID
FROM CDB_ROLE_PRIVS WHERE GRANTEE='C##USER1';

conn C##USER1/ABcd##1234

SELECT * FROM SESSION_ROLES;
```

22. Grant the common role to the common user from the root as common (to be applicable in all the containers)

```
conn / as sysdba

-- specifying CONTINAER=ALL makes the grant common
GRANT C##ROLE1 TO C##USER1 CONTAINER=ALL;

-- you should see the common grants in addition to the local grant
SELECT GRANTEE, GRANTED_ROLE, COMMON, CON_ID
FROM CDB_ROLE_PRIVS WHERE GRANTEE='C##USER1';

-- connect as the common user to the root and check out the applicable role
conn C##USER1/ABcd##1234
SELECT * FROM SESSION_ROLES;

-- connect as the common user to PDB2 and check out the applicable role
conn C##USER1/ABcd##1234@//srv1/pdb2.localdomain
SELECT * FROM SESSION_ROLES;
```

23. Revoke the common role from the common user so that the role cannot be used in any container.

```
conn / as sysdba

-- this revokes the common grant
REVOKE C##ROLE1 FROM C##USER1 CONTAINER=ALL;

-- the local grant is still applicable
SELECT GRANTEE, GRANTED_ROLE, COMMON, CON_ID
FROM CDB_ROLE_PRIVS WHERE GRANTEE='C##USER1';

-- revoke the local grant
REVOKE C##ROLE1 FROM C##USER1;

-- verify
SELECT GRANTEE, GRANTED_ROLE, COMMON, CON_ID
FROM CDB_ROLE_PRIVS WHERE GRANTEE='C##USER1';
```

```
-- verify again
conn C##USER1/ABcd##1234
SELECT * FROM SESSION_ROLES;

conn C##USER1/ABcd##1234@//srv1/pdb2.localdomain
SELECT * FROM SESSION_ROLES;
```

24. Grant the common role to a local user in `PDB2`.

```
connect SYSTEM/ABcd##1234@//srv1/pdb2.localdomain

GRANT C##ROLE1 TO LUSER2;

SELECT GRANTEE, GRANTED_ROLE, COMMON, CON_ID
FROM CDB_ROLE_PRIVS WHERE GRANTEE='LUSER2';

-- Test the connection as the local user.
conn LUSER2/ABcd##1234@//srv1/pdb2.localdomain
SELECT * FROM SESSION_ROLES;
```

25. Grant the local role to the local user in PDB2.

```
conn SYSTEM/ABcd##1234@//srv1/pdb2.localdomain

GRANT LROLE_PDB2 TO LUSER2;

SELECT GRANTEE, GRANTED_ROLE, COMMON, CON_ID
FROM CDB_ROLE_PRIVS WHERE GRANTEE='LUSER2';

-- test the connection as the local user
conn LUSER2/ABcd##1234@//srv1/pdb2.localdomain
SELECT * FROM SESSION_ROLES;
```

Granting the Privileges as Common or Local

In this section, you will manage privileges granted as common or local in the root and PDBs.

26. Check how the system privilege `CREATE SESSION` is granted to `C##USER1` and `LUSER2` users.

There is nothing called common privileges or local privileges. Privileges are neither common nor local, but they can be granted as common or as local.

```
conn SYSTEM/ABcd##1234

col grantee format a18
col privilege format a14
SELECT GRANTEE, PRIVILEGE, COMMON, CON_ID
FROM   CDB_SYS_PRIVS
WHERE  GRANTEE IN ('C##USER1', 'LUSER2');

conn SYSTEM/ABcd##1234@//srv1/pdb2.localdomain
SELECT GRANTEE, PRIVILEGE, COMMON
FROM DBA_SYS_PRIVS
WHERE GRANTEE IN ('C##USER1', 'LUSER2');
```

27. Grant **commonly** the system privileges `CREATE TABLE` and `UNLIMITED TABLESPACE` to the common user `C##USER1`.

```
conn system/ABcd##1234
GRANT CREATE TABLE, UNLIMITED TABLESPACE TO C##USER1 CONTAINER=ALL;

col grantee format a12
col privilege format a30
SELECT GRANTEE, PRIVILEGE, COMMON, CON_ID
FROM CDB_SYS_PRIVS
WHERE GRANTEE = 'C##USER1' ORDER BY 1,2;
```

28. Grant **locally** the system privilege `CREATE SEQUENCE` to `C##USER1` to be applicable in root only

```
conn system/ABcd##1234
GRANT CREATE SEQUENCE TO C##USER1 CONTAINER=CURRENT;

col grantee format a12
SELECT GRANTEE, PRIVILEGE, COMMON, CON_ID
FROM CDB_SYS_PRIVS
WHERE GRANTEE = 'C##USER1' AND PRIVILEGE ='CREATE SEQUENCE';
```

29. Grant the system privilege `CREATE SYNONYM` to `C##USER1` to be applicable in `PDB2` only (a privilege locally granted)

