Oracle 19c Multi Tenant LAbs

Knowlogy®

Multitenant Labs

Lab 1: Installing HR

Overview

In this practice, you will manually install the HR sample schema.

Assumptions

You have a connection to the compute node through PuTTY or SSH and are logged in as the `oracle` user.

Tasks

1. In your terminal window, navigate to the `$ORACLE_HOME/demo/schema/human_resources` directory.

```
[oracle@MYDBCS ~]$ cd $ORACLE_HOME/demo/schema/human_resources
[oracle@MYDBCS human_resources]$
```

2. Use the `ls` command to view the contents of the `human_resources` directory. In a later step, you will execute the `hr_main.sql` to create the HR user, objects and load data into the HR tables.

```
[oracle@MYDBCS human_resources]$ ls
hr_analz.sql  hr_comnt.sql  hr_drop_new.sql  hr_idx.sql
hr_main.sql
hr_code.sql   hr_cre.sql    hr_drop.sql      hr_main_new.sql
hr_popul.sql
[oracle@MYDBCS human_resources]$
```

3. Start SQL*Plus and connect to the root container as the SYS user with the SYSDBA privilege.

```
[oracle@MYDBCS human_resources]$ sqlplus / as sysdba
…
SQL>
```

4. Switch to PDB1.

```
SQL> ALTER SESSION SET CONTAINER = PDB1;
Session altered.

SQL>
```

5. Execute the **`hr_main.sql`** script and respond to the prompts as follows.
   a. Enter the password for the HR user as specified in the *Course Practice Environment: Security Credentials*.
   b. Enter **`USERS`** as the default tablespace for the HR user.
   c. Enter **`TEMP`** as the temporary tablespace for the HR user.
   d. Enter **`$ORACLE_HOME/demo/schema/log/`** for the log directory.

```
SQL> @hr_main

specify password for HR as parameter 1:
Enter value for 1: password

specify default tablespeace for HR as parameter 2:
Enter value for 2: USERS

specify temporary tablespace for HR as parameter 3:
Enter value for 3: TEMP

specify log path as parameter 4:
Enter value for 4: $ORACLE_HOME/demo/schema/log/

PL/SQL procedure successfully completed.

User created.

User altered.

Grant succeeded.
…

Comment created.

Commit complete.

PL/SQL procedure successfully completed.

SQL>
```

e. Exit from SQL*Plus.

```
SQL> exit
…
[oracle@MYDBCS human_resources]$
```

6. Query the `USER_TABLES` view as the `HR` user to verify that the user and tables were created.

   a. Connect as the `HR` user. Be sure to provide the correct service name for your PDB as you did in Practice 5-4, step 3c.

```
[oracle@MYDBCS human_resources]$ sqlplus
hr/password@localhost:1521/PDB1.588436052.oraclecloud.internal
…
SQL>
```

b. Query `USER_TABLES`.

```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME
--------------------------------------------------------------------------------
REGIONS
COUNTRIES
LOCATIONS
DEPARTMENTS
JOBS
EMPLOYEES
JOB_HISTORY

7 rows selected.

SQL>
```

7. Exit from SQL*Plus and close the connection to the computenode.

```
SQL> exit
Disconnected from Oracle Database 19c Enterprise Edition Release
18.0.0.0.0 - Production
Version 18.1.0.0.0
[oracle@MYDBCS human_resources]$ exit
```

Lab 2: Reviewing the Multitenant configuration

Overview

In this practice, you will learn how to do the following things:

- Set the Oracle environment variables
- Connect to the root container by using SQL*Plus
- Query the data dictionary to view information about the containers, data files, users, instance, and services in a CDB
- List the services created automatically for each container

Some things to remember when you want to query the data dictionary for multiple PDBs or the whole CDB:

- Log in to the root container as a common user. A CDB common user is a database account created in the root container and is inherited by all PDBs in the CDB.

- Query container data objects, such as views whose names begin with V$ and CDB_. For more information, refer to the following sections in *Oracle Database Administrator's Guide*:

- About Viewing Information When the Current Container is the CDB Root
- Viewing Information About the Containers in a CDB

In some of the steps below, you will format columns by using the `COLUMN` command. For example, applying the format `A55` specifies an alphabetic format of 55 characters wide. Format `999` is an example of a numeric format. See *SQL*Plus User's Guide and Reference* for additional information.

Commands in the practices are in uppercase and variables are in lower case. Any commands that you need to enter are bolded, for example:

```
SQL> SELECT regions FROM hr.departments;
```

Assumptions

You are connected to the compute node as the `oracle` user. See Practice 5-2 for detail.

Tasks

1. Set the Oracle environment variables. You need to set these each time you open a new terminal window.

   a. In the terminal window, list the search path that holds the `oraenv` script.

   b. Source the `oraenv` script. `oraenv` sets the required environment variables needed for you to connect to your database instance. The `oraenv` script sets the `ORACLE_SID` and `ORACLE_HOME` environment variables and includes the `$ORACLE_HOME/bin` directory in the `PATH` environment variable setting. Environment variables that this script sets will persist in the terminal window until you close it. For the ORACLE_SID value, enter ORCL.

```
[oracle@MYDBCS ~]$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base has been set to /u01/app/oracle
[oracle@MYDBCS ~]$
```

c. View the environment variables set by the oraenv command.

```
[oracle@MYDBCS ~]$ set | grep ORACLE
OLD_ORACLE_BASE=/u01/app/oracle
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/18.0.0/dbhome_1
ORACLE_HOSTNAME=MYDBCS.compute-588436052.oraclecloud.internal
ORACLE_SID=ORCL
ORACLE_UNQNAME=ORCL
[oracle@MYDBCS ~]$
```

Note: Remember that from this point on, each time you open a terminal window you will need to source the oraenv script to set the environment variables for your CDB.

2. Connect to the root container by using SQL*Plus.

   a. Start SQL*Plus and log in to the root container of your CDB as the SYS user with the SYSDBA privilege. You can connect to a database without a password when you have a local connection (on the same machine) and the current operating system user is a member of the privileged OSDBA group.

   b. Sqlplus / as sysdba

   c. Sql> select * from v$containers;

   b. Verify that you are logged in to the root container as the SYS user by using the SHOW USER command.

```
SQL> SHOW user
USER is "SYS"
SQL>
```

3. View information about the containers in your CDB.

a. Verify that you have a container database by querying the `V$DATABASE` view. The `NAME` column should contain `ORCL`, the CDB column should contain `YES`, and the `ID` should be 0 (zero). A value of zero is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.

```
SQL> SELECT name, cdb, con_id FROM v$database;


NAME        CDB      CON_ID
----------- ---- ------------
ORCL        YES           0


SQL>
```

b. Show the current container name. Because you're currently connected to the root container, the name should be `CDB$ROOT`.

```
SQL> SHOW con_name

CON_NAME
------------------------------
CDB$ROOT


SQL>
```

c. Show the current container ID. Because you're currently connected to the root container, the ID should be `1`.

```
SQL> SHOW con_id

CON_ID
------------------------------
1


SQL>
```

d. Determine the version of Oracle Database by querying the `V$VERSION` view. This view displays version numbers of core library components in Oracle Database.

```
SQL> SELECT banner FROM v$version;



--------------------------------------------------------------------------------
```

e. List all the containers in your CDB by querying the `V$CONTAINERS` view. The results should show three containers—the root container (`CDB$ROOT`), the seed PDB (`PDB$SEED`), and `PDB1`.

```
SQL> COLUMN name FORMAT A8
SQL> SELECT name, con_id FROM v$containers ORDER BY con_id;

NAME          CON_ID
--------

CDB$ROOT           1
PDB$SEED           2
PDB1               3

SQL>
```

f. List the PDBs in the CDB by using the `SHOW` command. The result should show two PDBs—the seed PDB (`PDB$SEED`) and `PDB1`. You can also list PDBs by querying the `V$PDBS` view. The `SHOW` command includes information about the open mode of each PDB and whether the PDB is restricted. The open mode for a PDB determines what type of activities a PDB will allow at that time. `PDB$SEED` is in `READ ONLY` mode and `PDB1` is in `READ WRITE` mode. The `RESTRICTED` column indicates whether only users possessing the `RESTRICTED SESSION` privilege can connect to the PDB.

```
SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ WRITE NO
SQL>
```

g. View the status of all PDBs in the CDB by querying the `CDB_PDBS` view. The status of a PDB describes the state of the PDB. For example, if the PDB is new, but never opened, the status is `NEW`. If it is available and ready for use, the status is `NORMAL`.

```
SQL> COLUMN pdb_name FORMAT A8
SQL> SELECT pdb_name, status FROM cdb_pdbs ORDER BY 1;

PDB_NAME STATUS
-------- ----------
PDB1     NORMAL
PDB$SEED NORMAL

SQL>
```

4. View information about the data files in your CDB.

a. List all the data files in the CDB (for the root container and all PDBs) by querying the `CDB_DATA_FILES` view. The order of your results may vary.

```
SQL> COLUMN file_name FORMAT A50
SQL> COLUMN tablespace_name FORMAT A10
SQL> SELECT file_name, tablespace_name FROM cdb_data_files;

FILE_NAME                                                    TABLESPACE
------------------------------------------------------------ ------------
/u02/app/oracle/oradata/ORCL/users01.dbf                     USERS
/u02/app/oracle/oradata/ORCL/undotbs01.dbf                   UNDOTBS1
/u02/app/oracle/oradata/ORCL/system01.dbf                    SYSTEM
/u02/app/oracle/oradata/ORCL/sysaux01.dbf                    SYSAUX
/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf               SYSTEM
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf               SYSAUX
/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf              UNDOTBS1
/u02/app/oracle/oradata/ORCL/PDB1/PDB1_users01.dbf           USERS

8 rows selected.

```

```
SQL>
```

b. List all the tablespaces in the CDB (for both the root container and all the PDBs) by querying the `V$DATAFILE` and `V$TABLESPACE` views.

```
SQL> COL name FORMAT A12
SQL> SELECT d.file#, ts.name, ts.ts#, ts.con_id
  2  FROM v$datafile d, v$tablespace ts
  3  WHERE d.ts#=ts.ts# AND d.con_id=ts.con_id
  4  ORDER BY 4;

     FILE# NAME                TS#     CON_ID
---------- ------------ ---------- ----------
         1 SYSTEM                0          1
         3 SYSAUX                1          1
         4 UNDOTBS1              2          1
         7 USERS                 4          1
         6 SYSAUX                1          2
        13 USERS                 5          2
         8 UNDOTBS1              2          2
         5 SYSTEM                0          2
         9 SYSTEM                0          3
        10 SYSAUX                1          3
        11 UNDOTBS1              2          3
        12 USERS                 5          3
12 rows selected.

SQL>
```

c. List all temp files in the CDB (for the root container and all PDBs) by querying the `CDB_TEMP_FILES` view.

```
SQL> SELECT file_name, tablespace_name FROM cdb_temp_files;

FILE_NAME                                                   TABLESPACE
----------------------------------------------------------- ----------
/u04/app/oracle/oradata/temp/temp01.dbf                     TEMP
/u02/app/oracle/oradata/ORCL/PDB1/pdbseed_temp0120 TEMP
18-02-19_18-48-12-642-PM.dbf

SQL>
```

d. List all the redo log files in the CDB (for the root container and all PDBs) by querying the `V$LOGFILE` view.

```
SQL> COLUMN member FORMAT A42
SQL> SELECT group#, member, con_id FROM v$logfile;

    GROUP# MEMBER                                          CON_ID
---------- ------------------------------------------ ----------
         3 /u04/app/oracle/redo/redo03.log                  0
         2 /u04/app/oracle/redo/redo02.log                  0
         1 /u04/app/oracle/redo/redo01.log                  0

SQL>
```

e. List the control files in the CDB by querying the `V$CONTROLFILE` view. There should be two—`control01.ctl` and `control02.ctl`.

```
SQL> COLUMN name FORMAT A55
SQL> SELECT name, con_id FROM v$controlfile;

NAME                                                    CON_ID
------------------------------------------------------- ----------
/u02/app/oracle/oradata/ORCL/control01.ctl                       0
/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl            0

SQL>
```

5. View information about the pre-created users in your CDB.

a. List only the common users in the CDB by querying the `CDB_USERS` view.

```
SQL> SELECT DISTINCT username FROM cdb_users
  2  WHERE common ='YES' ORDER BY 1;

USERNAME
```

```
ANONYMOUS
APPQOSSYS
AUDSYS
C##DBAAS_BACKUP
…
SYSTEM
WMSYS
XDB
XS$NULL

38 rows selected.

SQL>
```

b. List all the users in every PDB in the CDB by querying the `CDB_USERS` view. In the results, notice that the `SYS`, `SYSTEM`, and `PDBADMIN` user accounts are listed for `PDB1`. The root container's id is `1` and PDB1's id is `3`.

```
SQL> COLUMN username FORMAT A25
SQL> SELECT con_id, username FROM cdb_users
  2  ORDER BY username, con_id;
```

```
SQL>
```

6. View information about the database instance and the services.

   a. View the database instance name, its status, and which container database it is associated with by querying the `V$INSTANCE` view. The instance's status is `OPEN`, which means users can access the CDB and PDB.

```
SQL> SELECT instance_name, status, con_id FROM v$instance;

INSTANCE_NAME    STATUS            CON_ID

---------------- ------------ ----------
ORCL             OPEN                  0


SQL>
```

   b. List the services for all the containers in the CDB by querying the `V$SERVICES` view. The query returns five services. The `PDB$SEED` service is not listed because no one should connect to it and no operation should be performed with it. It is reserved as a template to create other PDBs.

```
SQL> SELECT con_id, name FROM v$services ORDER BY 1;


    CON_ID NAME
---------- ----------------------------------------------------------------
         1 SYS$BACKGROUND
         1 ORCL.588436052.oraclecloud.internal
         1 ORCL.588436052.oraclecloud.internalXDB
         1 SYS$USERS
         3 pdb1


SQL>
```

7. Exit SQL*Plus.

```
SQL > exit
Disconnected from Oracle Database 19c EE High Perf Release
18.0.0.0.0 - Production
Version 18.1.0.0.0
[oracle@MYDBCS ~]$
```

Lab 3: Exploring the PDB configuration

Tasks

1. Connect to PDB1 indirectly through the root container.
   a. Start SQL*Plus and connect to the root container as the SYS user with the SYSDBA privilege. Oracle allows any DBA group user at the operating system level to log into SQL*Plus without any authentication.

```
[oracle@MYDBCS ~]$ sqlplus / as sysdba

...


SQL>
```

   b. Verify that PDB1 is open. After DBCA creates a PDB, it opens it automatically. The results below indicate that the open mode is READ WRITE, which means PDB1 is open. PDB users with the SYSDBA, SYSOPER, SYSBACKUP, SYSDG, SYSKM, or SYSRAC privilege can connect to a closed PDB; however, all other PDB users can connect only when the PDB is open.

```
SQL> COLUMN con_id FORMAT 999
SQL> COLUMN name FORMAT A10
SQL> SELECT con_id, name, open_mode FROM v$pdbs;

CON_ID NAME       OPEN_MODE
```

```
-------- -------------- --------------
       2 PDB$SEED       READ ONLY
       3 PDB1           READ WRITE

SQL>
```

c. If PDB1 is closed for some reason and its open mode was MOUNTED in the previous step, open it by using the ALTER PLUGGABLE DATABASE command.

```
SQL> ALTER PLUGGABLE DATABASE PDB1 OPEN;

Pluggable database altered.

SQL>
```

d. Switch to PDB1. When logged in to a CDB as an appropriately privileged user, you can use the ALTER SESSION command to switch between containers within the CDB. From this point on, your queries against the data dictionary will retrieve information for PDB1 only.

```
SQL> ALTER SESSION SET CONTAINER = PDB1;

Session altered.

SQL>
```

e. Verify that the container name is PDB1.

```
SQL> SHOW con_name

CON_NAME
------------------------------
PDB1
SQL>
```

