Practice 24: PDB Cloning and Relocation using DBCA

Practice Overview

In this practice, you will practice the PDB cloning and relocation methods in DBCA.

Specifically, you will use the `dbca` utility to perform the following:

- Create a PDB clone from the Seed container
- Create a PDB clone locally
- Create a PDB clone remotely
- Plug in an unplugged PDB
- Create a PDB from an RMAN backup

Practice Assumptions

- You have the `ORADB` (in `srv1`) and `ORADB2` (in `srv2`) databases up and running.

Preparing the Environment for the Practice

In the following steps, you prepare `oradb` and `oradb2` for the practice.

1. In Oracle VirtualBox, take a snapshot for `srv1` and `srv2`.

Caution: Do not proceed with the practice before taking a snapshot for the `srv1` and `srv2` first.

2. Open two Putty sessions two `srv1` and `srv2` as `oracle`

3. In `srv1`, invoke SQL*Plus, login to `oradb` as `sysdba`.

```
sqlplus / as sysdba
```

4. Verify that the archivelog mode is enabled in `oradb` database.

   If the archivelog mode is not enabled in a database, the dbca automatically shuts down the source PDB when it is used to clone the PDB.

```
archive log list;
```

5. Verify that local undo is enabled.

   If the local undo is not enabled in a database, the dbca automatically shuts down the source PDB when it is used to clone the PDB.

```
col PROPERTY_NAME for a30
col PROPERTY_VALUE for a30

SELECT PROPERTY_NAME, PROPERTY_VALUE
FROM   DATABASE_PROPERTIES
WHERE  PROPERTY_NAME = 'LOCAL_UNDO_ENABLED';
```

Creating a PDB from Seed

In the following steps, you will create a new PDB (named as PDB2 in oradb) from the seed container.

6. In srv1 session, display the available PDBs in oradb

```
show pdbs
```

7. Check out if the OMF is enabled or disabled.

OMF is disabled in oradb

```
show parameter DB_CREATE_FILE_DEST
```

8. Open a new Putty session to srv1 as oracle. Issue the following command in it to monitor the changes in the alertlog file of oradb.

In the rest of the practice, this session is referred to as the **srv1 monitoring** session. This session helps us to understand what the dbca executes in the background.

```
tail -f $ORACLE_BASE/diag/rdbms/oradb/oradb/trace/alert_oradb.log
```

9. In the other srv1 session, issue the following dbca command to create a PDB named as PDB2 from the seed container.

While the dbca is executing the command, you can look at the monitoring session to check out the commands executed by the utility.

The dbca creates the PDB from the seed container because the parameter createpdbfrom is set to DEFAUTL.

```
dbca -silent -createpluggabledatabase -createpdbfrom DEFAULT -sourcedb oradb -
pdbName pdb2 -pdbAdminUserName pdb2admin -pdbAdminPassword ABcd##1234
```

10. Display the available PDBs in oradb and make sure PDB2 is open in read/write mode.

```
sqlplus / as sysdba
show pdbs
```

11. Display the datafiles of PDB2.

Obtain the value of PDB2 CON_ID from the preceding step.

Observe that the PDB2 datafiles are saved in the following directory:
$ORACLE_BASE/oradata/$ORACLE_SID/<pdb>

```
SELECT FILE_NAME FROM CDB_DATA_FILES WHERE CON_ID=&PDB2_CON_ID;
```

Let's test if setting the `dbca` parameter `pdbDatafileDestination` changes the clone PDB datafiles destination directory.

12. Drop the cloned PDB.

```
ALTER PLUGGABLE DATABASE PDB2 CLOSE;
DROP PLUGGABLE DATABASE PDB2 INCLUDING DATAFILES;
```

13. Create a testing directory.

```
host mkdir -p /home/oracle/oradata
```

14. Issue the following `dbca` command to create a PDB named as `PDB2` from the seed container and set the `PDB2` datafiles in the created directory using the `pdbDatafileDestination` parameter.