```
conn system/ABcd##1234@//srv1/pdb2.localdomain
GRANT CREATE SYNONYM TO C##USER1 CONTAINER=CURRENT;

col grantee format a18
SELECT GRANTEE, PRIVILEGE, COMMON, CON_ID
FROM CDB_SYS_PRIVS
WHERE GRANTEE = 'C##USER1' AND PRIVILEGE ='CREATE SYNONYM';
```

30. Grant the system privilege `UNLIMITED TABLESPACE` to local user `LUSER2` to be applicable in `PDB2` only. (a privilege locally granted)

```
conn SYSTEM/ABcd##1234@//srv1/pdb2.localdomain
GRANT UNLIMITED TABLESPACE TO LUSER2;

col grantee format a18
SELECT GRANTEE, PRIVILEGE, COMMON, CON_ID
FROM CDB_SYS_PRIVS
WHERE GRANTEE = 'LUSER2';
```

Enabling Common Users to View Information about Specific PDBs

In this section of the practice, you will manage the CONTAINER_DATA attributes of common users to control what information common users can view in the container data objects when connecting to the root.

First, let's understand the default behavior of the database when querying the container data objects.

31. Query the container data views in the database.

Those views are affected by the CONTAINER_DATA user attributes. When common users query any of those views, the users see only the rows allowed by the CONTAINER_DATA user attributes (if there are any).

```
conn / as sysdba
SELECT VIEW_NAME FROM CDB_VIEWS WHERE CONTAINER_DATA = 'Y';
```

32. Display the CONTAINER_DATA attributes defined in the database.

The output shows the Oracle maintained users can by default query about all the container data objects.

```
column username format a10
column default_attr format a7
column owner format a6
column object_name format a11
column all_containers format a3
column container_name format a10
set pages 100
set line 200

SELECT USERNAME, DEFAULT_ATTR, OWNER, OBJECT_NAME,ALL_CONTAINERS, CONTAINER_NAME
FROM CDB_CONTAINER_DATA
WHERE USERNAME NOT IN ('GSMADMIN_INTERNAL', 'APPQOSSYS', 'DBSNMP')
ORDER BY USERNAME;
```

33. As SYS, query the data files in all the PDBs.

The SYS user can see all the datafiles in all the containers.

```
SELECT CON_ID, COUNT(*) FROM CDB_DATA_FILES GROUP BY CON_ID ORDER BY CON_ID;
```

34. Create the common user named as C##USER2 and commonly grant the system privilege CREATE SESSION and the role DBA to the user on all the containers.

```
CREATE USER C##USER2 IDENTIFIED BY ABcd##1234 CONTAINER=ALL;
GRANT CREATE SESSION, DBA TO C##USER2 CONTAINER=ALL;
```

35. Display the CONTAINER_DATA attributes defined for the new common user.

No CONTAINER_DATA attributes are set for the new common user.

```
SELECT USERNAME, DEFAULT_ATTR, OWNER, OBJECT_NAME,ALL_CONTAINERS, CONTAINER_NAME
FROM CDB_CONTAINER_DATA
WHERE USERNAME ='C##USER2'
ORDER BY USERNAME;
```

36. As C##USER2, query the data files in all the PDBs.

C##USER2 sees only the datafiles of the current container.

```
conn c##user2/ABcd##1234
SELECT CON_ID, COUNT(*) FROM CDB_DATA_FILES GROUP BY CON_ID ORDER BY CON_ID;
```

37. Change the current container to PDB1 and query the data files again.

C##USER2 sees the datafiles of PDB1 only after switching the current container to it. This is the default behavior of querying container data objects for newly created common users.

```
ALTER SESSION SET CONTAINER=PDB1;
SELECT CON_ID, COUNT(*) FROM CDB_DATA_FILES GROUP BY CON_ID ORDER BY CON_ID;
```

38. Set the CONTAINER_DATA attributes for C##USER2 so that it can query all the containers from the root.

```
conn / as sysdba
ALTER USER C##USER2 SET CONTAINER_DATA=ALL CONTAINER=CURRENT;

-- verify the attributes are set:
SELECT USERNAME, DEFAULT_ATTR, OWNER, OBJECT_NAME,ALL_CONTAINERS, CONTAINER_NAME
FROM CDB_CONTAINER_DATA
WHERE USERNAME ='C##USER2'
ORDER BY USERNAME;
```

39. As C##USER2, query the data files in all the PDBs.

C##USER2 can see the datafiles of all the containers from the root.

```
conn c##user2/ABcd##1234
SELECT CON_ID, COUNT(*) FROM CDB_DATA_FILES GROUP BY CON_ID ORDER BY CON_ID;
```

We allowed C##USER2 to see the information in container data objects from the root. Let's see how we can restrict its visibility on the container data objects to only specific PDBs.

40. Remove the CONTAINER_DATA attributes of C##USER2.