2. Query the data dictionary to list the data files and temp files for PDB1.

   a. List the data files for PDB1 and the tablespaces to which they belong by querying the DBA_DATA_FILES view.

```
SQL> col file_name format a60
SQL> col tablespace_name format a10
SQL> SELECT file_name, tablespace_name FROM dba_data_files;

FILE_NAME                                                    TABLESPACE
------------------------------------------------------------ ----------
/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf               SYSTEM
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf               SYSAUX
/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf              UNDOTBS1
```

```
/u02/app/oracle/oradata/ORCL/PDB1/PDB1_users01.dbf     USERS

SQL>
```

b. List the temp files for `PDB1` and the tablespaces to which they belong by querying the `DBA_TEMP_FILES` view.

```
SQL> SELECT file_name, tablespace_name FROM dba_temp_files;

FILE_NAME                                                   TABLESPACE
----------------------------------------------------        ----------
/u02/app/oracle/oradata/ORCL/PDB1/pdbseed_temp012018-02-19_1 TEMP
8-48-12-642-PM.dbf
SQL>
```

c. List the local users for `PDB1` by querying the `DBA_USERS` view.

```
SQL> SELECT DISTINCT username FROM dba_users WHERE common='NO';

USERNAME
--------------------------------------------------------------------------------
PDBADMIN
APEX_LISTENER
APEX_PUBLIC_USER
APEX_REST_PUBLIC_USER
FLOWS_FILES
APEX_050100
APEX_INSTANCE_ADMIN_USER
SCOTT

8 rows selected.

SQL>
```

3. Make a direct connection to PDB1 by using the Easy Connect syntax. The Easy Connect syntax enables you to connect to the PDB without 1) requiring a connection to the root container and 2) having to set up a net service name for the PDB.

a. Disconnect from the PDB.

```
SQL > DISCONNECT
Disconnected from Oracle Database 19c EE High Perf Release
18.0.0.0.0 - Production
Version 18.1.0.0.0
SQL>
```

b. Verify that you aren't connected as any user. The `SHOW user` command returns " " indicating that you are not connected.

```
SQL> SHOW user
```

```
USER is ""
SQL>
```

c. Connect to `PDB1` directly as the `SYSTEM` user by using the Easy Connect syntax. See *Course Practice Environment: Security Credentials* for the `SYSTEM` user password. In Practice 5-3, step 6b, you queried `V$SERVICES`. Append the value in the query results following `ORCL` to `pdb1` to create the service name as shown in this example.

```
SQL> CONNECT
system/password@localhost:1521/pdb

Connected.
SQL>
```

d. Verify that you are now connected as the `SYSTEM` user by using the `SHOW USER` command again.

```
SQL> SHOW user
SQL> USER is "SYSTEM"
SQL>
```

4. Exit SQL*Plus.

```
SQL> EXIT
…
[oracle@MYDBCS ~]$
```

Lab 4: Viewing Initialization Parameters by Using SQL*Plus

1. Start SQL*Plus and connect to the root container as the SYS user with the SYSDBA privilege.

```
$ sqlplus / as sysdba
..
SQL>
```

2. View the values of the DB_NAME and DB_DOMAIN parameters. Together, these values create the global database name.

   a. View the value of the DB_NAME parameter. This parameter specifies the current database identifier of up to eight characters. If you have multiple databases, the value of this parameter should match the Oracle instance identifier of each one to avoid confusion with other databases running on the system.

```
SQL> SHOW PARAMETER db_name
NAME                                  TYPE        VALUE
------------------------------------- ----------- ----------
db_name                               string      ORCL
SQL>
```

   b. View the value of the DB_DOMAIN parameter. In a distributed database system, DB_DOMAIN specifies the logical location of the database within the network structure. You should set this parameter if this database is or ever will be part of a distributed system. There is no default value.

```
SQL> SHOW PARAMETER db_domain
NAME                           TYPE     VALUE
```

```
db_domain                    string  588436052.oraclecloud.internal
SQL>
```

3. View the `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` parameters. These parameters set the location of the fast recovery area and its size.

   The `DB_RECOVERY_FILE_DEST` parameter specifies the default location for the fast recovery area. The fast recovery area contains multiplexed copies of current control files and online redo logs, as well as archived redo logs, flashback logs, and Recovery Manager (RMAN) backups. If you specify a value for `DB_RECOVERY_FILE_DEST`, you must also specify a value for the `DB_RECOVERY_FILE_DEST_SIZE` initialization parameter.

   The `DB_RECOVERY_FILE_DEST_SIZE` parameter specifies (in bytes) the hard limit on the total space to be used by target database recovery files created in the fast recovery area.

```
SQL> SHOW PARAMETER db_recovery_file_dest
NAME                          TYPE   VALUE
----------------------        ------ ----------------------------------
db_recovery_file_dest         string /u03/app/oracle/fast_recovery_area
db_recovery_file_dest_size    big integer 4G
SQL>
```

4. View the `SGA_TARGET` and `SGA_MAX_SIZE` parameters.

   `SGA_TARGET` specifies the total amount of SGA memory available to a database instance and `SGA_MAX_SIZE` sets a maximum size for the SGA.

   If you set the `SGA_TARGET` parameter, you enable the Automatic Shared Memory Management (ASMM) feature. The Oracle Database server will automatically distribute memory among the various SGA memory pools (buffer cache, shared pool, large pool, java pool, and streams pool), ensuring the most effective memory utilization. Note, the log buffer pool, other buffer caches (such as `KEEP` and `RECYCLE`), other block sizes, fixed SGA, and other internal allocations must be manually sized and are not affected by ASMM. The memory allocated to these pools is deducted from the total available memory for `SGA_TARGET` when ASMM is enabled.

   The manageability monitor process (MMON) computes the values of the automatically tuned memory pools to support ASMM.

   In addition to `SGA_TARGET` and `SGA_MAX_SIZE`, you can set minimum nonzero values for each memory pool if an application component needs a minimum amount of memory to function properly. ASMM will treat those values as minimum levels.

   The range of values for `SGA_TARGET` can be from `64 MB` to an operating system-dependent value. You can't modify this value in a PDB.

```
SQL> SHOW PARAMETER sga
NAME                                 TYPE        VALUE

allow_group_access_to_sga            boolean     FALSE
lock_sga                             boolean     FALSE
pre_page_sga                         boolean     TRUE
```

```
sga_max_size                         big integer 2640M
sga_min_size                         big integer 0
sga_target                           big integer 2640M
unified_audit_sga_queue_size         integer     1048576
SQL>
```

5. View the `UNDO_TABLESPACE` parameter. This parameter specifies the undo tablespace to be used when an instance starts. Oracle Database creates and manages information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as undo and are stored in the undo tablespace. The results below indicate that the undo tablespace in your environment is `UNDOTBS1`.

```
SQL> SHOW PARAMETER undo_tablespace
NAME                                 TYPE        VALUE

undo_tablespace                      string      UNDOTBS1
SQL>
```

6. View the `COMPATIBLE` parameter. This parameter specifies the release with which Oracle must maintain compatibility. It enables you to use a new release of Oracle, while at the same time guaranteeing backward compatibility with an earlier release. This is helpful if it becomes necessary to revert to the earlier release. By default, the value for the compatible entry for this parameter is equal to the version of the Oracle Database that you have installed.

```
SQL> SHOW PARAMETER compatible
NAME                                 TYPE        VALUE

compatible                           string      18.0.0
noncdb_compatible                    boolean     FALSE
SQL>
```

7. View the `CONTROL_FILES` initialization parameter. This parameter specifies one or more control files, separated by commas, and including paths. One to eight file names are listed. Oracle strongly recommends that you multiplex and mirror control files. The output has been formatted for legibility.

```
SQL> SHOW PARAMETER control_files
NAME                                 TYPE
------------------------------------ -----------
control_files                        string

VALUE
------------------------------------------------------------------------------
/u02/app/oracle/oradata/ORCL/control01.ctl,
/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl
SQL>
```

8. View the `PROCESSES`, `SESSIONS`, and `TRANSACTIONS` initialization parameters.

   a. View the `PROCESSES` parameter. This parameter specifies the maximum number of operating system user processes that can simultaneously connect to an Oracle server. This value should allow for all background processes and user processes. The default values of the `SESSIONS` and `TRANSACTIONS` initialization parameters are derived from the `PROCESSES` parameter. Therefore, if you change the value of `PROCESSES`, you should evaluate whether to adjust the values of those derived parameters. The range of values is from six to an OS-dependent value. The default value is dynamic and dependent on the number of CPUs.

```
SQL> SHOW PARAMETER processes
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------
aq_tm_processes                      integer     1
db_writer_processes                  integer     1
gcs_server_processes                 integer     0
global_txn_processes                 integer     1
job_queue_processes                  integer     4000
log_archive_max_processes            integer     4
processes                            integer     300
SQL>
```

   b. View the `SESSIONS` parameter. This parameter specifies the maximum number of sessions that can be created in the system. Because every login requires a session, this parameter effectively determines the maximum number of concurrent users in the system. Notice in the results that the session entry has a value of 472. You should always set this parameter explicitly to a value equivalent to your estimate of the maximum number of concurrent users, plus the number of background processes, plus approximately 10% for recursive sessions.

```
SQL> SHOW PARAMETER sessions
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------
java_max_sessionspace_size           integer     0
java_soft_sessionspace_limit         integer     0
license_max_sessions                 integer     0
license_sessions_warning             integer     0
sessions                             integer     472
shared_server_sessions               integer
SQL>
```

   c. View the `TRANSACTIONS` parameter. This parameter specifies how many rollback segments to bring online when the `UNDO_MANAGEMENT` initialization parameter is equal to `MANUAL`. A transaction is assigned to a rollback segment when the transaction starts, and it can't change for the life of the transaction. A transaction table exists in the rollback segment header with limited space, limiting how many transactions a single segment can support. Therefore, X number of concurrent transactions require at least X number of rollback segments. With Oracle Automatic Undo Management, the database creates rollback segments, brings them online, takes them offline, and drops them as needed.

```
SQL> SHOW PARAMETER transactions
NAME                                 TYPE        VALUE

transactions                         integer     519
transactions_per_rollback_segment    integer     5
SQL>
```

9. View the configuration for the `DB_FILES` initialization parameter. This parameter specifies the maximum number of database files that can be opened for this database. The range of values is OS-dependent.

```
SQL> SHOW PARAMETER db_files
NAME                                 TYPE        VALUE

db_files                             integer     500
SQL>
```

View Advanced Parameters

In this section, you use the `SHOW PARAMETER` command to view advanced parameters.

1. View the `COMMIT_LOGGING` parameter. This parameter is used to control how redo is batched by the Log Writer process. There is no default value, as shown below. You can modify this parameter in a PDB.

```
SQL> SHOW PARAMETER commit_logging
NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------------
commit_logging                       string
SQL>
```

2. View the `COMMIT_WAIT` parameter. This parameter is used to control when the redo for a commit is flushed to the redo logs. There is no default value.

```
SQL> SHOW PARAMETER commit_wait
NAME                                 TYPE        VALUE

commit_wait                          string
SQL>
```

3. View the `SHARED_POOL_SIZE` parameter. This parameter specifies the size of the shared pool in bytes. The shared pool contains objects such as shared cursors, stored procedures, control structures, and parallel execution message buffers. The range of values is OS-dependent. The default value is zero if the `SGA_TARGET` parameter is set. Otherwise, the value is 128 MB for a 64-bit platform or 48 MB for a 32-bit platform.

```
SQL> SHOW PARAMETER shared_pool_size
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
shared_pool_size                     big integer 0
SQL>
```

4. View the `DB_BLOCK_SIZE` parameter. This parameter specifies the standard Oracle database block size (in bytes) and is used by all tablespaces by default. Its value is set during database creation and cannot be subsequently changed. The range of values is from 2048 to 32768 (OS-dependent). The default value is 8192.

```
SQL> SHOW PARAMETER db_block_size
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
db_block_size                        integer     8192
SQL>
```

5. View the `DB_CACHE_SIZE` initialization parameter. You configure this parameter to specify the size of the standard block buffer cache (default buffer pool). The range of values is at least 4 MB times the number of CPUs. Smaller values are automatically rounded up to this value. The default value is zero if the SGA_TARGET initialization parameter is set, otherwise the larger of 48 MB or (4 MB*CPU_COUNT).

```
SQL> SHOW PARAMETER db_cache_size
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
db_cache_size                        big integer 0
SQL>
```

6. View the `UNDO_MANAGEMENT` parameter. This parameter specifies the undo space management mode that the system should use. When set to `AUTO`, the instance is started in automatic undo management mode. Otherwise, it is started in rollback undo mode. In rollback undo mode, undo space is allocated as rollback segments. In automatic undo mode, undo space is allocated as undo tablespaces. The value is `AUTO` or `MANUAL`. If the `UNDO_MANAGEMENT` parameter is omitted when the instance is started, the default value `AUTO` is used.

```
SQL> SHOW PARAMETER undo_management
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
undo_management                      string      AUTO
SQL>
```

7. View the `MEMORY_TARGET` and `MEMORY_MAX_TARGET` parameters. `MEMORY_TARGET` specifies the Oracle system-wide usable memory. The database server tunes memory to the `MEMORY_TARGET` value, reducing or enlarging the SGA and PGA as needed. `MEMORY_MAX_TARGET` sets a maximum value for `MEMORY_TARGET`.

In a PFILE, if you omit `MEMORY_MAX_TARGET` and include a value for `MEMORY_TARGET`, the database automatically sets `MEMORY_MAX_TARGET` to the value of `MEMORY_TARGET`. If you omit the line for `MEMORY_TARGET` and include a value for `MEMORY_MAX_TARGET`, the `MEMORY_TARGET` parameter defaults to zero. After startup, you can dynamically change `MEMORY_TARGET` to a nonzero value if it does not exceed the value of `MEMORY_MAX_TARGET`. For `MEMORY_TARGET`, values range from `152 MB` to `MEMORY_MAX_TARGET`.

a. View the `MEMORY_TARGET` parameter.

```
SQL> SHOW PARAMETER memory_target
NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------------
memory_target                        big integer 0
SQL>
```

b. View the `MEMORY_MAX_TARGET` parameter.

```
SQL> SHOW PARAMETER memory_max_target
NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------------
memory_max_target                    big integer 0
SQL>
```

8. View the `PGA_AGGREGATE_TARGET` parameter. This parameter specifies the amount of Program Global Area (PGA) memory available to all server processes attached to the database instance. This memory does not reside in the System Global Area (SGA). The database uses this parameter as a target amount of PGA memory to use. When setting this parameter, subtract the SGA from the total memory on the system that is available to the Oracle instance. The minimum value is 10 MB and the maximum value is 4096 GB minus 1. The default value is 10 MB or 20% of the size of the SGA, whichever is greater.

```
SQL> SHOW PARAMETER pga_aggregate_target
NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------------
pga_aggregate_target                 big integer 1837647360
SQL>
```

Query Views for Parameter Values

In this section, you query views to learn about parameters.

1. Query the data dictionary to find views that contain the word "parameter." The query below returns 66 rows. Not all of these views contain information about initialization parameters. Among these rows are the `V$PARAMETER`, `V$SPPARAMETER`, `V$PARAMETER2`, and `V$SYSTEM_PARAMETER` views, which you'll examine next.

```
SQL> SET PAGES 100
SQL> SELECT table_name FROM dict WHERE table_name LIKE
'%PARAMETER%';
```

```
TABLE_NAME
--------------------------------------------------------------------------------
USER_ADVISOR_EXEC_PARAMETERS
USER_ADVISOR_PARAMETERS
USER_ADVISOR_SQLW_PARAMETERS
USER_XS_ACL_PARAMETERS
ALL_APPLY_PARAMETERS
ALL_CAPTURE_PARAMETERS
ALL_XS_ACL_PARAMETERS
…
V$PARAMETER_VALID_VALUES
V$SPPARAMETER
V$SYSTEM_PARAMETER
V$SYSTEM_PARAMETER2
V$SYSTEM_RESET_PARAMETER
V$SYSTEM_RESET_PARAMETER2

66 rows selected.

SQL>
```

2. Explore the `V$PARAMETER` view. This view displays the current parameter values in the current session.

   a. View the columns in the `V$PARAMETER` view by using the `DESCRIBE` command. This command returns column names, whether null values are allowed (`NOT NULL` is displayed if the value cannot be null), and column data types.

