Practice 3

Oracle RAC Administration Topics

Practice Overview

In this practice you will perform some common Oracle RAC Administration tasks. Specifically, you will perform the following:

- Use multiple connection methods to Oracle RAC database
- Use the methods to start and stop Oracle RAC instances
- Use the help system within the `srvctl` utility
- Examine the initialization parameters in Oracle RAC databases

Practice Assumptions

The practice assumes that you have the Oracle RAC database up and running in the virtual machines `srv1` and `srv2`.

Practice Procedures

A. Using multiple methods to connect to Oracle RAC database

Using different authentication methods

In this sub-section of the practice, you will test multiple authentication methods to connect to Oracle RAC database. Specifically, you will study the following:

- Operating system authentication method
- Password file authentication method
- Oracle database authentication method

Note: discussing all the aspects of the authentication methods is beyond the scope of this course. We are here concentrating on the RAC particularities.

1. In a Putty session, login to `srv1` as `oracle` user

2. Using SQL*Plus utility, connect to `rac` database as sysdba without providing the password. Check out to which instance the session has been connected.

```
sqlplus / as sysdba

SELECT INSTANCE_NAME FROM V$INSTANCE;
```

This type of connection is called "operating system authenticated" connection. Consider the following about this connection type:

- It does not require a password.
- It always connects to the local instance.
- An operating system account is needed for the user.
- For an Oracle RAC database, it requires the `ORACLE_SID` variable to set to the local instance name.

3. Set the `ORACLE_SID` variable to "`rac`" and try connecting to the database using the operating system authentication.

```
# display the current value of ORACLE_SID:
echo $ORACLE_SID

# set its value to rac:
export ORACLE_SID=rac

# try connecting to the database instance:
sqlplus / as sysdba

# set the variable value back again to the instance name:
export ORACLE_SID=rac1
```

The operating system connection fails if the `ORACLE_SID` variable is not set to the local instance name.

4. Try connecting to the local instance as sysdba using the password of sys.

The password of sys is saved in the password file. That is why this method of connection is called "password file authenticated" connection.

```
sqlplus sys/oracle as sysdba
```

5. Try connecting to the local instance as sysdba using a wrong password.

```
sqlplus sys/wrong as sysdba
```

The connection succeeds, although the password is incorrect. This is because the operating system authentication by default supersedes the password file authentication. Therefore, the command above actually used the operating system authentication to log in to the database.

If you want to force using the password file authentication, login via the listener. Login via the listener will be covered later in the practice.

6. Connect as system user. Once using a correct password and another time using an incorrect password.

```
conn system/oracle
conn system/wrong
```

This method of authentication is called database authenticated connection. In this method, the password is saved in the database and the password must be correct to log in.

7. Connect as system user with a wrong password but use the as sysdba option.

```
conn system/wrong as sysdba
show user
```

When you use the as sysdba, Oracle logs on as sys user, regardless of the username you used in your connection attempt. The previous command succeeds because Oracle used the operating system authentication.

8. Connect as sysdba using sys as a username and an incorrect password but use the tns naming method this time.

```
conn sys/wrong@rac as sysdba
```

Tns naming method supersedes any other connection method. This connection goes through the listener and Oracle considers this connection as a client connection, not a local connection.

Note: For further information, refer to the section "Configuring Authentication" in the documentation "Database Administration".

Examining the Connection Methods to Oracle RAC Database

In this sub-section of the practice, you will examine the following connection methods to Oracle RAC database:

- Use Easy Connect Naming method.
- Configure tnsnames.ora file (using the Local Naming method).

9. Make sure the EZCONNECT method is configured in the sqlnet.ora file.

```
host cat $TNS_ADMIN/sqlnet.ora
```

10. Using SQL*Plus utility, connect to rac database using Easy Connect method.

```
# repeat the following commands multiple times:
conn system/oracle@//srv-scan/rac.localdomain
SELECT INSTANCE_NAME FROM V$INSTANCE;
```

Following is the full format of the Easy Connect method:

```
username@[//]host[:port][/service_name][:server][/instance_name]
```

Note: Oracle recommends against using the easy connect method with SCAN host name because the easy connect method does not have the ability to specify timeouts and retries for connection establishment. Instead, applications should use an Oracle Net connect descriptor.

11. Use the easy connect method to connect to the instance rac1

```
# repeat the following commands multiple times:
conn system/oracle@//srv-scan/rac.localdomain:srv1/rac1
SELECT INSTANCE_NAME FROM V$INSTANCE;
```

You can configure the tnsnames.ora to connect to specific instances. In the following steps, you will configure the tnsnames.ora file to store connections to rac1 and rac2. The connections that you will configure does not use the load balancing feature. Each connection connects to specific instance and it fails if that instance is down and even if the other instance is up. That is the reason you do not use the SCAN name nor the VIP address to define the host name in such configuration.