In the monitoring session observe the following statement was run by the `dbca`:

```
CREATE PLUGGABLE DATABASE pdb2 ADMIN USER PDBADMIN IDENTIFIED BY *
ROLES=(CONNECT)  file_name_convert=
```

The parameter `file_name_convert` was used to set the location of the each PDB datafile.

```
dbca -silent -createpluggabledatabase -createpdbfrom DEFAULT -sourcedb oradb -
pdbName pdb2 -pdbAdminUserName pdb2admin -pdbAdminPassword ABcd##1234 -
pdbDatafileDestination '/home/oracle/oradata'
```


15. Display the `PDB2` datafiles.

`PDB2` datafiles where created under the directory pointed by `pdbDatafileDestination` value

```
sqlplus / as sysdba
show pdbs
SELECT FILE_NAME FROM CDB_DATA_FILES WHERE CON_ID=&PDB2_CON_ID;
```

Let's study the impact of enabling OMF on the parameter `pdbDatafileDestination`

16. Drop the cloned PDB.

```
ALTER PLUGGABLE DATABASE PDB2 CLOSE;
DROP PLUGGABLE DATABASE PDB2 INCLUDING DATAFILES;
```

17. Create a testing OMF directory.

```
host mkdir -p /home/oracle/omf
```

18. Set the OMF to the created directory.

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST='/home/oracle/omf' SCOPE=MEMORY;

show parameter DB_CREATE_FILE_DEST
```

19. Issue the following `dbca` command to create `PDB2` from the seed container and set the `pdbDatafileDestination` parameter.

In the monitoring session observe the following statement was run by the `dbca`:

```
CREATE PLUGGABLE DATABASE pdb2 ADMIN USER PDBADMIN IDENTIFIED BY *
ROLES=(CONNECT)  file_name_convert=NONE
```

The `file_name_convert` is set to `NONE`, which means the parameter `pdbDatafileDestination` was ignored.

```
dbca -silent -createpluggabledatabase -createpdbfrom DEFAULT -sourcedb oradb -
pdbName pdb2 -pdbAdminUserName pdb2admin -pdbAdminPassword ABcd##1234 -
pdbDatafileDestination '/home/oracle/oradata'
```

20. Display the datafiles of `PDB2`.

The parameter `pdbDatafileDestination` is ignored when OMF is enabled.

As expected, the datafiles were created under the OMF. The subdirectories were automatically created by the database.

```
sqlplus / as sysdba
show pdbs

SELECT FILE_NAME FROM CDB_DATA_FILES WHERE CON_ID=&PDB2_CON_ID;
```


Clean up

21. Drop the cloned PDB.

```
ALTER PLUGGABLE DATABASE PDB2 CLOSE;
DROP PLUGGABLE DATABASE PDB2 INCLUDING DATAFILES;
```

22. Disable the OMF.

Use the `RESET` option for disabling the OMF as indicated by the code below. Do not set the parameter to null or `''`. From my testing, I faced issues when setting the parameter to null.

```
ALTER SYSTEM RESET DB_CREATE_FILE_DEST SCOPE=MEMORY;
```

Creating a Clone PDB Locally

In the following steps, you will clone PDB1 in oradb and name the cloned PDB as PDB2.

23. In the other srv1 session, issue the following dbca command to clone PDB1 into a new PDB named as PDB2.

The dbca understands that we want to clone a local PDB because the parameter createpdbfrom is set to PDB

```
dbca -silent -createpluggabledatabase -sourcedb oradb -createpdbfrom PDB -pdbName
pdb2 -sourcepdb pdb1
```

24. Display the datafiles of PDB2.

PDB2 datafiles are saved in the following directory:

$ORACLE_BASE/oradata/$ORACLE_SID/<pdb>

```
sqlplus / as sysdba
show pdbs
SELECT FILE_NAME FROM CDB_DATA_FILES WHERE CON_ID=&PDB2_CON_ID;
```

The impact of pdbDatafileDestination parameter and enabling OMF on this scenario is the same as their impact on the previous scenario. If you want to, you can test it yourself.

Let's us test the impact of the fileNameConvert parameter.

25. Drop the cloned PDB.