   The results below contain a column named `ISSYS_MODIFIABLE`. This column is important because it tells you whether a parameter is static or dynamic. If its value is `FALSE`, then the parameter is static; otherwise it's dynamic. To change a static parameter, you must shut down and restart the database; however, you can modify a dynamic parameter in real time while the database is online.

```
SQL> DESCRIBE v$parameter
Name                                        Null?    Type
 ----------------------------------------- -------- ------------
 NUM                                                 NUMBER
 NAME                                                VARCHAR2(80)
 TYPE                                                NUMBER
 VALUE                                               VARCHAR2(4000)
 DISPLAY_VALUE                                       VARCHAR2(4000)
 DEFAULT_VALUE                                       VARCHAR2(255)
 ISDEFAULT                                           VARCHAR2(9)
 ISSES_MODIFIABLE                                    VARCHAR2(5)
 ISSYS_MODIFIABLE                                    VARCHAR2(9)
```

```
  ISPDB_MODIFIABLE                                  VARCHAR2(5)
  ISINSTANCE_MODIFIABLE                             VARCHAR2(5)
  ISMODIFIED                                        VARCHAR2(10)
  ISADJUSTED                                        VARCHAR2(5)
  ISDEPRECATED                                      VARCHAR2(5)
  ISBASIC                                           VARCHAR2(5)
  DESCRIPTION                                       VARCHAR2(255)
  UPDATE_COMMENT                                    VARCHAR2(255)
  HASH                                              NUMBER
  CON_ID                                            NUMBER

SQL>
```

b. Query NAME, ISSYS_MODIFIABLE, and VALUE in the V$PARAMETER view. The query returns many rows.

The TRANSACTIONS parameter is static as indicated by FALSE in the ISSYS_MODIFIABLE column. The PLSQL_WARNINGS parameter is dynamic as indicated by IMMEDIATE in the ISSYS_MODIFIABLE column.

Optional: Before entering the following command, you can enter SET PAUSE ON to cause a pause after each page output. Press Enter to display each next page. After all pages have been displayed, you can issue the SET PAUSE OFF command to stop this feature.

```
SQL> SELECT name, issys_modifiable, value FROM v$parameter;

NAME                                 ISSYS_MOD    VALUE
------------------------------------ ------------ ----------------

lock_name_space                      FALSE
processes                            FALSE        300
…
multishard_query_data_consistency    IMMEDIATE    strong
multishard_query_partial_results     IMMEDIATE    not allowed

433 rows selected.

SQL>
```

c. Query the V$PARAMETER view again, but this time be more specific. Include a WHERE clause to specify all parameters that contain the word "pool." The query returns eight parameters that contain the word "pool."

```
SQL> COLUMN name FORMAT A30
SQL> COLUMN value FORMAT A10
SQL> SELECT name, value FROM v$parameter
  2  WHERE name LIKE '%pool%';
```

```
NAME                                  VALUE
------------------------------------- -------------
shared_pool_size                      0
large_pool_size                       0
java_pool_size                        0
streams_pool_size                     0
shared_pool_reserved_size             26843545
memoptimize_pool_size                 0
buffer_pool_keep
buffer_pool_recycle
olap_page_pool_size                   0

9 rows selected.

SQL>
```

3. Explore the `V$SPPARAMETER` view. This view contains information about the contents of the server parameter file. If a server parameter file was not used to start the instance, each row of the view will contain `FALSE` in the `ISSPECIFIED` column.

   a. View the columns in the `V$SPPARAMETER` view by using the `DESCRIBE` command.

```
SQL> DESCRIBE v$spparameter
Name                                      Null?    Type
----------------------------------------- -------- ----------------
FAMILY                                             VARCHAR2(80)
SID                                                VARCHAR2(80)
NAME                                               VARCHAR2(80)
TYPE                                               VARCHAR2(11)
VALUE                                              VARCHAR2(255)
DISPLAY_VALUE                                      VARCHAR2(255)
ISSPECIFIED                                        VARCHAR2(6)
ORDINAL                                            NUMBER
UPDATE_COMMENT                                     VARCHAR2(255)
CON_ID                                             NUMBER

SQL>
```

   b. Query `NAME` and `VALUE` in the `V$SPPARAMETER` view. Browse the rows returned by the query. The results below have been formatted for easier viewing and show only a small portion of the results.

```
SQL> SELECT name, value FROM v$spparameter;
NAME                                      VALUE
----------------------------------------- -------------
```

```
lock_name_space
processes                          300
sessions
timed_statistics
timed_os_statistics
…
shrd_dupl_table_refresh_rate
multishard_query_data_consistency
multishard_query_partial_results


437 rows selected.

SQL>
```

4. Explore the `V$PARAMETER2` view. This view contains information about the initialization parameters that are currently in effect for the session, with each parameter value appearing as a row in the view. A new session inherits parameter values from the instance-wide values displayed in the `V$SYSTEM_PARAMETER2` view.

   a. View the columns in the `V$PARAMETER2` view by using the `DESCRIBE` command.

```
SQL> DESCRIBE v$parameter2
Name                                          Null?    Type
--------------------------------------------- -------- ------------------
NUM                                                    NUMBER
NAME                                                   VARCHAR2(80)
TYPE                                                   NUMBER
VALUE                                                  VARCHAR2(4000)
DISPLAY_VALUE                                          VARCHAR2(4000)
ISDEFAULT                                              VARCHAR2(6)
ISSES_MODIFIABLE                                       VARCHAR2(5)
ISSYS_MODIFIABLE                                       VARCHAR2(9)
ISPDB_MODIFIABLE                                       VARCHAR2(5)
ISINSTANCE_MODIFIABLE                                  VARCHAR2(5)
ISMODIFIED                                             VARCHAR2(10)
ISADJUSTED                                             VARCHAR2(5)
ISDEPRECATED                                           VARCHAR2(5)
ISBASIC                                                VARCHAR2(5)
DESCRIPTION                                            VARCHAR2(255)
ORDINAL                                                NUMBER
UPDATE_COMMENT                                         VARCHAR2(255)
CON_ID                                                 NUMBER

SQL>
```

b. Query NAME and VALUE in the V$PARAMETER2 view. Browse the rows returned by the query. The results below have been formatted for easier viewing and show only a very small portion of the results.

```
SQL> SELECT name, value FROM v$parameter2;

NAME                                 VALUE
------------------------------       ----------
lock_name_space
processes                            300
sessions                             472
timed_statistics                     TRUE
timed_os_statistics                  0
resource_limit                       TRUE
…
shrd_dupl_table_refresh_rate         60
multishard_query_data_consistency    strong
multishard_query_partial_results     not allowed


438 rows selected.

SQL>
```

5. Explore the V$SYSTEM_PARAMETER view. This view contains information about the initialization parameters that are currently in effect for the instance.

   a. View the columns in the V$SYSTEM_PARAMETER view by using the DESCRIBE command.

```
SQL> DESCRIBE v$system_parameter
Name                                       Null?    Type
----------------------------------------- -------- ---------------
NUM                                                NUMBER
NAME                                               VARCHAR2(80)
TYPE                                               NUMBER
VALUE                                              VARCHAR2(4000)
DISPLAY_VALUE                                      VARCHAR2(4000)
DEFAULT_VALUE                                      VARCHAR2(255)
ISDEFAULT                                          VARCHAR2(9)
ISSES_MODIFIABLE                                   VARCHAR2(5)
ISSYS_MODIFIABLE                                   VARCHAR2(9)
ISPDB_MODIFIABLE                                   VARCHAR2(5)
ISINSTANCE_MODIFIABLE                              VARCHAR2(5)
ISMODIFIED                                         VARCHAR2(8)
ISADJUSTED                                         VARCHAR2(5)
```

```
ISDEPRECATED                                       VARCHAR2(5)
ISBASIC                                            VARCHAR2(5)
DESCRIPTION                                        VARCHAR2(255)
UPDATE_COMMENT                                     VARCHAR2(255)
HASH                                               NUMBER
CON_ID                                             NUMBER

SQL>
```

b. Query `NAME` and `VALUE` in the `V$SYSTEM_PARAMETER` view. Browse the rows returned by the query. The results below have been formatted for easier viewing and show only a very small portion of the results.

```
SQL> SELECT name, value FROM v$system_parameter;

NAME                                 VALUE
------------------------------------ ----------
lock_name_space
processes                            300
sessions                             472
timed_statistics                     TRUE
timed_os_statistics                  0
resource_limit                       TRUE
…
parallel_servers_target              16
common_user_prefix
multishard_query_data_consistency strong
multishard_query_partial_results     not allowed

457 rows selected.

SQL>
```

6. Exit SQL*Plus.

```
SQL> EXIT
Disconnected from Oracle Database 19c Enterprise Edition Release
18.0.0.0.0 - Production
Version 18.1.0.0.0
[oracle@MYDBCS ~]$
```

Lab 5: Modifying Database parameters

Overview

In this practice, you modify the following kinds of initialization parameters (parameters) with SQL*Plus:

- Session-level parameter
- Dynamic system-level parameter
- Static system-level parameter

Assumptions

You are connected to the compute node as the `oracle` user.

Tasks

Modify a Session-Level Parameter

In this section, you modify the `NLS_DATE_FORMAT` parameter. This parameter defines the default date format to use with the `TO_CHAR` and `TO_DATE` functions. The `NLS_TERRITORY` parameter determines the default value of `NLS_DATE_FORMAT`. `NLS_DATE_FORMAT` is one of the National Language Support (NLS) parameters that you can customize just for your session, therefore making it a session-level parameter. When your session ends, your modification expires, and the parameter is returned to its default value.

1. Start SQL*Plus and log in to the database as the `SYS` user with the `SYSDBA` privilege.

```
$ sqlplus / as sysdba
..
SQL>
```

2. Learn about the `NLS_DATE_FORMAT` parameter by querying the `V$PARAMETER` view. Include a `WHERE` clause to narrow down the query to just the `NLS_DATE_FORMAT` parameter. Remember that in the `V$PARAMETER` view, the parameter names are in lowercase.

```
SQL> SELECT name, isses_modifiable, issys_modifiable,
  2  ispdb_modifiable, value
  3  FROM v$parameter
  4  WHERE name = 'nls_date_format';

NAME                  ISSES  ISSYS_MOD ISPDB          VALUE
---------------------- ------- ------------ ----------------- --------------------
nls_date_format       TRUE   FALSE     TRUE



SQL>
```

3. Find out the default date format for the database by querying the NLS_TERRITORY parameter in the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the NLS_TERRITORY parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase.

```
SQL> SELECT name, value FROM v$parameter
  2  WHERE name = 'nls_territory';

NAME                            VALUE
------------------------------- ------------------------------------------

nls_territory                   AMERICA

SQL>
```

4. Connect to PDB1. Run a simple query against the sample data to view an example of the current default date format in use.

   a. Switch to PDB1 by using the ALTER SESSION command.

```
SQL> ALTER SESSION SET container = PDB1;
Session altered.

SQL>
```

   b. Query the LAST_NAME and HIRE_DATE columns in the HR.EMPLOYEES table. Notice the date format is dd-mon-rr.

```
SQL> SELECT last_name, hire_date FROM hr.employees;
LAST_NAME                 HIRE_DATE
------------------------- ---------
King                      17-JUN-03
Kochhar                   21-SEP-05
De Haan                   13-JAN-01
Hunold                    03-JAN-06
Ernst                     21-MAY-07
…
Mavris                    07-JUN-02
Baer                      07-JUN-02
Higgins                   07-JUN-02
Gietz                     07-JUN-02

107 rows selected.

SQL>
```

5. Modify the NLS_DATE_FORMAT parameter to use the format mon dd yyyy by using the ALTER SESSION command.

```
SQL> ALTER SESSION SET nls_date_format = 'mon dd yyyy';
```

6. Rerun the query against the `HR.EMPLOYEES` table. Notice that the date format has changed from `dd-mon-rr` to `mon dd yyyy`.

```
SQL> SELECT last_name, hire_date FROM hr.employees;
LAST_NAME                 HIRE_DATE
        _________________ ______

King                      jun 17 2003
Kochhar                   sep 21 2005
De Haan                   jan 13 2001
Hunold                    jan 03 2006
Ernst                     may 21 2007
…
Mavris                    jun 07 2002
Baer                      jun 07 2002
Higgins                   jun 07 2002
Gietz                     jun 07 2002

107 rows selected.

SQL>
```

7. Query the `NLS_DATE_FORMAT` parameter again by using the `SHOW PARAMETER` command. The value column now reflects the custom date format.

```
SQL> SHOW PARAMETER nls_date_format
NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------------
nls_date_format                      string      mon dd yyyy
SQL>
```

8. Disconnect from `PDB1` to end your session.

```
SQL> DISCONNECT
Disconnected from Oracle Database 19c Enterprise Edition Release
18.0.0.0.0 - Production
Version 18.1.0.0.0
SQL>
```

9. Connect to `PDB1` again as the `SYSTEM` user by using the Easy Connect syntax. See *Course Practice Environment: Security Credentials* for the `SYSTEM` user password. In Practice 5-3, step 6b, you queried `V$SERVICES`. Append the value in the query results following `ORCL` to `pdb1` to create the service name as shown in this example.

```
SQL> connect
system/password@localhost:1521/PDB1.588436052.oraclecloud.intern
al
Connected.
SQL>
```

10. Rerun the query against the `HR.EMPLOYEES` table. The date format has reverted back to the default format `dd-mon-rr`. A session-level parameter change only lasts for the duration of the session.

```
SQL> SELECT last_name, hire_date FROM hr.employees;
LAST_NAME                 HIRE_DATE
------------------------- ---------
King                      17-JUN-03
Kochhar                   21-SEP-05
De Haan                   13-JAN-01
Hunold                    03-JAN-06
Ernst                     21-MAY-07
…
Mavris                    07-JUN-02
Baer                      07-JUN-02
Higgins                   07-JUN-02
Gietz                     07-JUN-02

107 rows selected.

SQL>
```

11. Query the `NLS_DATE_FORMAT` parameter again by using the `SHOW PARAMETER` command. The `VALUE` column no longer has the custom date format.

```
SQL> SHOW PARAMETER nls_date_format
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
nls_date_format                      string
SQL>
```

Modify a Dynamic System-Level Parameter

In this section, you modify the `JOB_QUEUE_PROCESSES` parameter. This parameter specifies the maximum number of job slaves per database instance that can be created for the execution of `DBMS_JOB`jobs and Oracle Scheduler (`DBMS_SCHEDULER`) jobs.

1. Exit SQL*Plus, and connect to the root container with the `SYSDBA` privilege. If you try to update the `JOB_QUEUE_PROCESSES` parameter from `PDB1`, you'll get an error. Also, you'll need the `SYSDBA` privilege to restart the database instance later on.

```
SQL> exit
```

```
…
[oracle@MYDBCS ~]$ sqlplus / as sysdba
…
SQL>
```

2. Learn about the `JOB_QUEUE_PROCESSES` parameter by querying the `V$PARAMETER` view. Include a `WHERE` clause to narrow down the query to just the `JOB_QUEUE_PROCESSES` parameter. Remember that in the `V$PARAMETER` view, the parameter names are in lowercase.

```
SQL> SELECT name, isses_modifiable, issys_modifiable, value
FROM v$parameter WHERE name = 'job_queue_processes';

NAME                      ISSES  ISSYS_MOD           VALUE
------------------------- -----  ------------------- ----------------
job_queue_processes       FALSE  IMMEDIATE           4000


SQL>
```

3. Change the `JOB_QUEUE_PROCESSES` parameter value to `15` by using the `ALTER SYSTEM` command. Set `SCOPE` equal to `BOTH` so that the change happens in both the database instance memory (which makes the change immediate) and in the SPFILE (which makes the change permanent).

```
SQL> ALTER SYSTEM SET job_queue_processes=15 SCOPE=BOTH;

System altered.

SQL>
```

4. Use the `SHOW PARAMETER` command to verify that the `JOB_QUEUE_PROCESSES` parameter value is now equal to `15`. Notice that only `job` was entered with the `SHOW PARAMETER` command instead of the full name, `job_queue_processes`. Remember, when you use the `SHOW PARAMETER` command, you don't have to enter the full name. The database server will find all parameters that contain the letters `job`. In this example, the database server found two parameters that contain the letters `job`: `job_queue_processes` and `max_datapump_jobs_per_pdb`. The query result indicates that the `job_queue_processes` value in memory is now `15`.

```
SQL> SHOW PARAMETER job

NAME                                    TYPE         VALUE
--------------------------------------- ------------ ----------------
job_queue_processes                     integer      15
max_datapump_jobs_per_pdb               integer      100
SQL>
```

5. Verify that the new value for the JOB_QUEUE_PROCESSES parameter persists after the database instance is restarted.

   a. Shut down the database instance with the IMMEDIATE mode.

