Practice 14

Creating a Multitenant RAC Database

Practice Overview

In this practice, you will perform basic tasks to manage a multitenant RAC database. Specifically, you will perform the following:

- Create a Multitenant RAC database
- Use different methods to connect to a CDB or PBD
- Use different methods to startup and shutdown a CDB or PBD
- Clone a PDB
- Manage a PDB service
- Drop a PDB

Practice Assumptions

- The practice assumes that you have the virtual machines `srv1` and `srv2` up and running.

Creating a CDB RAC database

In this section of the practice, you will create a new Oracle CDB RAC database.

1. In the VirtualBox window of `srv1`, login as `oracle`.

2. Start the `dbca` utility and use it to create a new CDB RAC database. Response to the utility windows as follows:

Utility Window	Response
Database Operations	Select **Create Database**
Creation Mode	Select **Advanced Mode**
Deployment Type	Database Type: **Oracle Real Application Cluster (RAC) Database** Configuration Type: **Admin Managed** Select the **General Purpose or Transaction Processing** template
Nodes Selection	Make sure all nodes are selected
Database Identification	Global Database Name: `mtdb.localdomain` SID: `mtdb` **Mark "Create as Container Database"** Select **User Local Undo tablespace for PDBs** Number of PDBs: **1** PDB name: **pdb1**
Storage Options	In the Database files storage type: select **Automatic Storage Management (ASM)** as the Storage Type. Enter **+DATA/{DB_UNIQUE_NAME}** in the Database File Locations field (can be selected by clicking on the **Browse** button). Select **Oracle-Managed Files**
Fast Recovery Option	Select **Specify Fast Recovery Area** and enter `+FRA` in the Fast Recovery Area field. Storage Type: **Automatic Storage Management (ASM)** Fast Recovery Area: **+FRA** Fast Recovery Area Size: **10240 MB** **Mark Enable Archiving**
Oracle Vault Option	click on **Next** button
Configuration Options	Under Memory tab: SGA Size: **979** PGA Size: **327**

	Under Sizing tab: Processes: **500**
Management Options	**Unmark** "Run Cluster Verification Utility..." **Unmark** "Configure Enterprise Manager (EM) Database Express"
User Credentials	Select "Use the same Administrative password." Enter `oracle` as the password.
Creation Options	Select **Create Database** and click **Next**
Pre-requisites Checks	Select **Ignore All**
Summary	Click on **Finish** button
Progress Page	Click on **Close** button

3. Start Putty and login to `srv1` as `oracle`.

4. Using the `srvctl` utility, check the status of the database and its configuration.

```
srvctl status database -db mtdb
srvctl config database -db mtdb
```

5. Switch the current user to `grid` and check the services registered in the listener.

Observe that both the CDB database (`mtdb`) and the pluggable database (`pdb1`) are registered in the listener.

```
su - grid
lsnrctl services
```

6. Check the services registered in the listener in `srv2`.

Observe that the same database services are registered in the listener.

```
ssh srv2
lsnrctl services
exit
```

Examining Connecting to CDB database

In this section of the practices, you will examine the different ways of connecting to the CDB root and the pluggable database.

7. Switch the current user to `oracle`.

8. Connect to the CDB container (`mtdb`) using the Easy Connect method and verify that the database is a multitenant database.

Observe that when you connect to the CDB container, you are technically connected to the root container. Typically, this container should not have any application data in it.

```
sqlplus system/oracle@//srv1/mtdb.localdomain

SHOW CON_NAME
SELECT NAME, CDB, CON_ID FROM V$DATABASE;
```

9. Connect to the pluggable database container (`pdb1`) using the Easy Connect method.

PDB is the database that application connect to. To the application perspective, the PDB looks and behaves exactly as the traditional Oracle non-CDB database.

```
sqlplus system/oracle@//srv1/pdb1.localdomain

SHOW CON_NAME
```

10. Login to the local root as `sysdba` then obtain the instance name and `CON_ID` of the current instance.

```
export ORACLE_SID=mtdb1
sqlplus / as sysdba

SELECT INSTANCE_NAME, CON_ID FROM V$INSTANCE;
```

11. Check the open mode of `pdb1` in all the instances.

```
col name format a10
SELECT INST_ID,CON_ID,NAME,OPEN_MODE FROM GV$PDBS WHERE NAME='PDB1';
```

12. Close `PDB1` in the current instance, check on which instance the PDB was closed then open it again.

Observe that the statement by default opens/closes the PDB in the current instance.

```
ALTER PLUGGABLE DATABASE PDB1 CLOSE;
SELECT INST_ID,CON_ID,NAME,OPEN_MODE FROM GV$PDBS WHERE NAME='PDB1';
ALTER PLUGGABLE DATABASE PDB1 OPEN;
```

13. Close `pdb1` in `mtdb2` instance.

The statement allows you to specify the instance in which you want to close the PDB.

```
ALTER PLUGGABLE DATABASE pdb1 CLOSE INSTANCES=('mtdb2');
```

14. Try connecting to `pdb1` using Easy Connect, first in the first node, then in the second node.

Observe connecting to the second node fails.

```
conn system/oracle@//srv1/pdb1.localdomain
conn system/oracle@//srv2/pdb1.localdomain
```

15. Open `pdb1` in `mtdb2` instance.

```
conn / as sysdba
ALTER PLUGGABLE DATABASE pdb1 OPEN INSTANCES=('mtdb2');
```

16. Switch the current container to the root and then retrieve information about the redo log groups.

You will notice that each instance in the CDB has four multiplexed redo groups (total 16) and they are all managed by the root container. Redo log groups are always associated to the CDB and they are used by all the PDBs within the CDB. You cannot create a redo group for a specific PDB.

```
ALTER SESSION SET CONTAINER=CDB$ROOT;

SELECT INST_ID, GROUP#, CON_ID FROM GV$LOGFILE ORDER BY 1,2,3;
```