```
ALTER PLUGGABLE DATABASE PDB2 CLOSE;
DROP PLUGGABLE DATABASE PDB2 INCLUDING DATAFILES;
```

26. List the location of PDB1 files.

PDB1 datafiles are saved in the following directory:

$ORACLE_BASE/oradata/ORADB/pdb1/

```
SELECT FILE_NAME FROM CDB_DATA_FILES WHERE CON_ID=3;
```

27. Issue the following `dbca` command to clone `PDB1` into a new PDB named as `PDB2`. Use `fileNameConvert` parameter to set `PDB2` datafiles location to `/home/oracle/oradata`

Note: If you try creating the pluggable database `PDB2` from SQL using the following command, it works fine:

```
CREATE PLUGGABLE DATABASE pdb3 FROM pdb1
FILE_NAME_CONVERT=('/u01/app/oracle/oradata/ORADB/pdb1','/home/oracle/oradata');
```

```
dbca -silent -createpluggabledatabase -sourcedb oradb -createpdbfrom PDB -pdbName pdb2 -sourcepdb pdb1 -fileNameConvert '/u01/app/oracle/oradata/ORADB/pdb1','/home/oracle/oradata'
```

28. Display the datafiles of `PDB2`.

The value of the file `fileNameConvert` parameter did not affect the `PDB2` datafiles location. This is not the expected behavior.

```
sqlplus / as sysdba
show pdbs
SELECT FILE_NAME FROM CDB_DATA_FILES WHERE CON_ID=&PDB2_CON_ID;
```

Clean up

29. Drop the cloned PDB.

```
ALTER PLUGGABLE DATABASE PDB2 CLOSE;
DROP PLUGGABLE DATABASE PDB2 INCLUDING DATAFILES;
```

Creating a Clone PDB Remotely

In the following steps, you will clone PDB1 in oradb (which is hosted in srv1) into oradb2 (which is hosted in srv2). You will name the new PDB as PDB2.

30. In srv2 session, verify the OMF is disabled.

```
sqlplus / as sysdba

show parameter DB_CREATE_FILE_DEST
```

31. In srv1 session, create a common user and grant it the privileges needed to clone a PDB.

```
CREATE USER c##ruser IDENTIFIED BY abc##1234;
GRANT CREATE SESSION, RESOURCE, CREATE PLUGGABLE DATABASE TO c##ruser
CONTAINER=ALL;
GRANT SYSOPER TO c##ruser CONTAINER=ALL;
```

32. In srv2 session, invoke dbca to clone PDB1 into oradb2 and name the new PDB as PDB2.

Observe the common user is not enough for the dbca to execute the command. It requires a sysdba user as well.

```
dbca -silent -createPluggableDatabase -createFromRemotePDB -sourceDB oradb2 -
remotePDBName pdb1 -remoteDBConnString srv1:1521/oradb.localdomain -
remoteDBSYSDBAUserName SYS -remoteDBSYSDBAUserPassword oracle -dbLinkUsername
c##ruser -dbLinkUserPassword abc##1234 -sysDBAUserName SYS -sysDBAPassword oracle
-pdbName pdb2
```

33. Verify PDB2 is created in oradb2 and is open in read/write mode.

```
sqlplus / as sysdba
show pdbs
```

34. Check out the PDB2 datafiles location.

```
SELECT FILE_NAME FROM CDB_DATA_FILES WHERE CON_ID=&PDB2_CON_ID;
```

Clean up

35. In srv2, drop the cloned PDB.

```
ALTER PLUGGABLE DATABASE PDB2 CLOSE;
DROP PLUGGABLE DATABASE PDB2 INCLUDING DATAFILES;
```

Plugging in an Unplugged PDB

In this section of the practice, you will plug in an unplugged PDB.

Because we do not want to change `PDB1` in `srv1`, you will first locally clone `PDB1` in `oradb` into a new PDB named as `PDB2`. You will then unplug `PDB2` and plug it into `oradb2` as `PDB3`.

36. In `srv1` session, locally clone `PDB1` as a new PDB named `PDB2`.