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

   b. Start the database instance by using the STARTUP command.

```
SQL> STARTUP
ORACLE instance started.

Total System Global Area 2768239832 bytes
Fixed Size                  8899800 bytes
Variable Size             704643072 bytes
Database Buffers 1979711488 bytes   Redo
Buffers                    74985472 bytes
Database mounted.
Database opened.
SQL>
```

   c. View the configuration for the JOB_QUEUE_PROCESSES parameter again by using the SHOW PARAMETER command. The value is 15, which proves that your change to the parameter persisted after the database instance was restarted.

```
SQL> SHOW PARAMETER job
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
job_queue_processes                  integer     15
max_datapump_jobs_per_pdb            integer     100
SQL>
```

Modify a Static System-Level Parameter

In this section, you modify the SEC_MAX_FAILED_LOGIN_ATTEMPTS parameter. This parameter specifies the number of authentication attempts that can be made by a client on a connection to the server process. These login attempts can be for multiple user accounts in the same connection. After the specified number of failure attempts, the connection will be automatically dropped by the server process.

1. Learn about the SEC_MAX_FAILED_LOGIN_ATTEMPTS parameter by querying the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the SEC_MAX_FAILED_LOGIN_ATTEMPTS parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase. The query results below have been formatted for easier viewing.

```
SQL> SELECT name, isses_modifiable, issys_modifiable, value
FROM v$parameter WHERE name = 'sec_max_failed_login_attempts';

NAME                              ISSES ISSYS_MOD  VALUE
--------------------------------- ---------------- ----------------
sec_max_failed_login_attempts     FALSE FALSE      3


SQL>
```

2. Change the `SEC_MAX_FAILED_LOGIN_ATTEMPTS` parameter value to `2` by using the `ALTER SYSTEM` command. Include the comment `'Reduce for tighter security'` and set the scope equal to `SPFILE` so that the change is made only in the SPFILE. When you specify `SCOPE` as `SPFILE` or as `BOTH`, an optional `COMMENT` clause lets you associate a text string with the parameter update. The comment is written to the SPFILE.

```
SQL> ALTER SYSTEM SET sec_max_failed_login_attempts = 2
COMMENT='Reduce for tighter security.' SCOPE=SPFILE;

System altered.

SQL>
```

3. View the `SEC_MAX_FAILED_LOGIN_ATTEMPTS` parameter value by using the `SHOW PARAMETER` command. The query result indicates that the value hasn't been updated yet. It's still equal to `3` because you need to restart the database instance for the change to take effect, which is required for static parameters.

```
SQL> SHOW PARAMETER sec_max

NAME                                   TYPE        VALUE
-------------------------------------- ----------- ----------------
sec_max_failed_login_attempts          integer     3
SQL>
```

4. Restart the database and then verify that the new value for the `SEC_MAX_FAILED_LOGIN_ATTEMPTS` parameter is updated.

   a. Shut down the database instance with the `IMMEDIATE` mode.

```
SQL> SHUTDOWN immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

   b. Start the database instance by using the `STARTUP` command.

```
SQL> STARTUP
ORACLE instance started.
```

```
Total System Global Area 2768239832 bytes
Fixed Size                  8899800 bytes
Variable Size             704643072 bytes
Database Buffers 1979711488 bytes   Redo
Buffers                    74985472 bytes
Database mounted.
Database opened.
SQL>
```

c. View the `SEC_MAX_FAILED_LOGIN_ATTEMPTS` parameter value again by using the `SHOW PARAMETER` command. The query result indicates that the parameter's value was successfully changed to `2`.

```
SQL> SHOW PARAMETER sec_max
NAME                                  TYPE        VALUE
_____________________________________ ___________ ________________
sec_max_failed_login_attempts         integer     2
SQL>
```

d. View the `NAME` and `UPDATE_COMMENT` columns in the `V$PARAMETER` view for the `SEC_MAX_FAILED_LOGIN_ATTEMPTS` parameter. Notice that the comment you added is stored in this view. The results below are formatted for easier reading.

```
SQL> SELECT name, update_comment
FROM v$parameter WHERE name='sec_max_failed_login_attempts';

NAME                             UPDATE_COMMENT
-------------------------------- -------------------------------
sec_max_failed_login_attempts    Reduce for tighter security.

SQL>
```

5. Exit SQL*Plus and close the terminal window.

```
SQL> EXIT
```

Lab 6: Starting and Stopping the container

1. Source the `oraenv` script.

```
[oracle@MYDBCS ~]$ . oraenv
ORACLE_SID = [ORCL] ?
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@MYDBCS ~]$
```

2. Start SQL*Plus and log in to the database as the `SYS` user with the `SYSDBA` privilege.

```
$ sqlplus / as sysdba
…
SQL>
```

3. Shut down the database instance in `NORMAL` mode. Normal is the default shutdown mode if no mode is specified. During this mode of shutdown, the database instance closes the database—all data files and online redo log files are closed. Next, the database instance dismounts the database—all control files associated with the database instance are closed. Lastly, the Oracle software shuts down the database instance—background processes are terminated, and the System Global Area (SGA) is removed from memory. When a database instance shuts down in normal mode, the database instance waits for all users to disconnect before completing the shutdown, and no new connections are allowed. Control is not returned to the session that initiates a database shutdown until shutdown is complete.

```
SQL> SHUTDOWN
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

4. Show the current user. Note that SQL*Plus is still running and the current user is `SYS`.

```
SQL> SHOW USER
USER is "SYS"
SQL>
```

5. Show the current container name. This step returns an error because the database is shut down.

```
SQL> SHOW con_name
ERROR:
ORA-01034: ORACLE not available
Process ID: 0
Session ID: 0 Serial number: 0


SP2-1545: This feature requires Database availability.
SQL>
```

6. Start up the database instance in `NOMOUNT` mode. During this step, the Oracle software locates the parameter file (SPFILE or PFILE), allocates memory to the System Global Area (SGA), starts the background processes, and opens the alert log and trace files . At this stage, the database instance is started; however, users cannot access it yet. You would usually start in `NOMOUNT` mode if you were creating a database, re-creating control files, or performing certain backup and recovery tasks.

```
SQL> STARTUP NOMOUNT
ORACLE instance started.

Total System Global Area 2768239832 bytes
Fixed Size                  8899800 bytes
Variable Size             704643072 bytes
Database  Buffers 1979711488  bytes   Redo
Buffers                    74985472 bytes
SQL>
```

7. Mount the database by using the `ALTER DATABASE MOUNT` command. During this step, the database instance mounts the database. This means that the database instance locates and opens all the control files specified in the initialization parameter file and reads the control files to obtain the names and statuses of the data files and online redo log files. The database instance does not, however, verify the existence of the data files and online redo log files at this time. You must mount the database, but not open it when you want to rename data files, enable/disable online redo log file archiving options, or perform a full database recovery.

```
SQL> ALTER DATABASE MOUNT;
Database altered.

SQL>
```

8. Open the database by using the `ALTER DATABASE` command. During this step, the database instance opens the data files for the CDB and online redo log files and checks the consistency of the database. When the database is open, all users can access the database instance.

```
SQL> ALTER DATABASE OPEN;
Database altered.

SQL>
```

9. Show the current container name.

```
SQL> SHOW con_name

CON_NAME
------------------------------
CDB$ROOT
SQL>
```

10. Show the current user.

```
SQL> SHOW user
USER is "SYS"
SQL>
```

11. Determine the state of the database: select open_mode from v$database;

12. Check whether PDB1 is open by querying the OPEN_MODE column in the V$PDBS view.

```
SQL> COLUMN con_id FORMAT 999
SQL> COLUMN name FORMAT A10
SQL> SELECT con_id, name, open_mode FROM v$pdbs;
```

Lab 7: View Diagnostic information

1. Start SQL*Plus and log in to the database as the `SYS` user with the `SYSDBA` privilege.

```
[oracle@MYDBCS ~]$ sqlplus / as sysdba
…
SQL>
```

2. View the locations of the various diagnostics directories in the ADR. The results below have been formatted for easier reading.
   - The path that corresponds to the `Diag Alert` entry in the `NAME` column is for the XML version. This path is `/u01/app/oracle/diag/rdbms/orcl/ORCL/alert`.
   - The path that corresponds to the `Diag Trace` entry is for the text-only version. This path is `/u01/app/oracle/diag/rdbms/orcl/ORCL/trace`.

```
SQL> SELECT name, value FROM v$diag_info;
```

```

NAME                 VALUE
-------------------- -------------------------------------------
Diag Enabled         TRUE
ADR Base             /u01/app/oracle
ADR Home             /u01/app/oracle/diag/rdbms/orcl/ORCL
Diag Trace           /u01/app/oracle/diag/rdbms/orcl/ORCL/trace
Diag Alert           /u01/app/oracle/diag/rdbms/orcl/ORCL/alert
Diag Incident        /u01/app/oracle/diag/rdbms/orcl/ORCL/incident
Diag Cdump           /u01/app/oracle/diag/rdbms/orcl/ORCL/cdump
Health Monitor       /u01/app/oracle/diag/rdbms/orcl/ORCL/hm
Default Trace File
/u01/app/oracle/diag/rdbms/orcl/ORCL/trace/ORCL_ora_2600.trc
Active Problem Count  1
Active Incident Count 6

11 rows selected.

SQL>
```

3. Exit SQL*Plus.

```
SQL> EXIT
```

Use an Editor to View the Alert Log

1. View the XML version of the alert log. The `log.xml` file is the XML version of the alert log.

   a. Browse to the `/u01/app/oracle/diag/rdbms/orcl/ORCL/alert` directory.

   ```
   [oracle@MYDBCS ~]$ cd /u01/app/oracle/diag/rdbms/orcl/ORCL/alert
   [oracle@MYDBCS alert]$
   ```

   b. List the contents of the directory. Notice that there is a `log.xml` file in this directory.

   ```
   [oracle@MYDBCS alert]$ ls
   log.xml
   [oracle@MYDBCS alert]$
   ```

   c. Use `cat` or `more` to scroll through the file. Notice that it is a chronological log of messages about non-default initialization parameters used at startup, errors, SQL statements, and so on. Oracle Database uses the alert log to keep a record of these events as an alternative to displaying the information on an operator's console.

   ```
   [oracle@MYDBCS alert]$ more log.xml
   <msg time='2018-03-07T22:19:08.858+00:00' org_id='oracle'
   comp_id='rdbms'
    msg_id='opistr_real:1244:2538814769' type='NOTIFICATION'
   group='startup'
    level='16' host_id='MYDBCS' host_addr='10.18.24.38'
    pid='8090' version='1' con_uid='1'
   ```

```
 con_id='1' con_name='CDB$ROOT'>
 <txt>Starting ORACLE instance (normal) (OS id: 8090)
…
```

2. View the text-only version of the alert log.

   a. Change to the `/u01/app/oracle/diag/rdbms/orcl/ORCL/trace` directory.

```
[oracle@MYDBCS alert]$ cd
/u01/app/oracle/diag/rdbms/orcl/ORCL/trace
[oracle@MYDBCS trace]$
```

   b. The `alert_ORCL.log` (format is `alert_SID.log`) file is the text-only version. In this directory, you also have server process trace files (TRC files) and trace map files(TRM files). Each server and background process can write to an associated trace file. When a process detects an internal error, it dumps information about the error to its trace file. Trace map files contain structural information about trace files and are used for searching and navigation.

```
[oracle@MYDBCS trace]$ ls
alert_ORCL.log                ORCL_ora_12783.trc
ORCL_ora_27673.trc
alert_ORCL_v20180310.log.gz   ORCL_ora_12783.trm
ORCL_ora_27673.trm
ORCL_aqpc_10857.trc           ORCL_ora_12785.trc
ORCL_ora_27674.trc
…
[oracle@MYDBCS trace]$
```

   c. Open the file with an editor or use a command such as `tail` to view the contents of the alert log.

```
[oracle@MYDBCS trace]$ tail -500 alert_ORCL.log
2018-03-15T20:41:22.507272+00:00
db_recovery_file_dest_size of 6144 MB is 33.63% used. This is a
user-specified limit on the amount of space that will be used by
this
database for recovery-related files, and does not reflect the
amount of
space available in the underlying filesystem or ASM diskgroup.
…
Pluggable database PDB1 opened read write
2018-03-16T15:10:48.837547+00:00
Completed: ALTER PLUGGABLE DATABASE ALL OPEN
Starting background process CJQ0
Completed: ALTER DATABASE OPEN
2018-03-16T15:10:49.521562+00:00
CJQ0 started with pid=65, OS id=4909
[oracle@MYDBCS trace]$
```

Use ADRCI to View the Alert Log

1. Start the ADRCI tool. Recall that you set the Oracle environment variables at the beginning of this practice; however, only the `ORACLE_HOME` environment variable needs to be set prior to starting ADRCI. If you ever need to set just that one variable, you can do so by entering the following at the command prompt: `export PATH=$PATH:$ORACLE_HOME/bin`.

```
[oracle@MYDBCS trace]$ adrci
```

2. View the alert log by using the `SHOW ALERT` command. The `SHOW ALERT` command opens the alert log file in the `vi` editor, by default.

```
adrci> SHOW ALERT
ADR Home = /u01/app/oracle/diag/rdbms/orcl/ORCL:
****************************************************************
*********
Output the results to file: /tmp/alert_11237_1404_ORCL_1.ado
2018-03-07 22:19:08.858000 +00:00
Starting ORACLE instance (normal) (OS id: 8090)
****************************************************************
******
Dump of system resources acquired for SHARED GLOBAL AREA (SGA)
 Per process system memlock (soft) limit = 128G
 Expected per process system memlock (soft) limit to lock
 SHARED GLOBAL AREA (SGA) into memory: 2642M
 Available system pagesizes:
  4K, 2048K
…
```

3. Enter **G** to move to bottom of the alert file.

```
2018-03-16 15:10:45.273000 +00:00
Opening pdb with no Resource Manager plan active
2018-03-16 15:10:47.226000 +00:00
Pluggable database PDB1 opened read write
2018-03-16 15:10:48.837000 +00:00
Completed: ALTER PLUGGABLE DATABASE ALL OPEN
```

```
Starting background process CJQ0
Completed: ALTER DATABASE OPEN
CJQ0 started with pid=65, OS id=4909
```

4. Enter **?Starting ORACLE instance?** and press return. Press **N** to search from thebottom of the file to find the last time the instance was started. The following will be similar to your alert log. Note: Here lowercase and uppercase are important because `vi` distinguishes them, unless you ignore them by setting `:set ic`.

```
Starting ORACLE instance (normal) (OS id: 8090)
***************************************************************
******
Dump of system resources acquired for SHARED GLOBAL AREA (SGA)
 Per process system memlock (soft) limit = 128G
 Expected per process system memlock (soft) limit to lock
 SHARED GLOBAL AREA (SGA) into memory: 2642M
 Available system pagesizes:
  4K, 2048K
 Supported system pagesize(s):
  PAGESIZE  AVAILABLE_PAGES  EXPECTED_PAGES  ALLOCATED_PAGES
ERROR(s)
        4K       Configured               4          675844
NONE
     2048K                0            1321               0
NONE
RECOMMENDATION:
 1. For optimal performance, configure system with expected
number
 of pages for every supported system pagesize prior to the next
 instance restart operation.
***************************************************************
******
LICENSE_MAX_SESSION = 0
LICENSE_SESSIONS_WARNING = 0
Initial number of CPU is 2
Number of processor cores in the system is 2
Number of processor sockets in the system is 1
search hit BOTTOM, continuing at TOP
```

5. Search forward by entering **/ ALTER** to find the line that starts with `ALTER DATABASE MOUNT`. Here lowercase and uppercase are important because `vi` distinguishes them.

```
ALTER DATABASE   MOUNT
2018-03-07 22:21:04.104000 +00:00
Using default pga_aggregate_limit of 3505 MB
2018-03-07 22:21:06.471000 +00:00
```

```
.... (PID:9128): Redo network throttle feature is disabled at
mount time
Successful mount of redo thread 1, with mount id 2299076813
Database mounted in Exclusive Mode
Lost write protection disabled
.... (PID:9128): Using STANDBY_ARCHIVE_DEST parameter default
value as USE_DB_RECOVERY_FILE_DEST [krsd.c:17695]
Completed: ALTER DATABASE   MOUNT
```