```
conn / as sysdba
ALTER USER C##USER2 SET CONTAINER_DATA=DEFAULT CONTAINER=CURRENT;

-- verify the attributes are removed:
SELECT USERNAME, DEFAULT_ATTR, OWNER, OBJECT_NAME,ALL_CONTAINERS, CONTAINER_NAME
FROM CDB_CONTAINER_DATA
WHERE USERNAME ='C##USER2'
ORDER BY USERNAME;
```

41. Set the CONTAINER_DATA attributes for C##USER2 on only PDB1.

```
ALTER USER C##USER2 SET CONTAINER_DATA=(CDB$ROOT,PDB1) CONTAINER=CURRENT;

-- verify the attributes are set for c##user2:
SELECT USERNAME, DEFAULT_ATTR, OWNER, OBJECT_NAME,ALL_CONTAINERS, CONTAINER_NAME
FROM CDB_CONTAINER_DATA
WHERE USERNAME ='C##USER2'
ORDER BY USERNAME;
```

42. As C##USER2, query the data files in all the PDBs from the root.

C##USER2 can see the datafiles of only the root and PDB1.

```
conn c##user2/ABcd##1234
SELECT CON_ID, COUNT(*) FROM CDB_DATA_FILES GROUP BY CON_ID ORDER BY CON_ID;
```

43. Change the current container to PDB2 and try querying its data files.

The common user can see the PDB2 data files after switching the current container to it. This means CONTAINER_DATA attributes taken effect only when querying the container data objects from the root. We cannot rely on it alone to control what a common user can see about specific containers.

```
ALTER SESSION SET CONTAINER=PDB2;
SELECT CON_ID, COUNT(*) FROM CDB_DATA_FILES GROUP BY CON_ID ORDER BY CON_ID;
```

We allowed C##USER2 to see the information in container data objects for only the root and PDB1. Let's see how we can restrict its visibility on a specific PDB and in a specific container data object.

44. Remove the CONTAINER_DATA attributes of C##USER2.

```
conn / as sysdba
ALTER USER C##USER2 SET CONTAINER_DATA=DEFAULT CONTAINER=CURRENT;
```

45. Set the CONTAINER_DATA attributes for C##USER2 on only PDB1 and on the view CDB_DATA_FILES.

```
ALTER USER C##USER2 SET CONTAINER_DATA=(CDB$ROOT,PDB1) FOR CDB_DATA_FILES
CONTAINER=CURRENT;

-- verify the attributes are set for c##user2:
col OBJECT_NAME for a15
SELECT USERNAME, DEFAULT_ATTR, OWNER, OBJECT_NAME,ALL_CONTAINERS, CONTAINER_NAME
FROM CDB_CONTAINER_DATA
WHERE USERNAME ='C##USER2'
ORDER BY USERNAME;
```

46. As C##USER2, query the data files in all the PDBs from the root.

C##USER2 can see the datafiles of only the root and PDB1.

```
conn c##user2/ABcd##1234
SELECT CON_ID, COUNT(*) FROM CDB_DATA_FILES GROUP BY CON_ID ORDER BY CON_ID;
```

47. Query the tablespaces in all the PDBs from the root.

C##USER2 can see the tablespaces of only the root. This is expected because the CONTAINER_DATA attributes for C##USER2 includes only the CDB_DATA_FILES in PDB1.

```
SELECT CON_ID, COUNT(*) FROM CDB_TABLESPACES GROUP BY CON_ID ORDER BY CON_ID;
```

48. Clean up steps

```
conn / as sysdba

DROP USER C##USER1;
DROP USER C##USER2;
DROP ROLE C##ROLE1;
ALTER PLUGGABLE DATABASE PDB2 CLOSE IMMEDIATE;
DROP PLUGGABLE DATABASE PDB2 INCLUDING DATAFILES;
```

Summary

In this practice you have tested and examined the following basic rules that control managing common and local users, common and local roles, and granting the privileges commonly and locally.

Following are the summary points those rules:

- Common users can only be created on the root.
- Local users can only be created in a PDB.
- Common roles can only be created on the root.
- Local roles can only be created in a PDB.
- Common roles can be granted to common or local users.
- Common users can be granted common or local roles.
- A common role can be grated as common or as local (default) to a common user.
- Privileges can be granted as common (to common users) or as local.
- You can control which data can be seen by common users when they query the container dictionary objects from the root by setting the `CONTAINER_DATA` attributes for the users. However, if a PDB is not included in the `CONTAINER_DATA` attributes of a common user, the common user can still query the container data objects about that PDB after switching to the PDB.

Tips

- To avoid confusion, always use the `CONTAINER` clause when you issue the `GRANT` command.
- Use common users for management and maintenance tasks.
- Create common roles for only common users.
- Create local roles for only local users.