```
dbca -silent -createpluggabledatabase -sourcedb oradb -createpdbfrom PDB -pdbName
pdb2 -sourcepdb pdb1
```

37. Use `dbca` to unplug `PDB2`. Save the unplugged PDB files into an archive file in the sharing folder.

```
dbca -silent -unplugDatabase -sourceDB oradb -pdbName pdb2 -archiveType TAR -
pdbArchiveFile '/media/sf_staging/pdb2.arc'
```

38. Check out the size of the generated tar file.

```
ls -alh /media/sf_staging/pdb2.arc
```

39. In `srv2`, start the `dbca` in silent mode to plug in the unplugged PDB. Name the new PDB as `PDB3`.

```
dbca -silent -createPluggableDatabase -sourceDB oradb2 -pdbName pdb3 -
createPDBFrom FILEARCHIVE -pdbArchiveFile '/media/sf_staging/pdb2.arc' -workArea
'/media/sf_staging/'
```

40. Checkout the location of the `PDB3` in `oradb2`.

As expected, the files are saved into the following directory:

`$ORACLE_BASE/oradata/ORADB2/pdb3`

```
sqlplus / as sysdba
show pdbs
SELECT FILE_NAME FROM CDB_DATA_FILES WHERE CON_ID=&PDB3_CON_ID;
```

Creating a PDB from an RMAN backup

In this section of the practice, you will use dbca to create a PDB from an RMAN backupset file.

41. In `srv1` session, invoke RMAN and take a backup of `PDB1` as backupset. Save the backupset file into the staging folder.

```
$ORACLE_HOME/bin/rman target /

RMAN> backup as backupset pluggable database pdb1 format
'/media/sf_staging/pdb1.back';
```

42. Display the produced backup piece file.

```
LIST BACKUPSET OF DATABASE;
```

43. Produce the metadata xml file of `PDB1`.

```
sqlplus / as sysdba
ALTER SESSION SET CONTAINER=PDB1;
exec DBMS_PDB.DESCRIBE( PDB_DESCR_FILE => '/media/sf_staging/pdb1.xml')
```

44. In `srv2` session, invoke dbca in silent mode to create a PDB named as `PDB4`.

```
dbca -silent -createPluggableDatabase  -createPDBFrom RMANBACKUP -pdbBackUpfile
'/media/sf_staging/pdb1.back' -pdbMetadataFile '/media/sf_staging/pdb1.xml' -
pdbName pdb4 -sourceDB oradb2
```

45. Verify that `PDB4` is created in `srv2`

```
sqlplus / as sysdba
show pdbs
```

Clean up

46. Shutdown `srv1` and `srv2`

47. In Oracle VirtualBox, restore `srv1` and `srv2` from their snapshots that were taken in the first section of the practice

48. Delete the snapshots created for `srv1` and `srv2`

49. Delete the backupset file, xml file, and archive file created in the staging folder.

Summary

- `dbca` can be used to create a PDB clone in the following scenarios:
  - Creating a PDB clone locally
  - Creating a PDB clone remotely
  - Plugging in an unplugged PDB
  - Creating a PDB from an RMAN backup
- The datafiles location of the PDB cloned by dbca is set based on the following algorithm:
  - If OMF is enabled, the datafiles will be saved on it (regardless of the `pdbDatafileDestination` and `fileNameConvert` values)
  - If OMF is disabled and the parameter `pdbDatafileDestination` is set, datafiles will be saved in the directory pointed by the parameter
  - If OMF is disabled and the parameter `fileNameConvert` is set, datafiles should be saved in the directory pointed by the parameter but that is not practically happening (looks like a bug)
  - If OMF is disable and neither of `pdbDatafileDestination` or `fileNameConvert` is set, then the datafiles will be saved in the following path:

    `$ORACLE_BASE/oradata/$ORACLE_SID/<pdb>`