6. Search forward again by entering **/ ALTER** to find the line that starts with `ALTER DATABASE OPEN`. Notice that the stages that the database goes through during startup are `MOUNT` and `OPEN`.

```
ALTER DATABASE OPEN
Ping without log force is disabled:
  instance mounted in exclusive mode.
Buffer Cache Full DB Caching mode changing from FULL CACHING
DISABLED to FULL CACHING ENABLED
Crash Recovery excluding pdb 2 which was cleanly closed.
Crash Recovery excluding pdb 3 which was cleanly closed.
2018-03-07 22:21:08.617000 +00:00
Endian type of dictionary set to little
LGWR (PID:9095): STARTING ARCH PROCESSES
Starting background process ARC0
```

7. Exit the `vi` editor by entering **:q** and pressing Enter.

8. Exit adrci and close the terminal window.

```
adrci > exit
[oracle@MYDBCS trace]$
```

Log DDL Statements in the DDL Log File

1. Determine if DDL logging is enabled in `PDB1`. If not, enable it by setting the value for the `ENABLE_DDL_LOGGING` initialization parameter to `TRUE`.

   a. Start SQL*Plus and log in to the database as the `SYS` user with the `SYSDBA` privilege.

```
[oracle@MYDBCS trace]$ sqlplus / as sysdba
…
SQL>
```

   b. Switch to `PDB1`.

```
SQL> ALTER SESSION SET CONTAINER = PDB1;
Session altered.

SQL>
```

c. Issue the `SHOW PARAMETER` command to view the value for `ENABLE_DDL_LOGGING`. In Oracle Database Cloud Service, `ENABLE_DDL_LOGGING` is set to `TRUE` by default. The default value for `ENABLE_DDL_LOGGING` is `FALSE` in non-Cloud installations.

```
SQL> SHOW PARAMETER enable_ddl_logging
NAME                                    TYPE        VALUE
--------------------------------------- ----------- ----------------
enable_ddl_logging                      boolean     TRUE
SQL>
```

d. If DDL logging was not enabled, you could enable it for just this session by using the `ALTER SESSION` command.

```
SQL> ALTER SESSION SET enable_ddl_logging = TRUE;
Session altered.

SQL>
```

2. Create and drop a table to generate statements that will logged.

```
SQL> CREATE TABLE TEST (name varchar2(15));
Table created.
SQL> DROP TABLE TEST;
Table dropped.
SQL>
```

3. Exit SQL*Plus.

```
SQL> EXIT
…
[oracle@MYDBCS trace]$
```

4. Change to the directory where the text version of the DDL log file resides.

```
[oracle@MYDBCS trace]$ cd
/u01/app/oracle/diag/rdbms/orcl/ORCL/log
[oracle@MYDBCS log]$
```

5. List the contents of the log directory.

```
[oracle@MYDBCS log]$ ls
ddl  ddl_ORCL.log  debug  debug.log  hcs  imdb  test
[oracle@MYDBCS log]$
```

6. View the `ddl_ORCL.log` file by using the `cat` command. Your output will be different from the output shown below.

```
[oracle@MYDBCS log]$ cat ddl_ORCL.log
2018-03-16T19:31:30.795903+00:00
diag_adl:CREATE TABLE TEST (name varchar2(15) )
2018-03-16T19:31:57.762139+00:00
diag_adl:DROP TABLE TEST
[oracle@MYDBCS log]$
```

Lab 8: Creating a Local Users

Overview

In this practice, you log in to PDB1 as the local administrator (PDB1_ADMIN1) and create a local user account called INVENTORY, which will own the new Inventory software application. INVENTORY is an example of a user account that does not represent a person.

Assumptions

You are logged in to the compute node as the oracle user.

Tasks

Create the INVENTORY User Account

1. Start SQL*Plus and connect to PDB1 as the PDB1_ADMIN1 user.

```
[oracle@MYDBCS ~]$ sqlplus PDB1_ADMIN1/password@PDB1
…
SQL>
```

2. Create a local user account named INVENTORY. Setthe default tablespace to the USERS tablespace and grant unlimited quota on that tablespace. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
SQL> CREATE USER INVENTORY IDENTIFIED BY password DEFAULT
TABLESPACE users QUOTA UNLIMITED ON users;

User created.

SQL>
```

3. Grant the CREATE SESSION privilege to INVENTORY.

```
SQL> GRANT CREATE SESSION TO INVENTORY;

Grant succeeded.

SQL>
```

4. List the local user accounts for PDB1 by querying the DBA_USERS view. The INVENTORY account is included in the list.

```
SQL> SELECT DISTINCT username FROM dba_users WHERE common='NO'
ORDER BY username;

USERNAME
--------------------------------------------------------------------------------
APEX_050100
APEX_INSTANCE_ADMIN_USER
```

```
APEX_LISTENER
APEX_PUBLIC_USER
APEX_REST_PUBLIC_USER
FLOWS_FILES
HR
INVENTORY
PDB1_ADMIN1
PDBADMIN
SCOTT

11 rows selected.

SQL>
```

Connect as INVENTORY and Verify Privileges

1. Disconnect `PDB1_ADMIN1` from `PDB1`.

```
SQL> DISCONNECT
…
SQL>
```

2. Verify that the `INVENTORY` user account can connect to `PDB1`.

```
SQL> CONNECT INVENTORY/password@PDB1
Connected.
SQL>
```

3. List the privileges for `INVENTORY` by querying the `SESSION_PRIVS` view. The results show that `INVENTORY` has the `CREATE SESSION` privilege.

```
SQL> SELECT * FROM session_privs ORDER BY privilege;

PRIVILEGE
----------------------------------------
CREATE SESSION

SQL>
```

4. Exit SQL*Plus.

```
SQL> EXIT
…
[oracle@MYDBCS ~]$
```

Lab 9: Granting Privilege

Explore the Privileges and Roles Granted to `PDBADMIN`

1. Start SQL*Plus and connect as the `SYS` user with the `SYSDBA` privilege.

   Note: `PDBADMIN` does not have the required privileges to view data from the `DBA_SYS_PRIVS` view in `PDB1`, which you will do in the next step.

```
$ sqlplus / AS SYSDBA
...
SQL>
```

2. List the system privileges granted to the `PDBADMIN` user by querying the `DBA_SYS_PRIVS` view. This view describes system privileges granted to users and roles. The results show that no system privileges are explicitly granted to `PDBADMIN`. However, there may be privileges granted through roles.

```
SQL> SELECT * FROM dba_sys_privs WHERE grantee='PDBADMIN';
no rows selected

SQL>
```

3. List the roles granted to the `PDBADMIN` user by querying the `CDB_ROLE_PRIVS` view. This view describes the roles granted to all users and roles in the database. The results show that `PDBADMIN` is granted the `PDB_DBA` role. Also, the `ADMIN OPTION` is enabled (`ADM=YES`), which means that `PDBADMIN` can grant the `PDB_DBA` role to other users.

```
SQL> col granted_role format a10
SQL> SELECT granted_role, admin_option FROM cdb_role_privs WHERE
grantee='PDBADMIN';

GRANTED_RO ADM
---------- ---
PDB_DBA    YES
```

```
DBA          NO

SQL>
```

4. List the system privileges granted to the PDB_DBA role by querying the ROLE_SYS_PRIVS view.
   a. Switch to PDB1. You must be connected to PDB1 to retrieve data, and you must be connected as the SYS user.

```
SQL> ALTER SESSION SET CONTAINER = PDB1;
Session altered.

SQL>
```

   b. Query the ROLE_SYS_PRIVS view. This view describes system privileges granted to roles. Information is provided only about roles to which the user has access. Because you're connected to PDB1 as the SYS user, you have access to all role information. The results show that the PDB_DBA role consists of three system privileges: CREATE SESSION, SET CONTAINER, and CREATE PLUGGABLE DATABASE.

```
SQL> SELECT privilege FROM role_sys_privs WHERE role='PDB_DBA'
ORDER BY privilege;

PRIVILEGE
----------------------------------------
CREATE PLUGGABLE DATABASE
CREATE SESSION
SET CONTAINER

SQL>
```

5. List the roles that are granted to the PDB_DBA role by querying the DBA_ROLE_PRIVS view. The results show that the PDB_DBA role is granted the CONNECT role.

```
SQL> SELECT granted_role FROM dba_role_privs WHERE grantee =
'PDB_DBA';

GRANTED_RO
----------
CONNECT

SQL>
```

6. List the privileges granted to the CONNECT role by querying the ROLE_SYS_PRIVS view. The results show that the CONNECT role consists of the SET CONTAINER and CREATE SESSION privileges.

```
SQL> SELECT privilege FROM role_sys_privs WHERE role='CONNECT'
ORDER BY privilege;
```

```
PRIVILEGE

CREATE SESSION
SET CONTAINER

SQL>
```

7. Let's summarize our findings: From these queries, you learned that the PDBADMIN user is granted the PDB_DBA role by default, and that role consists of the CONNECT role and the CREATE PLUGGABLE DATABASE system privilege. The CONNECT role contains the SET CONTAINER and CREATE SESSION system privileges.

Grant the DBA Role to PDBADMIN

1. Grant the DBA role locally to PDBADMIN.

```
SQL> GRANT dba TO pdbadmin;

Grant succeeded.

SQL>
```

2. List the roles that are granted to PDBADMIN by querying the DBA_ROLE_PRIVS view. The results show that PDBADMIN is now granted the DBA and PDB_DBA roles.

```
SQL> SELECT granted_role FROM dba_role_privs WHERE grantee =
'PDBADMIN' ORDER BY granted_role;

GRANTED_RO
-------------
DBA
PDB_DBA

SQL>
```

3. Exit SQL*Plus and close the terminal window.

```
SQL> EXIT
...
[oracle@MYDBCS ~]$ exit
```

Lab 10: Creating a Default Role for a User

Tasks

Configure a Default Role for JGOODMAN

1. Start SQL*Plus and connect to PDB1 as the PDBADMIN user.

```
$ sqlplus PDBADMIN/password@PDB1
…
SQL>
```

2. View the current roles for JGOODMAN by querying the DBA_ROLE_PRIVS view. Also, show whether the roles are default roles. The results show that JGOODMAN is granted two roles, HRMANAGER and HRCLERK, and both are default roles (the DEF column = YES).

```
SQL> COLUMN granted_role FORMAT A20
SQL> SELECT granted_role, default_role FROM dba_role_privs WHERE
grantee='JGOODMAN';

GRANTED_ROLE         DEF
-------------------- ----
HRCLERK              YES
HRMANAGER            YES

SQL>
SQL> ALTER USER JGOODMAN DEFAULT ROLE HRCLERK;

User altered.

SQL>
```

3. Set the default role for JGOODMAN to be HRCLERK only by using the ALTER USER command and DEFAULT ROLE clause.

4. View the current roles and default role settings for JGOODMAN again by querying the DBA_ROLE_PRIVS view. The results show that the default role is HRCLERK and the HRMANAGER role is no longer a default role. Jenny still has this role; however, she'll need to enable it to exercise its privileges.

```
SQL> SELECT granted_role, default_role FROM dba_role_privs WHERE
grantee='JGOODMAN';


GRANTED_ROLE          DEF
--------------------- ----
HRCLERK               YES
HRMANAGER             NO


SQL>
```

5. Disconnect PDBADMIN from PDB1.

```
SQL> DISCONNECT
…
SQL>
```

Enable a Non-Default Role

1. Connect to PDB1 as JGOODMAN.

```
SQL> CONNECT JGOODMAN/password@PDB1
Connected.
SQL>
```

2. View the roles for the current session. Notice that the default role, HRCLERK, is in effect.

```
SQL> SELECT * FROM session_roles;

ROLE

--------------------------------------------------------------------------------
HRCLERK


SQL>
SQL> SET ROLE HRMANAGER;

Role set.

SQL>
```

3. Suppose JGOODMAN needs to operate as an HR Manager, and not an HR Clerk. Change the enabled role to HRMANAGER. Caution: If you use the SET ROLE command, any roles not included in the command will be disabled.

4. View the roles for the current session again. The HRMANAGER role is now enabled.

```
SQL> SELECT * FROM session_roles;

ROLE
--------------------------------------------------------------------------------
HRMANAGER

SQL>
```

5. Suppose JGOODMAN needs both roles. Use the SET ROLE command to enable them both.

```
SQL> SET ROLE HRMANAGER, HRCLERK;

Role set.

SQL>
```

6. View the roles for the current session again. The HRMANAGER and HRCLERK roles are now in effect.

```
SQL> SELECT * FROM session_roles;

ROLE
--------------------------------------------------------------------------------
HRCLERK
HRMANAGER

SQL>
```

7. Exit SQL*Plus.

```
SQL> EXIT
…
[oracle@MYDBCS ~]$
```

Lab 11 : Creating a PDB from Seed

Overview

In this practice, you create an empty PDB named `PDB2` in your CDB by using the seed PDB.

Note: You can use Database Configuration Assistant, SQL Developer, or SQL commands to create a PDB from seed. This practice shows you how to do it by using SQL commands in SQL*Plus.

Assumptions

You are logged in as the `oracle` user.

Tasks

1. Open a new terminal window and connect to the compute node as the `oracle` user.

```
$ sudo su - oracle
```

2. Source the `oraenv` script.

```
[oracle@MYDBCS ~]$ . oraenv
ORACLE_SID = [ORCL] ?
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@MYDBCS ~]$
```

3. Start SQL*Plus and log in to your CDB with the `SYSDBA` privilege.

```
[oracle@MYDBCS ~]$ sqlplus / as sysdba
…
SQL>
```

4. Create `PDB2` by using the `CREATE PLUGGABLE DATABASE` command. Specify an admin user named `PDB2ADMIN` and grant this user the `DBA` role. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
SQL> CREATE PLUGGABLE DATABASE PDB2
  2  ADMIN USER PDB2ADMIN IDENTIFIED BY password
  3  ROLES=(dba);

Pluggable database created.

SQL>
```

In a non-DBCS installation of Oracle Database, the seed PDB does not have a `USERS` tablespace. You can include the `DEFAULT TABLESPACE USERS` clause to create a default permanent tablespace for any non-administrative users for which you do not specify a different permanent tablespace as shown in this example:

```
CREATE PLUGGABLE DATABASE PDB2
…
  DEFAULT TABLESPACE USERS
  DATAFILE '/u02/app/oracle/oradata/ORCL/PDB2/users01.dbf'
  SIZE 250M AUTOEXTEND ON
…
```

In DBCS, OMF is enabled by default, so the datafiles for PDB2 will be created in the location set by the `DB_CREATE_FILE_DEST` initialization parameter. In a database that is not OMF-enabled, you can specify the target location of the data files by using the `FILE_NAME_CONVERT` clause. This clause enables you to specify the target locations of the files based on the names of the source files. The first parameter in the clause is the source directory of the seed data files. The second is the destination directory for the new PDB data files. Here is an example using the `FILE_NAME_CONVERT` clause:

```
CREATE PLUGGABLE DATABASE PDB2
…
  FILE_NAME_CONVERT=
  ('/u02/app/oracle/oradata/ORCL/pdbseed/',
   '/u02/app/oracle/oradata/ORCL/PDB2/',
   '/u04/app/oracle/oradata/temp/',
   '/u04/app/oracle/oradata/temp/PDB2/')
…
```

5. Open `PDB1`.

   a. View the open mode for `PDB1`. After a PDB is created, its open mode is `MOUNTED`. When a PDB is in mounted mode, it behaves like a CDB in mounted mode. It does not allow changes to any objects, and it is accessible only to database administrators connected as `SYSDBA`. Information about the PDB is removed from memory caches. Cold backups of the PDB are possible.

```
SQL> COLUMN con_id FORMAT 999
SQL> COLUMN name FORMAT A10
SQL> SELECT con_id, name, open_mode FROM v$pdbs;

CON_ID NAME       OPEN_MODE
------ ---------- ----------
     2 PDB$SEED   READ ONLY
     3 PDB1       READ WRITE
     4 PDB2       MOUNTED
```

   b. Open `PDB2` by using the `ALTER PLUGGABLE DATABASE` command.

```
SQL> ALTER PLUGGABLE DATABASE PDB2 OPEN;
```

```
Pluggable database altered.

SQL>
```

c. Verify that the open mode for `PDB2` is now `READ WRITE`.

```
SQL> SELECT con_id, name, open_mode FROM v$pdbs;

CON_ID NAME        OPEN_MODE
-------- ------------- -------------
     2 PDB$SEED    READ ONLY
     3 PDB1        READ WRITE
     4 PDB2        READ WRITE
SQL>
```