12. In srv1 and srv2, add the following connect descriptors to the tnsnames.ora file.

Do not copy from the PDF file. Copy the code from the downloadable tnsnames.ora file.

```
vi $TNS_ADMIN/tnsnames.ora
```

```
RAC1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = srv1.localdomain)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = rac1)
    )
  )
```

```
RAC2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = srv2.localdomain)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = rac2)
    )
  )
```

13. Test the connection configurations that you made in the previous step. Verify that you are connected to the required instance.

```
sqlplus /nolog

conn system/oracle@rac1
SELECT INSTANCE_NAME FROM V$INSTANCE;

conn system/oracle@rac2
SELECT INSTANCE_NAME FROM V$INSTANCE;

exit
```

B. Use the Various Methods to Start and Stop Oracle RAC Instances

In this section of the practice, you will examine the methods to stop and startup an Oracle RAC instance and an Oracle RAC database.

14. In a Putty session, make sure you are connected to `srv1` as `oracle` user.

15. Using the `srvctl` utility, check the status of the database.

```
srvctl status database -d rac
```

16. Using the `srvctl` utility, stop the instance `rac1`

Note: the switch `-d` is an abbreviation to the switch `-db`. The switch `-i` is an appreciation to the switch `-instance`. Both the long form and the abbreviated form can be used.

```
srvctl stop instance -d rac -i rac1 -o immediate
```

17. Verify that the instance `rac1` is down.

```
# check out the status of rac1 in the output of the following command:
srvctl status database -d rac

# check out the processes in the OS level
ps -ef | grep pmon

# connect to the database and verify to which instance the session is connected
sqlplus system/oracle@rac
SELECT INSTANCE_NAME FROM V$INSTANCE;
```

18. Stop the instances `rac1` and `rac2` in a single `srvctl` utility command.

```
srvctl stop instance -d rac -i rac1,rac2 -o immediate
```

The command reports that `rac1` was already down but it still shuts down `rac2`.

At this stage the database is down and no user can connect to it.

19. Using SQL*Plus, connect to the local Oracle database instance as sysdba using the operating system authentication. Then, issue the `startup` command.

Note: in Oracle RAC database, this is **not** the recommended method to startup an instance. Always stick to using the `srvctl` utility command.

```
sqlplus / as sysdba
startup
```

20. Exit from SQL*Plus and check the status of the database using the `srvctl` utility.

```
srvctl status database -db rac
```

Observe that the `startup` command started the local instance (`rac1`) but not the entire database.

21. Startup the database using the `srvctl` utility.

Note: the `startoption` is optional and it defaults to open. With start command, it accepts the open, `mount` and `nomout` options.

```
srvctl start database -db rac -startoption open
```

22. Checkout the status of the database.

```
srvctl status database -d rac
```

Stopping the Complete ORACLE_HOME Component Stack

In the following steps, you will use the `srvctl` utility to stop all the components that are running from Oracle home. This a quick way to stop the resources running in Oracle home before upgrading Oracle Home or applying a patch on it.

23. Display the syntax usage help for `srvctl` status home command.

```
srvctl status home -help
```

24. Use the "`srvctl status home`" command to check the state of all the resources that are running from Oracle home. Save the status of the resources in a file in the oracle home directory for node `rac1`.

```
srvctl status home -oraclehome $ORACLE_HOME -statefile ~/rac1_state.dmp -node
srv1

# remove the file now, not needed.
rm ~/rac1_state.dmp
```

25. Display the syntax usage help for the command "`srvctl stop home`".

```
srvctl stop home -help
```

26. Stop all the resources running from ORACLE_HOME.

```
srvctl stop home -oraclehome $ORACLE_HOME -statefile ~/rac1_state.dmp -node srv1
```

27. Check the status of the database on each node.

```
srvctl status database -d rac
```

28. Start all the resource for ORACLE_HOME

```
srvctl start home -oraclehome $ORACLE_HOME -node srv1 -statefile
~/rac1_state.dmp
```

29. Check the status of the database on each node.

```
srvctl status database -d rac
```

30. Delete the state file.

```
rm ~/rac1_state.dmp
```

C. Managing the Initialization Parameters in Oracle RAC database

In this section of the practice, you will examine how to manage the initialization parameters in Oracle RAC database. The section concentrates on managing the parameters in spfile, not in pfile. This is because spfile is the format of the initialization parameters file that is highly recommended to use in production databases and it is the default.

31. In a Putty session, make sure you are connected to `srv1` as `oracle` user.

32. Login as `sys` to `rac1` instance and check out the location of the spfile.

```
sqlplus sys/oracle@rac1 as sysdba
show parameter spfile
```

33. Login as `system` to `rac2` instance and check out the location of the spfile.

```
conn sys/oracle@rac2 as sysdba
show parameter spfile
```

Each instance is connected to the same spfile.

34. Login back again to `rac1` instance and create pfile from spfile. View the contents of the generated pfile.

```
conn sys/oracle@rac1 as sysdba
host mkdir /home/oracle/scripts
CREATE PFILE='/home/oracle/scripts/racpfile.ora' FROM SPFILE;
host cat /home/oracle/scripts/racpfile.ora
```

Observe that the following parameters have different values for each instance.