17. Execute the following command to retrieve information about the undo tablespaces in the CDB.

```
# list of all the undo tablespaces in the CDB:
#  in a CDB, tablespace name is not unique. Tablespaces are uniquely identified
#  by their names and their CON_ID.

SELECT TABLESPACE_NAME, CON_ID
FROM CDB_TABLESPACES
WHERE CONTENTS = 'UNDO';

# to know which one is being used by the current instance:
show parameter UNDO_TABLESPACE

# Switch the current container to PDB1 and run the same queries above.
#  because the current container isn't the root, the views will retrieve
#  the information of the current container only
ALTER SESSION SET CONTAINER=PDB1;
```

Cloning a PDB in a RAC CDB

In this section of the practices, you will create a new container named `pdb2` by cloning `pdb1`.

18. Switch the current user to `oracle`.

19. Issue the following command to create a new container named `pdb2` by cloning `pdb1`.

Observe that with a single SQL command, you managed to create a new pluggable database by copying online another pluggable database. That was nearly impossible with non-CDB databases.

```
CREATE PLUGGABLE DATABASE pdb2 FROM pdb1;
```

20. Open `pdb2` by issuing the command `ALTER PLUGGABLE DATABASE ... OPEN`

When the `OPEN_MODE` is `MOUNTED` for a PDB, it means it is closed.

```
SELECT INST_ID, OPEN_MODE FROM GV$PDBS WHERE NAME='PDB2' ORDER BY 1;
ALTER PLUGGABLE DATABASE pdb2 OPEN INSTANCES=ALL;
SELECT INST_ID, OPEN_MODE FROM GV$PDBS WHERE NAME='PDB2' ORDER BY 1;
```

21. Verify that the new PDB is registered in the listener then try connecting to it.

```
host lsnrctl services | grep pdb2
conn system/oracle@//srv1/pdb2.localdomain
conn system/oracle@//srv2/pdb2.localdomain
```

Managing PDB services in a RAC CDB

In this section of the practices, you will create a service for pdb2 that will be managed by clusterware.

22. Connect to the local instance as sysdba then switch the current container to pdb2.

```
sqlplus / as sysdba

ALTER SESSION SET CONTAINER=pdb2;
```

23. Verify that a service exists that has the same name as the PDB.

```
col name format a30
col pdb format a10

SELECT NAME, PDB FROM DBA_SERVICES ORDER BY 1;
```

24. Restart the CDB and observe the open mode of pdb2.

Observe that PDB2 is not started. This is basically because PDB2 service is not registered in the clusterware.

```
srvctl stop database -d mtdb
srvctl start database -d mtdb

sqlplus / as sysdba

SELECT INST_ID, OPEN_MODE FROM GV$PDBS WHERE NAME='PDB2' ORDER BY 1;
```

25. Verify that the retrieved service name is not managed by the clusterware.

When the command displays nothing, it means no service is being managed by the clusterware.

```
srvctl status service -db mtdb
```

26. Start up pdb2 in all the instances.

```
ALTER PLUGGABLE DATABASE pdb2 OPEN INSTANCES=ALL;
```

27. Create and start a PDB service for pdb2. Set the first node as the preferred node for the service.

```
srvctl add service -db mtdb -pdb pdb2 -s pdb2srv -preferred mtdb1 -available
mtdb2
srvctl start service -db mtdb -s pdb2srv
```

28. Make sure the created service is registered in the listener.

```
lsnrctl  services | grep pdb2srv
```

29. Verify that the service will start automatically when you restart the system.

```
srvctl config service -db mtdb -s pdb2srv | grep "Management policy"
```

30. Try connecting to the `pdb2` via `pdb2srv` service, first in the first instance and then in the second instance.

The service is available in the first instance only, therefore, connecting to the second instance fails.

```
sqlplus system/oracle@//srv1/pdb2srv.localdomain
conn system/oracle@//srv2/pdb2srv.localdomain
```

31. Verify that restarting the CDB starts the pdb service.

```
srvctl stop database -d mtdb
srvctl start database -d mtdb
srvctl status service -db mtdb -s pdb2srv
```

Dropping the PDB

In this section of the practice, you will drop `pdb2` including its datafiles.

32. Issue the following command to delete the service associated with `pdb2`. The service must be stopped before you can delete it.

```
srvctl stop service -db mtdb -s pdb2srv
srvctl remove service -db mtdb -s pdb2srv

# verify that it is gone from the listener registered services:
lsnrctl  services | grep pdb2srv
```

33. Connect to the local instance as `sysdba`, stop `pdb2`, then drop it including its datafiles.

Observe that while you are dropping `pdb2`, the other pdbs are still online and in operation.

```
sqlplus / as sysdba

ALTER PLUGGABLE DATABASE pdb2 CLOSE INSTANCES=ALL;
DROP PLUGGABLE DATABASE pdb2 INCLUDING DATAFILES;
```

Cleanup

34. Run the `dbca` utility and drop the `mtdb` database.

Summary

Multitenant architecture is a significant solution for database consolidation and it is definitely the future of how Oracle database will be developed. In this practice you learnt some fundamentals about Oracle multitenant database. Specifically, you performed the following:

- Create a Multitenant RAC database
- Use different methods to connect to a CDB or PBD
- Use different methods to startup and shutdown a CDB or PBD
- Clone a PDB
- Manage a PDB service
- Drop a PDB