6. View the list of services registered with the listener. When you create a PDB, a service is created and started. The name of the service is the same name as the PDB. You will connect to this service in the next step.

```
SQL> !lsnrctl status
LSNRCTL for Linux: Version 18.0.0.0.0 - Production on 22-MAR-
2018 16:00:59

Copyright (c) 1991, 2017, Oracle.  All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MYDBCS.compute-
588436052.oraclecloud.internal)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 18.0.0.0.0
- Production
Start Date                19-MAR-2018 15:23:07
Uptime                    3 days 0 hr. 37 min. 52 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/u01/app/oracle/product/18.0.0/dbhome_1/network/admin/listener.o
ra
Listener Log File
/u01/app/oracle/diag/tnslsnr/MYDBCS/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=MYDBCS.compute-
588436052.oraclecloud.internal)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=MYDBCS.compute-
588436052.oraclecloud.internal)(PORT=5500))(Security=(my_wallet_
directory=/u01/app/oracle/admin/ORCL/xdb_wallet))(Presentation=H
TTP)(Session=RAW))
Services Summary...
Service
"66db11a937912ac9e0532618120ab4b9.588436052.oraclecloud.internal
" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
Service
"68035adc452a1241e0532618120ad62d.588436052.oraclecloud.internal
" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
Service "ORCL.588436052.oraclecloud.internal" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
Service "ORCL.588436052.oraclecloud.internalXDB" has 1
instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
Service "PDB1.588436052.oraclecloud.internal" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
Service "PDB2.588436052.oraclecloud.internal" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
The command completed successfully

SQL>
```

7. Connect to `PDB2` as the `PDB2ADMIN` user by using the Easy Connect method.

```
SQL> CONNECT
PDB2ADMIN/password@localhost:1521/PDB2.588436052.oraclecloud.int
ernal
Connected.
SQL>
```

Note: Alternatively, you could have switched to `PDB2` by using the `ALTER SESSION` command.

```
ALTER SESSION SET container = PDB2;
```

8. Explore `PDB2`.

   a. Show the current container.

```
SQL> SHOW con_name
```

```
CON_NAME
------------------------------
PDB2
SQL>
```

b. Show the current container ID.

```
SQL> SHOW con_id

CON_ID
------------------------------
4
SQL>
```

c. List the service for `PDB2` by querying the `V$SERVICES` view.

```
SQL> COLUMN name FORMAT A20
SQL> SELECT name FROM v$services;

NAME
--------------------
PDB2

SQL>
```

d. List the data files for `PDB2` and their respective tablespaces by querying the `DBA_DATA_FILES` view. Recall that DBCS uses OMF by default, so the files were created with the OMF file naming format.

```
SQL> col file_name format a55
SQL> col tablespace_name format a10
SQL> SELECT file_name, tablespace_name FROM dba_data_files;

FILE_NAME                                               TABLESPACE
------------------------------------------------------- ----------
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D SYSTEM
0A3C0D/datafile/o1_mf_system_fk2t2tmq_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D SYSAUX
0A3C0D/datafile/o1_mf_sysaux_fk2t2tmv_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D UNDOTBS1
0A3C0D/datafile/o1_mf_undotbs1_fk2t2tmy_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D USERS
0A3C0D/datafile/o1_mf_users_fk2t2tn4_.dbf
SQL>
```

e. List the temp files for `PDB2` by querying the `DBA_TEMP_FILES` view. The query returns one temp file. Your temp file name will be different from the one shown below.

```
SQL> SELECT file_name, tablespace_name FROM dba_temp_files;

FILE_NAME                                                   TABLESPACE
----------------------------------------------------------- ----------
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D TEMP
0A3C0D/datafile/o1_mf_temp_fk2t2tn2_.dbf


SQL>
```

f. List the local users for `PDB2` by querying the `DBA_USERS` view.

```
SQL> SELECT DISTINCT username FROM dba_users WHERE common='NO'
ORDER BY username;

USERNAME
-----------------------------------------------------------------
APEX_050100
APEX_INSTANCE_ADMIN_USER
APEX_LISTENER
APEX_PUBLIC_USER
APEX_REST_PUBLIC_USER
FLOWS_FILES
PDB2ADMIN

7 rows selected.

SQL>
```

g. List the common users for `PDB2` by querying the `DBA_USERS` view.

```
SQL> SELECT DISTINCT username FROM dba_users WHERE common='YES'
ORDER BY username;

USERNAME
-----------------------------------------------------------------
ANONYMOUS
APPQOSSYS
AUDSYS
C##CDB_ADMIN1
…
SYSTEM
WMSYS
XDB
```

```
XS$NULL

39 rows selected.

SQL>
```

h. Exit SQL*Plus.

```
SQL > EXIT
…
[oracle@MYDBCS ~]$
```

9. Add a service name entry to the `tnsnames.ora` file for `PDB2`.

a. Change the directory to `$ORACLE_HOME/network/admin`.

```
[oracle@MYDBCS ~]$ cd $ORACLE_HOME/network/admin
[oracle@MYDBCS admin]$
```

b. View the `tnsnames.ora` file by using the `cat` command.

```
[oracle@MYDBCS admin]$ cat tnsnames.ora

ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-
588436052.oraclecloud.internal)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL.588436052.oraclecloud.internal)
    )
  )


PDB1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-
588436052.oraclecloud.internal)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = pdb1.588436052.oraclecloud.internal)
    )
  )

[oracle@MYDBCS admin]$
```

c. Use an editor such as `vi` to add an entry for `PDB2` to the `tnsnames.ora` file.

```
PDB2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-
588436052.oraclecloud.internal)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = pdb2.588436052.oraclecloud.internal)
    )
  )
```

d. Use the `cat` command to view the `tnsnames.ora` file and ensure that your new entry is formatted correctly.

```
[oracle@MYDBCS admin]$ cat tnsnames.ora

ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-
588436052.oraclecloud.internal)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL.588436052.oraclecloud.internal)
    )
  )

PDB1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-
588436052.oraclecloud.internal)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = pdb1.588436052.oraclecloud.internal)
    )
  )

PDB2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-
588436052.oraclecloud.internal)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = pdb2.588436052.oraclecloud.internal)
    )
  )
[oracle@MYDBCS admin]$
```

10. Connect to `PDB2` by using the new service name and verify the current container.

    a. Start SQL*Plus and connect to `PDB2` as the `SYSTEM` user by using the `PDB2` net service name.

    ```
    [oracle@MYDBCS admin]$ sqlplus system/password@PDB2
    …
    SQL>
    ```

    b. Verify that the current container name is `PDB2`.

    ```
    SQL> SHOW con_name

    CON_NAME
    ------------------------------
    PDB2
    SQL>
    ```

    c. Exit SQL*Plus.

    ```
    SQL> exit
    …
    [oracle@MYDBCS admin]$
    ```

Lab 12: Cloning a PDB

Overview

In this practice, you use SQL*Plus to hot clone `PDB1` as `PDB3` in the CDB.

Tip

Because you use several windows at the same time in this practice, you may find it helpful to change the name of each of them in their banner at the top.

To set a title for a terminal window:

1. In the terminal window's menu, select **Terminal** and then **Set Title**. A Set Title dialog box is displayed.
2. In the Title box, enter the window number.
3. Click **OK**.

Assumptions

You are connected to the compute node as the `oracle` user.

Tasks

Window 1: Create a Directory for PDB3

1. Open a new terminal window and connect to the compute node as the `oracle` user. This terminal window will be called Window 1 throughout the practice.

```
$sudo su - oracle
```

2. Source the `oraenv` script.

```
[oracle@MYDBCS ~]$ . oraenv
ORACLE_SID = [ORCL] ?
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@MYDBCS ~]$
```

3. Create a directory for the new PDB, named `PDB3`, under the CDB file location. You'll first determine the correct location and then you'll create the directory.

   a. Log in to SQL*Plus as the `SYS` user with the `SYSDBA` privilege.

```
[oracle@MYDBCS ~]$ sqlplus / as sysdba
…
SQL>
```

   b. Query `V$DATAFILE` to determine the location of the root container (CDB) datafiles.

```
SQL> select name from v$datafile;
```

```
NAME
--------------------------------------------------------------------------------
/u02/app/oracle/oradata/ORCL/system01.dbf
/u02/app/oracle/oradata/ORCL/sysaux01.dbf
/u02/app/oracle/oradata/ORCL/undotbs01.dbf
/u02/app/oracle/oradata/ORCL/pdbseed/system01.dbf
/u02/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf
/u02/app/oracle/oradata/ORCL/users01.dbf
/u02/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf
/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf
/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf
/u02/app/oracle/oradata/ORCL/PDB1/PDB1_users01.dbf
/u02/app/oracle/oradata/ORCL/659622D851BF1AE2E0533620C40AD6D5/da
tafile/o1_mf_users_fjv8c7l1_.dbf
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da
tafile/o1_mf_system_fk2t2tmq_.dbf
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da
tafile/o1_mf_sysaux_fk2t2tmv_.dbf
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da
tafile/o1_mf_undotbs1_fk2t2tmy_.dbf


16 rows selected.


SQL>
```

c. Use the `host` command to exit to the operating system.

```
SQL> host
[oracle@MYDBCS ~]$
```

d. Create a new directory named `PDB3` in the location you determined in the previous step.

```
$ mkdir /u02/app/oracle/oradata/ORCL/PDB3
[oracle@MYDBCS ~]$
```

e. Enter `exit` to return to SQL*Plus.

```
[oracle@MYDBCS ~]$ exit
exit


SQL>
```

Window 1: Verify that the `HR` Account in `PDB1` is Unlocked

1. Switch to `PDB1`.

```
SQL> ALTER SESSION SET CONTAINER = PDB1;
```

```
Session altered.

SQL>
```

2. Verify that the HR user account is unlocked, by checking for a status of OPEN.

```
SQL> col username format a15
SQL> SELECT username, account_status FROM dba_users where
username = 'HR';

USERNAME        ACCOUNT_STATUS
--------------- --------------------------------
HR              OPEN

SQL>
```

3. Switch back to the root container (CDB$ROOT).

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

Session altered.

SQL>
```

Window 2: Start a Transaction in PDB1

Start a transaction in PDB1 to determine what happens during the cloning operation when there is an uncommitted transaction.

1. Open a new terminal window and connect to the compute node as the oracle user. This window will be referred to as Window 2 throughout the practice.

```
[oracle@edvm ~]$ cd ~/.ssh
[oracle@edvm .ssh]$ ssh -i your_private_key_file
oracle@your_compute_node_IP_Address
[oracle@MYDBCS ~]$
```

2. Source the oraenv script.

```
[oracle@MYDBCS ~]$ . oraenv
ORACLE_SID = [ORCL] ?
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@MYDBCS ~]$
```

3. Start SQL*Plus and connect to PDB1 as the HR user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
[oracle@MYDBCS ~]$ sqlplus hr/password@PDB1
…
SQL>
```

4. Issue a query against the EMPLOYEES table to display the salary for employee ID 100.

```
SQL> SELECT salary FROM hr.employees WHERE employee_id = 100;

    SALARY
----------
     24000


SQL>
```

5. Update the EMPLOYEES table so that the employee salaries are increased by 10%. You will commit this transaction after you clone PDB1.

```
SQL> UPDATE employees SET salary=salary * 1.1;

107 rows updated.

SQL>
```

6. Display the salary for employee ID 100 again. The salary changed from 24000 to 26400. Do not commit this transaction at this time.

```
SQL> SELECT salary FROM hr.employees WHERE employee_id = 100;

    SALARY
----------
     26400


SQL>
```

Window 1: Clone PDB1 as PDB3

In this section, you clone PDB1 as PDB3. PDB1 is currently open and in READ WRITE mode. There is also a pending transaction in PDB1. Cloning PDB1 while it is open and has a pending transaction is referred to as *hot cloning*.

1. In Window 1, create a clone named PDB3 from PDB1 by using the CREATE PLUGGABLE DATABASE statement.

   In Oracle Database Cloud Service, PDBs are encrypted, so you must include the KEYSTORE IDENTIFIED BY clause. The keystore password is the administrative password you entered when you created the database deployment.

   $ mkdir /u02/oradata

   Sql> alter system set db_create_file_dest = '/u02/oradata';

```
SQL> CREATE PLUGGABLE DATABASE PDB3 FROM PDB1
  2  '

Pluggable database created.

SQL>
```

2. Verify that the open mode for PDB1 is READ WRITE and the open mode for PDB3 is MOUNTED by querying the V$PDBS view.

```
SQL> COLUMN con_id FORMAT 999
SQL> COLUMN name FORMAT A10
SQL> SELECT con_id, name, open_mode FROM v$pdbs;

CON_ID NAME       OPEN_MODE
------ ---------- ----------
     2 PDB$SEED   READ ONLY
     4 PDB2       READ WRITE
     3 PDB1       READ WRITE
     5 PDB3       MOUNTED

SQL>
```

3. Open PDB3 so that its open mode is READ WRITE.

```
SQL> ALTER PLUGGABLE DATABASE PDB3 OPEN;

Pluggable database altered.

SQL>
```

4. Verify that the open mode for both PDB1 and PDB3 is READ WRITE.

```
SQL> SHOW PDBS

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ  ONLY  NO
         4 PDB2                           READ WRITE NO
         3 PDB1                           READ WRITE NO
         5 PDB3                           READ WRITE NO
SQL>
```

Window 2: Commit the Transaction

1. In Window 2, commit the pending transaction in PDB1.

```
SQL> COMMIT;

Commit complete.

SQL>
```

2. Display the new salary for employee ID 100. The salary is 26400.

```
SQL> SELECT salary FROM hr.employees WHERE employee_id = 100;
```

```
    SALARY
----------
     26400

SQL>
```

3. Question: Do you think the salaries are updated in the clone (PDB3)?

   Answer: Continue to the next section to find out.

Window 1: Explore PDB3

1. In Window 1, switch to `PDB3` by using the `ALTER SESSION` command. This command connects you to `PDB3` as the `SYS` user.

```
SQL> ALTER SESSION SET container = PDB3;

Session altered.

SQL>
```

2. What is the salary of employee ID 100 in `PDB3`?

```
SQL> SELECT salary FROM hr.employees WHERE employee_id = 100;


    SALARY
----------
     24000


SQL>
```

3. Question: The original salary was 24000. Earlier in Window 2, you updated the salary to 26400 in `PDB1`. Why isn't the salary showing as 26400 in `PDB3`?

   Answer: The salary was not increased because you entered the `COMMIT` statement after the clone operation had completed.

4. Display the service name for `PDB3` by querying the `V$SERVICES` view.

```
SQL> COLUMN name FORMAT A20
SQL> SELECT name FROM v$services;

NAME
--------------------
PDB3

SQL>
```

5. List the data files for `PDB3` and their respective tablespaces by querying the `DBA_DATA_FILES` view. The results are formatted for easier viewing.

```
SQL> SELECT file_name, tablespace_name FROM dba_data_files;

FILE_NAME
----------------------------------------------------------------
TABLESPACE_NAME
------------------------------
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_system_fcbkbm0d_.dbf
SYSTEM
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_sysaux_fcbkbm1o_.dbf
SYSAUX
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_undotbs1_fcbkbm1s_.dbf
UNDOTBS1
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_users_fcbkbm1z_.dbf
USERS


SQL>
```

6. Question: Do you notice a difference between the data file names in the previous step compared with the names when you created a PDB from seed?

   Answer: In this case, Oracle Managed Files (OMF) names the data files for you because you used the `CREATE_FILE_DEST` clause, which only defines the directory for the data files. This clause comes from the initialization parameter `DB_CREATE_FILE_DEST`. If you use this parameter, then all your PDB data files will end up in the same directory; whereas using the `CREATE_FILE_DEST` clause enables you to specify distinct directories for each PDB.