- `INSTANCE_NUMBER`
- `THREAD`
- `UNDO_TABLESPACE`

Observe that all the instances have the same value for the parameter `DB_NAME`

In the following steps you will examine the different ways to change a parameter in the memory of an instance and in the spfile in a RAC database. The parameter ddl_lock_timeout is used as an example.

35. Create a script file and add the code in it as shown in the following code.

The script simply prints out the parameter memory value in the current instance and the parameter value in the SPFILE.

```
# create the following script file:
host vi myfile.sql
```

```
# add the following code in it:
set serveroutput on
set feedback off
exec dbms_output.put_line('Parameter value in memory:');

col name format a25
col value format a10
SELECT INST_ID, NAME,VALUE,TYPE,ISSYS_MODIFIABLE, ISINSTANCE_MODIFIABLE
FROM GV$PARAMETER WHERE name ='ddl_lock_timeout' ;

exec dbms_output.put_line('Parameter value in SPFILE:');
col sid format a10
SELECT SID,NAME,VALUE,TYPE, ISSPECIFIED
FROM V$SPPARAMETER WHERE NAME ='ddl_lock_timeout';
```

36. Execute the script.

```
@myfile.sql
```

```
Parameter value in memory:
   INST_ID NAME                      VALUE            TYPE ISSYS_MOD ISINS
---------- ------------------------- ---------- ---------- --------- -----
         1 ddl_lock_timeout          0                   3 IMMEDIATE TRUE
         2 ddl_lock_timeout          0                   3 IMMEDIATE TRUE

Parameter value in SPFILE:
SID        NAME                      VALUE      TYPE        ISSPEC
---------- ------------------------- ---------- ----------- ------
*          ddl_lock_timeout                     integer     FALSE
```

The output means that the current effective value in the instances for this parameter is zero. The second query shows that the current value of this parameter in SPFILE is null. This typically means that the parameter has not been defined in the SPFILE.

For this parameter, when it is not defined, its default value is zero. That is why the first query returns zero for each instance.

The first query shows that for this parameter the ISSYS_MODIFIABLE is IMMEDIATE. This means that the parameter can be modified by the ALTER SYSTEM command without having to restart the instance.

The ISINSTANCE_MODIFIABLE means that this parameter can be assigned different values to the RAC instances.

37. Modify the value of the parameter to 60 in the database level.

```
ALTER SYSTEM SET DDL_LOCK_TIMEOUT =60 SID='*';
```

38. Run the script.

```
@myfile.sql
```

Because the SCOPE option was not used in the ALTER SYSTEM command, the modification will take effect on both the memory as well as in the SPFILE.

39. Set the value of the parameter to 120 only in the instance rac1. Do not specify the SCOPE.

```
ALTER SYSTEM SET DDL_LOCK_TIMEOUT =120 SID='rac1';
```

40. Run the script.

```
@myfile.sql
```

For rac1, the current effective value of the parameter in memory changed to the new value (120). In the SPFILE, you will see two entries. One entry is a value of the parameter to only rac1. The other entry is the value of the parameter for the rest of the RAC instances.

41. Delete the parameter setting from the spfile that applies on all the instances.

The command ALTER SYSTEM RESET is used to delete a parameter setting from spfile.

```
ALTER SYSTEM RESET DDL_LOCK_TIMEOUT;
```

42. Run the script.

```
@myfile.sql
```

Observe that the parameter value has not been changed in the memory. The parameter for SID='*' has been removed from the SPFILE. However, the parameter setting for rac1 still exists in the spfile. To remove that setting from the spfile, you have to set the SID value in the ALTER SYSTEM RESET command, as shown in the next step.

43. Delete the parameter setting from spifle for rac1.

```
ALTER SYSTEM RESET DDL_LOCK_TIMEOUT SID='rac1';
```

44. Run the script.

```
@myfile.sql
```

The parameter value appears as null by the second query, which means the parameter setting has been deleted from the SPFILE.

45. Set the parameter value in the memory to its default setting.

```
ALTER SYSTEM SET DDL_LOCK_TIMEOUT=0 SCOPE=MEMORY;
```

46. Run the script.

```
@myfile.sql
```

The parameter has been set to its default value in the memory.

47. Delete the script file.

```
host rm myfile.sql
```

Summary

- Regarding the authentication methods when connecting to an Oracle RAC database:
  - The following authentication methods are applicable:
    - Operating system authenticated method
    - Password file authenticated method
    - Oracle database authenticated method
  - When you connect as sysdba using the password file authentication method, the OS authentication method supersedes the password file authentication method.

- Regarding the connection methods to an Oracle RAC database:
  - The following connection methods are applicable:
    - Use Easy Connect Naming method.
    - Configure tnsnames.ora file
  - The application should use SCAN name to connect to the database.

- `srvctl` utility is the preferred method to start and stop an Oracle RAC database or a RAC instance.

- You have the option to shut down the entire Oracle Home resources using the `srvctl` utility.

- Some initialization parameters can be modified in the memory and/or in the SPFILE. Some of them accept modification in the instance level and the rest can be modified only in the database level.