7. List the temp file(s) for `PDB3` by querying the `DBA_TEMP_FILES` view. The query returns one temp file. The name of your temp file will be different from the one shown below.

```
SQL> SELECT file_name, tablespace_name FROM dba_temp_files;

FILE_NAME
----------------------------------------------------------------
TABLESPACE_NAME
------------------------------
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_temp_fcbkbm1x_.dbf
TEMP
```

```
SQL>
```

8. List the local users for `PDB3` by querying the `DBA_USERS` view.

```
SQL> col username format a30
SQL> SELECT DISTINCT username FROM dba_users WHERE common='NO'
ORDER BY username;

USERNAME
------------------------------
APEX_050100
APEX_INSTANCE_ADMIN_USER
APEX_LISTENER
APEX_PUBLIC_USER
APEX_REST_PUBLIC_USER
DHAMBY
FLOWS_FILES
HR
INVENTORY
JGOODMAN
PDB1_ADMIN1
PDBADMIN
RPANDYA
SCOTT

14 rows selected.

SQL>
```

9. List the common users for `PDB3` by querying the `DBA_USERS` view.

```
SQL> SELECT DISTINCT username FROM dba_users WHERE common='YES'
ORDER BY username;

USERNAME
------------------------------
ANONYMOUS
APPQOSSYS
AUDSYS
C##CDB_ADMIN1
…
SYSRAC
SYSTEM
WMSYS
XDB
XS$NULL
```

```

39 rows selected.

SQL>
```

10. Exit SQL*Plus.

```
SQL> EXIT
…
[oracle@MYDBCS ~]$
```

Window 2: Return Salary Values to Their Original Values

1. Return to Window 2. You should be logged in to `PDB1` as the `HR` user.
2. Return the `SALARY` column values in the `EMPLOYEES` table back to their original values.

```
SQL> UPDATE employees SET salary=salary / 1.1;

107 rows updated.

SQL>
```

3. Commit the transaction.

```
SQL> COMMIT;
Commit complete.

SQL>
```

4. Display the salary for employee ID 100. The result is 24000.

```
SQL> SELECT salary FROM hr.employees WHERE employee_id = 100;

    SALARY
----------
     24000


SQL>
```

5. Exit SQL*Plus, close the connection to the compute node, and close the terminal window.

```
SQL> EXIT
…
[oracle@MYDBCS ~]$
```

Lab 13: Unplugging and Plugging in a PDB

Overview

In this practice, you unplug `PDB3` from the `ORCL` CDB and plug it back into `ORCL` CDB. You give the PDB a new name (`HRPDB`) when you plug it back in.

Assumptions

You are logged in to the compute node as the `oracle` user.

You completed Practice 10-2 Hot Cloning a PDB.

Tasks

1. Unplug `PDB3` from the `ORCL` CDB.
   a. Start SQL*Plus and log in to `ORCL` with the `SYSDBA` privilege.

```
$ sqlplus / as sysdba
…
SQL>
```

   b. Close `PDB3`.

   PDBs must be closed before you can unplug them and drop them. If `PDB3` is already closed, you will receive an error message.

```
SQL> ALTER PLUGGABLE DATABASE PDB3 CLOSE IMMEDIATE;

Pluggable database altered.


SQL>
```

   c. Unplug PDB3 into an XML file named `/u02/app/oracle/oradata/PDB3.xml`.

   The unplugging operation makes changes in the PDB data files to record that the PDB was properly and successfully unplugged.

   Because the PDB is encrypted, you must include the `ENCRYPT USING` *`transport_secret`* clause. If you do not include the clause, you will receive an `ORA-46680: master keys of the container database must be exported` error. Supply a value of `TransPDB3` for *`transport_secret`* for the course practice.

   Because the PDB is still part of the CDB, you can back it up in Oracle Recovery Manager (Oracle RMAN). This backup provides a convenient way to archive the unplugged PDB. After backing it up, you can then remove it from the CDB catalog. However, you must preserve the data files for any subsequent plugging operations.

```
SQL> ALTER PLUGGABLE DATABASE PDB3
  2  UNPLUG INTO '/u02/app/oracle/oradata/PDB3.xml';


Pluggable database altered.
```

```

SQL>
```

d. Check the status of `PDB3` by querying `CDB_PDBS`.

```
SQL> col PDB_NAME format a10
SQL> SELECT pdb_name, status FROM cdb_pdbs;

PDB_NAME   STATUS
---------- ----------------------
PDB2       NORMAL
PDB$SEED   NORMAL
PDB1       NORMAL
PDB3       UNPLUGGED

SQL>
```

e. Drop `PDB3` while it is closed, but keep its datafiles so you can plug the PDB back in.

```
SQL> DROP PLUGGABLE DATABASE PDB3 KEEP DATAFILES;

Pluggable database dropped.

SQL>
```

f. Verify the status of the unplugged `PDB3` by querying the `CDB_PDBS` view. Note that `PDB3` is not included.

```
SQL> SELECT pdb_name, status FROM cdb_pdbs;

PDB_NAME   STATUS
---------- ----------------------
PDB2       NORMAL
PDB$SEED   NORMAL
PDB1       NORMAL

SQL>
```

2. Plug `PDB3` back into the `ORCL` CDB. The method would be similar if you were to plug the PDB into a different CDB.

   a. Make sure that `PDB3` is compatible with the `ORCL` CDB. Execution of the following PL/SQL block raises an error if it is not compatible.

   Tip: Enter each line, followed by a return, and the whole procedure will run after you close with a slash.

```
SQL> set serveroutput on
SQL> DECLARE
  2   compatible BOOLEAN := FALSE;
  3   BEGIN
4       compatible := DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
      pdb_descr_file =>
'/u02/app/oracle/oradata/PDB3.xml');
  5   if compatible then
  6   DBMS_OUTPUT.PUT_LINE('PDB3 is compatible');
  7   else DBMS_OUTPUT.PUT_LINE('PDB3 is not compatible');
  8   end if;
  9   END;
  10   /

PDB3 is compatible

PL/SQL procedure successfully completed.

SQL>
```

b. Plug `PDB3` back into the `ORCL` CDB by using the `NOCOPY` method. Rename the plugged-in PDB as `HRPDB`.

Because the PDB is encrypted, you must include the `IDENTIFIED BY` *`password`* clause and include the keystore password. In Oracle Database Cloud Service, the keystore password is the value you supplied when you created the database deployment.

You must also include the `DECRYPT USING` *`transport_secret`* clause. The value for *`transport_secret`* is the same value you specified when you unplugged the PDB. For the course practice, the value is `TransPDB3`.

The original data files of the unplugged PDB now belong to the new plugged-in PDB.

```
SQL> CREATE PLUGGABLE DATABASE HRPDB
  2  USING '/u02/app/oracle/oradata/PDB3.xml'
  3  NOCOPY TEMPFILE REUSE
  4  KEYSTORE IDENTIFIED BY password
  4  DECRYPT USING TransPDB3;

Pluggable database created.

SQL>
```

3. Examine the plugged-in PDB.

   a. List all the containers in your CDB by querying the `V$CONTAINERS` view. The results list five containers—the root container (`CDB$ROOT`), the seed PDB (`PDB$SEED`), `PDB1`, `PDB2`, and `HRPDB`.

```
SQL> COLUMN name FORMAT A8
SQL> SELECT name, con_id FROM v$containers ORDER BY con_id;
```

```
NAME          CON_ID
-------------
CDB$ROOT           1
PDB$SEED           2
PDB1               3
PDB2               4
HRPDB              5

SQL>
```

b. Show the status of `HRPDB` by querying the `CDB_PDBS` view.

```
SQL> SELECT pdb_name, status FROM cdb_pdbs WHERE
pdb_name='HRPDB';

PDB_NAME    STATUS
----------- ----------
HRPDB       NEW

SQL>
```

c. Show the open mode of `HRPDB` by querying the `V$PDBS` view.

```
SQL> SELECT open_mode FROM v$pdbs WHERE name='HRPDB';

OPEN_MODE
-------------
MOUNTED

SQL>
```

d. List the data files of `HRPDB` by querying the `V$DATAFILE` view. Recall that the `HRPDB` container's ID is 5. Your paths and data file names will differ from those shown below.

```
SQL> COLUMN name FORMAT A50
SQL> SELECT name FROM v$datafile WHERE con_id=5;
NAME
---------------------------------------------------------------------------------
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_system_fcbkbm0d_.dbf
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_sysaux_fcbkbm1o_.dbf
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_undotbs1_fcbkbm1s_.dbf
/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8
60AAA15/datafile/o1_mf_users_fcbkbm1z_.dbf
```

```
SQL>
```

4. Open and connect to HRPDB.

   a. Open HRPDB.

```
SQL> ALTER PLUGGABLE DATABASE HRPDB open;

Pluggable database altered.
SQL>
```

   b. Query V$SERVICES.

```
SQL> SELECT name FROM v$services ORDER BY name;


NAME
--------------------------------------------------------------------------------
ORCL.588436052.oraclecloud.internal
ORCL.588436052.oraclecloud.internalXDB
SYS$BACKGROUND
SYS$USERS
PDB1
PDB2
hrpdb

7 rows selected.

SQL>
```

   c. Connect to HRPDB as the SYS user with the SYSDBA privilege. Recall that you need to append the values following ORCL as shown in this example.

```
SQL> CONNECT
SYS/password@localhost:1521/hrpdb.588436052.oraclecloud.internal
AS SYSDBA
Connected.
SQL>
```

5. Verify the current container name is HRPDB.

```
SQL> SHOW con_name
CON_NAME
------------------------------
HRPDB
SQL>
```

6. Exit from SQL*Plus.

```
SQL> exit
…
[oracle@MYDBCS ~]$
```

Lab 14: Dropping a PDB

Overview

In this practice, you drop the HRPDB PDB.

Assumptions

You are logged in to the compute node as the oracle user.

You completed the following practices in this lesson:

- Creating a PDB from Seed
- Hot Cloning a PDB

Tasks

1. Start SQL*Plus and connect to the root container as the SYS user with the SYSDBA privilege.

```
[oracle@MYDBCS ~]$ sqlplus / as sysdba
..
SQL>
```

2. List the PDBs in ORCL. The results show four PDBs: PDB$SEED, PDB1, PDB2, and HRPDB.

```
SQL> SHOW PDBS

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ WRITE NO
         4 PDB2                           READ WRITE NO
         5 HRPDB                          READ WRITE NO
SQL>
```

3. Close HRPDB.

```
SQL> ALTER PLUGGABLE DATABASE HRPDB CLOSE;

Pluggable database altered.

SQL>
```

4. Drop HRPDB, including its data files, by using the DROP PLUGGABLE DATABASE command.

```
SQL> DROP PLUGGABLE DATABASE HRPDB INCLUDING DATAFILES;

Pluggable database dropped.

SQL>
```

5. List the PDBs in ORCL. The results show three PDBs: PDB$SEED, PDB1, and PDB2.

```
SQL> SHOW PDBS

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ  ONLY  NO
         3 PDB1                           READ WRITE NO
         4 PDB2                           READ WRITE NO
SQL>
```

6. Exit SQL*Plus and close the terminal window.

```
SQL> EXIT
…
[oracle@MYDBCS ~]$ exit
```

Lab 15: Viewing Tablespace Information

Overview

In this practice, you use SQL*Plus to query various views to learn about tablespace content in `PDB1`. You also view tablespace information with Enterprise Manager Database Express (EM Express).

Assumptions

You are logged in as the `oracle` user.

Tasks

1. Open a new terminal window and connect to the compute node as the `oracle` user.

```
$ sudo su - oracle
```

2. Source the `oraenv` script.

```
[oracle@MYDBCS ~]$ . oraenv
ORACLE_SID = [ORCL] ?
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@MYDBCS ~]$
```

3. Start SQL*Plus and connect to `PDB1` as the `PDBADMIN` user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
[oracle@MYDBCS ~]$ sqlplus PDBADMIN/password@PDB1
…
SQL>
```

4. List the columns in the `DBA_TABLESPACES` view by using the `DESCRIBE` command.

```
SQL> DESCRIBE dba_tablespaces
 Name                                       Null?    Type
 ----------------------------------------- -------- ------------
 TABLESPACE_NAME                            NOT NULL VARCHAR2(30)
 BLOCK_SIZE                                 NOT NULL NUMBER
 INITIAL_EXTENT                                      NUMBER
 NEXT_EXTENT                                         NUMBER
 MIN_EXTENTS                                NOT NULL NUMBER
…
 DEF_CELLMEMORY                                      VARCHAR2(14)
 DEF_INMEMORY_SERVICE                                VARCHAR2(12)
 DEF_INMEMORY_SERVICE_NAME                           VARCHAR2(1000)
 LOST_WRITE_PROTECT                                  VARCHAR2(7)
 CHUNK_TABLESPACE                                   VARCHAR2(1)

SQL>
```

5. List the tablespaces in `PDB1`.

```
SQL> SELECT DISTINCT tablespace_name FROM dba_tablespaces ORDER
BY tablespace_name;

TABLESPACE_NAME

------------------------------
SYSAUX

SYSTEM
TEMP
UNDOTBS1
USERS


SQL>
```

6. Find out which tablespace contains the `HR` schema by querying the `ALL_TABLES` view.

```
SQL> SELECT DISTINCT tablespace_name FROM all_tables WHERE
owner='HR';


TABLESPACE_NAME
------------------------------

USERS


SQL>
```

7. Query the `STATUS`, `CONTENTS`, `LOGGING`, `PLUGGED_IN`, `BIGFILE`, `EXTENT_MANAGEMENT`, and `ALLOCATION_TYPE` columns in the `DBA_TABLESPACES` view for the `SYSAUX` tablespace.

```
SQL> SELECT status, contents, logging, plugged_in, bigfile,
extent_management, allocation_type FROM dba_tablespaces where
tablespace_name='SYSAUX';

STATUS    CONTENTS              LOGGING  PLU  BIG  EXTENT_MAN  ALLOCA

--------- ------------------ --------- --- --- ---------- ---------
ONLINE PERMANENT LOGGING NO NO LOCAL SYSTEM  SQL>
```

- `STATUS` shows the value `ONLINE`, indicating the tablespace is available to users.
- `CONTENTS` indicates the `PERMANENT` tablespace type.

- LOGGING shows the value LOGGING, indicating that certain DML operations are logged in the redo log file.
- PLUGGED_IN shows the value NO, indicating that the tablespace is not plugged in.
- BIGFILE shows the value NO, indicating that the tablespace is a smallfile tablespace.
- EXTENT_MANAGEMENT shows the value LOCAL, indicating that the tablespace is locally managed (not dictionary managed).
- ALLOCATION_TYPE shows the value SYSTEM, indicating that the extents of the tablespace are managed by the system, and you cannot specify an extent size.

8. List the columns in the V$TABLESPACE view by using the DESCRIBE command. This view displays tablespace information from the control file.

```
SQL> DESCRIBE v$tablespace
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 TS#                                                NUMBER
 NAME                                               VARCHAR2(30)
 INCLUDED_IN_DATABASE_BACKUP                        VARCHAR2(3)
 BIGFILE                                            VARCHAR2(3)
 FLASHBACK_ON                                       VARCHAR2(3)
 ENCRYPT_IN_BACKUP                                  VARCHAR2(3)
 CON_ID                                             NUMBER

SQL>
```

9. Query the V$TABLESPACE view for the SYSAUX tablespace.

```
SQL> SELECT * FROM v$tablespace WHERE name='SYSAUX';

       TS# NAME                           INC BIG FLA ENC CON_ID
---------- ------------------------------ --- --- --- --- ------
         1 SYSAUX                         YES NO  YES          3

SQL>
```

- INCLUDED_IN_DATABASE_BACKUP contains the value YES, indicating that the tablespace is included in full database backups by using the BACKUP DATABASE RMAN command.
- BIGFILE contains the value NO, indicating that the tablespace is a smallfile tablespace.
- FLASHBACK_ON contains the value YES, indicating that the tablespace participates in FLASHBACK DATABASE operations.
- ENCRYPT_IN_BACKUP contains the value null, indicating that encryption is neither explicitly turned on nor off at the tablespace level.

- CON_ID indicates the container to which the data pertains. In this case, PDB1 is container ID 3.

10. List all the tables in the USERS tablespace owned by the HR account.

```
SQL> SELECT table_name FROM all_tables WHERE
tablespace_name='USERS' and owner='HR';

TABLE_NAME
--------------------------------------------------------------------------------
REGIONS
LOCATIONS
DEPARTMENTS
JOBS
EMPLOYEES
JOB_HISTORY

6 rows selected.

SQL>
```

11. List all the indexes in the USERS tablespace owned by the HR account.

```
SQL> SELECT index_name FROM all_indexes WHERE
tablespace_name='USERS' AND owner='HR' ORDER BY index_name;

INDEX_NAME
--------------------------------------------------------------------------------
COUNTRY_C_ID_PK
DEPT_ID_PK
DEPT_LOCATION_IX
EMP_DEPARTMENT_IX
EMP_EMAIL_UK
EMP_EMP_ID_PK
EMP_JOB_IX
EMP_MANAGER_IX
EMP_NAME_IX
JHIST_DEPARTMENT_IX
JHIST_EMPLOYEE_IX
JHIST_EMP_ID_ST_DATE_PK
JHIST_JOB_IX
JOB_ID_PK
LOC_CITY_IX
LOC_COUNTRY_IX
LOC_ID_PK
LOC_STATE_PROVINCE_IX
```

```
REG_ID_PK

19 rows selected.

SQL>
```

12. List the columns in the `DBA_DATA_FILES` view by using the `DESCRIBE` command. You can query this view to learn about the data files contained in a tablespace.

```
SQL> DESCRIBE dba_data_files
Name                                          Null?    Type
 ----------------------------------------- -------- ------------
 FILE_NAME                                         VARCHAR2(513)
 FILE_ID                                            NUMBER
 TABLESPACE_NAME                                    VARCHAR2(30)
 BYTES                                              NUMBER
 BLOCKS                                             NUMBER
 STATUS                                             VARCHAR2(9)
 RELATIVE_FNO                                       NUMBER
 AUTOEXTENSIBLE                                     VARCHAR2(3)
 MAXBYTES                                           NUMBER
 MAXBLOCKS                                          NUMBER
 INCREMENT_BY                                       NUMBER
 USER_BYTES                                         NUMBER
 USER_BLOCKS                                        NUMBER
 ONLINE_STATUS                                      VARCHAR2(7)
 LOST_WRITE_PROTECT                                 VARCHAR2(7)

SQL>
```

13. List data file information for the `SYSAUX` tablespace by querying various columns in the `DBA_DATA_FILES` view.

```
SQL> COLUMN file_name FORMAT A50
SQL> SELECT file_name, autoextensible, bytes, maxbytes,
user_bytes FROM dba_data_files WHERE tablespace_name='SYSAUX';

FILE_NAME                                          AUT
BYTES   MAXBYTES  USER_BYTES
-------------------------------------------------- --- ----------
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf YES  828375040
3.4360E+10  827326464



SQL>
```

The results show the following:

- AUTOEXTENSIBLE contains the value YES, indicating that the auto extend feature is enabled for a data file. The tablespace size can increase without you having to take any action.
- BYTES is the size of the file in bytes.
- MAXBYTES is the maximum file size allowed.
- USER_BYTES is the size of the file available for user data.

14. Find out how many segments are there in the SYSAUX tablespace by querying the DBA_SEGMENTS view.

```
SQL> SELECT count(segment_name) FROM dba_segments WHERE
tablespace_name='SYSAUX';


COUNT(SEGMENT_NAME)
-------------------
               2356


SQL>
```

15. Find out which index in the SYSAUX tablespace takes up the most space by querying the DBA_SEGMENTS view. The results indicate that the I_WRI$_OPTSTAT_H_OBJ#_ICOL#_ST index takes up the most space.

```
SQL> col segment_name format a35
SQL> SELECT *
  2  FROM (SELECT segment_name, segment_type, bytes
  3        FROM dba_segments
  4        WHERE segment_type = 'INDEX' AND
  5        tablespace_name ='SYSAUX'
  6        ORDER BY bytes desc)
  7  WHERE rownum < 2;

SEGMENT_NAME                        SEGMENT_TYPE            BYTES
----------------------------------- ------------------ ----------
I_WRI$_OPTSTAT_H_OBJ#_ICOL#_ST      INDEX                42991616


SQL>
```

16. Exit SQL*Plus.

```
SQL> exit
..
[oracle@MYDBCS ~]$
```

Lab 16: Creating a Tablespace

Overview

In this practice, you create and populate a tablespace named `INVENTORY`.

Assumptions

You are logged in to the compute node as the `oracle` user.

Tasks

Use SQL*Plus to Create the INVENTORY Tablespace and Table X

As the `PDBADMIN` user in SQL*Plus, execute the `CreateINVENTORYTablespace.sql` script to create the `INVENTORY` tablespace. Next, execute a script named `CreateTableX.sql` to create and populate a table called `X` in the `INVENTORY` tablespace. At first, you will get an error trying to populate the table. In the next section, you correct the problem.

1. Start SQL*Plus and connect to `PDB1` as the `PDBADMIN` user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
[oracle@MYDBCS ~]$ sqlplus PDBADMIN/password@PDB1
…
SQL>
```

2. Execute the `CreateINVENTORYTablespace.sql` script.

```
SQL> set echo on
SQL> @/home/oracle/labs/CreateINVENTORYTablespace.sql

SQL> CREATE SMALLFILE TABLESPACE INVENTORY
  2
           DATAFIL
  E  3
'/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF' SIZE 5242880
  4        DEFAULT NOCOMPRESS
  5        ONLINE
  6        SEGMENT SPACE MANAGEMENT AUTO
  7        EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

Tablespace

created.  SQL>
```

Lab 17: Verifying that the Control File is Multiplexed

Overview

In this practice, you verify that the control file is multiplexed.

A control file is a small binary file that describes the structure of the database. It must be available for writing by the Oracle server whenever the database is mounted or opened. Without this file, the database cannot be mounted, and recovery or re-creation of the control file is required. Your database should have a minimum of two control files on different storage devices to minimize the impact of a loss of one control file. The loss of a single control file causes the instance to fail because all control files must be available at all times. However, recovery can be a simple matter of copying one of the other control files. The loss of all control files is slightly more difficult to recover from, but is not usually catastrophic.

Assumptions

You are logged in as the `oracle` user.

Tasks

1. Open a new terminal window and connect to the compute node as the `oracle` user.

```
$sudo su - oracle
```

2. Source the `oraenv` script.

```
[oracle@MYDBCS ~]$ . oraenv
ORACLE_SID = [ORCL] ?
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@MYDBCS ~]$
```

3. Start SQL*Plus and connect to the CDB root as the `SYS` user with the `SYSDBA` privilege.

```
$ sqlplus / AS SYSDBA
…
SQL>
```

4. Find out how many control files exist in the database. The query returns the names of two control files (`control01.ctl` and `control02.ctl`), which verifies that the control files are multiplexed.

```
SQL> SELECT name FROM v$controlfile;

NAME
--------------------------------------------------------------------------------
/u02/app/oracle/oradata/ORCL/control01.ctl
/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl
```

```
SQL>
```

When the CDB was created, DBCA created two control files. When you use the `CREATE DATABASE` command in SQL*Plus to create a database, you configure the `CONTROL_FILES` parameter to generate two control files and set their names.

5. View the `CONTROL_FILES` parameter. Notice that the paths to the control files are stored in this parameter. The results below are formatted for easier viewing.

```
SQL> SHOW PARAMETER control_files

NAME                                                           TYPE
-----------------------------------------------------------    ---------
VALUE
------------------------------------------------------------------
control_files                                                  string
/u02/app/oracle/oradata/ORCL/control01.ctl,
/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl
SQL>
```

6. Create a parameter file (PFILE) from the server parameter file (SPFILE).

```
SQL> CREATE PFILE FROM SPFILE;

File created.

SQL>
```

7. Shut down the database instance in `IMMEDIATE` mode.

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

8. Exit SQL*Plus.

```
SQL> EXIT
```

9. Create a directory for the new control file.

```
[oracle@MYDBCS ~]$ mkdir -p
/u01/app/oracle/controlfiles_dir/ORCL
[oracle@MYDBCS ~]$
```

10. Before you edit your PFILE, make a backup copy of it.

```
[oracle@MYDBCS ~]$ cp $ORACLE_HOME/dbs/initORCL.ora
$ORACLE_HOME/dbs/backup_initORCL.ora
[oracle@MYDBCS ~]$
```

11. Copy one of the control files to the directory you created in a previous step (`/u01/app/oracle/controlfiles_dir/ORCL`) and name the file as `control03.ctl`.

```
[oracle@MYDBCS ~]$ cp /u02/app/oracle/oradata/ORCL/control01.ctl
/u01/app/oracle/controlfiles_dir/ORCL/control03.ctl
[oracle@MYDBCS ~]$
```

12. Open the PFILE (`initORCL.ora`) in the `vi` editor and add the name of the new control file to the end of the list of control files. Include the path. Be certain not to enter spaces between the single quotes and commas in the `control_files=` line. Be certain that this line is one continuous line, without line breaks. Save and close the file (:wq).

```
$ vi $ORACLE_HOME/dbs/initORCL.ora
...
*.control_files='/u02/app/oracle/oradata/ORCL/control01.ctl',
'/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl',
'/u01/app/oracle/controlfiles_dir/ORCL/control03.ctl'
...
$
```

13. Start SQL*Plus and connect to the root container as the `SYS` user with the `SYSDBA` privilege. You are connected to an idle instance.

```
$ sqlplus / AS SYSDBA
…
SQL>
```

14. Start the database instance.

```
SQL> STARTUP
ORACLE instance started.

Total System Global Area 2768239832 bytes
Fixed Size                  8899800 bytes
Variable Size             704643072 bytes
Database Buffers 1979711488 bytes   Redo
Buffers                    74985472 bytes
Database mounted.
Database opened.
SQL>
```

15. View the `CONTROL_FILES` parameter again.

```
SQL> SHOW PARAMETER control_files

NAME                                 TYPE
------------------------------------ -----------
VALUE
------------------------------
control_files                        string
/u02/app/oracle/oradata/ORCL/control01.ctl,
/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl
```

```
SQL>
```

16. Question: Why does the `CONTROL_FILES` parameter still show only two control files?

    Answer: By default, the database instance starts up with the SPFILE. If an SPFILE does not exist, then the instance starts up with a PFILE. In this case, both an SPFILE and PFILE are present, so the SPFILE takes precedence. You configured the PFILE, not the SPFILE. The SPFILE still contains only two references.

17. Re-create the third control file because the current version is no longer an exact copy of the others.

    a. Shut down the database instance with the `IMMEDIATE` option.

    ```
    SQL> SHUTDOWN IMMEDIATE
    Database closed.
    Database dismounted.
    ORACLE instance shut down.
    SQL>
    ```

    b. Exit SQL*Plus.

    ```
    SQL> EXIT
    ```

    c. Use the `cp` command to re-create `control03.ctl`.

    ```
    [oracle@MYDBCS ~]$ cp /u02/app/oracle/oradata/ORCL/control01.ctl
    /u01/app/oracle/controlfiles_dir/ORCL/control03.ctl
    [oracle@MYDBCS ~]$
    ```

18. Re-create the SPFILE from the updated PFILE.

    a. Start SQL*Plus and connect to the CDB root as the `SYS` user with the `SYSDBA` privilege. You are connected to an idle instance.

    ```
    [oracle@MYDBCS ~]$ sqlplus / AS SYSDBA
    …
    SQL>
    ```

    b. Create the SPFILE.

    ```
    SQL> CREATE SPFILE FROM PFILE;

    File created.

    SQL>
    ```

19. Start the database instance.

    ```
    SQL> STARTUP
    ORACLE instance started.

    Total System Global Area 2768239832 bytes
    Fixed Size                  8899800 bytes
    Variable Size             704643072 bytes
    Database Buffers         1979711488 bytes
    ```

```
Redo Buffers                 74985472 bytes
Database mounted.
Database opened.
SQL>
```

20. View the `CONTROL_FILES` parameter again. The third control file is now included in the list, which indicates that the SPFILE is configured properly. The results below are formatted for easier viewing.

```
SQL> SHOW PARAMETER control_files

NAME                                 TYPE
------------------------------------ -----------
VALUE
------------------------------
control_files                        string
/u02/app/oracle/oradata/ORCL/control01.ctl,
/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl,
/u01/app/oracle/controlfiles_dir/ORCL/control03.ctl
SQL>
```

21. Query the `V$CONTROLFILE` view to confirm the number of control files. The result indicates that three control files are defined.

```
SQL> SELECT name FROM v$controlfile;

NAME
-----------------------------------------------------------------
/u02/app/oracle/oradata/ORCL/control01.ctl
/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl
/u01/app/oracle/controlfiles_dir/ORCL/control03.ctl

SQL>
```

Lab 18: Configuring the Size of the Fast Recovery Area

Overview

In this practice, you review the fast recovery area (FRA) configuration and change its size to 12GB.

Assumptions

You are logged in to SQL*Plus from the previous practice.

Tasks

1. Question: How can you evaluate the space needed for the FRA?

   Answer: The amount of disk space to allocate for the FRA depends on the size and activity levels of your database. As a general rule, the larger the FRA, the more useful it is. Ideally, the FRA should be large enough for copies of your data and control files, as well as for flashback, online redo, and archived logs needed to recover the database with the backups kept based on the retention policy (covered in one of the next practices). In short, the FRA should be at least twice the size of the database so that it can hold one backup and several archived logs.

2. View the values of the `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` initialization parameters.

```
SQL> SHOW PARAMETER db_recovery_file_dest

NAME                                 TYPE
------------------------------------ ---------------------------
VALUE
------------------------------
db_recovery_file_dest                string
/u03/app/oracle/fast_recovery_area
db_recovery_file_dest_size           big integer
4G
SQL>
```

3. Question: Is the fast recovery area enabled?

   Answer: Yes. The `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` parameters values are not null, indicating that the fast recovery area is enabled.

4. Question: Which changes can you make to the fast recovery area?

   Answer: You can change the location and size for the fast recovery area.

5. Question: Does changing the size of the fast recovery area require the database to be restarted?

   Answer: No, a restart is not required for this change because the `DB_RECOVERY_FILE_DEST_SIZE` parameter is dynamic.

```
SQL> ALTER SYSTEM SET db_recovery_file_dest_size = 12G
SCOPE=both;

System altered.

SQL>
```

6. Change the size of the fast recovery area to `12GB` and set the scope to `BOTH`.

   Note: If the archived redo log file destination fills up or cannot be written to, the database will halt. You would then need to remove archived redo log files from the archived redo log file destination so that the database could resume operations. This activity is covered in one of the next practices.

7. View the `DB_RECOVERY_FILE_DEST_SIZE` initialization parameter again. The result verifies that the size has been set to `12GB`.

```
SQL> SHOW PARAMETER db_recovery_file_dest_size

NAME                                  TYPE         VALUE
------------------------------------ ----------- ------------------
db_recovery_file_dest_size           big integer  12G
SQL>
```

Lab 19: Verifying that the Redo Log File is Multiplexed

Overview

Ensure that there are at least two redo log members in each group. If you are using file system storage, then each member should be distributed on separate disks or controllers so that no single equipment failure impacts an entire log group. The loss of an entire current log group is one of the most serious media failures because it can result in data loss. The loss of a single member of a multi-member log group is trivial and does not affect database operation (other than causing an alert to be published in the alert log). One set of members should be stored in the FRA.

Assumptions

You are logged in to SQL*Plus from the previous practice.

Tasks

1. Query **`V$LOGFILE`** to determine the configuration (number of members) for each redo log group. The result shows that there are currently three log groups (1, 2, and 3) and only one member in each group.

```
SQL> SELECT group#, status, member FROM v$logfile;

 GROUP# STATUS   MEMBER
_______ _______  ________________________________________

      3          /u04/app/oracle/redo/redo03.log
      2          /u04/app/oracle/redo/redo02.log
      1          /u04/app/oracle/redo/redo01.log


SQL>
```

2. Question: Why is it recommended to have three groups when two would be sufficient?

   Answer: The Oracle Database server treats the online redo log groups as a circular buffer in which to store transaction information, filling one group and then moving on to the next. After all groups have been written to, the Oracle Database server begins overwriting information in the first log group. If the database is configured in `ARCHIVELOG` mode, the LGWR cannot overwrite data in the first log group if it has not been archived.

3. Question: Can multiplexing redo logs impact database performance?

   Answer: Multiplexing redo logs may heavily influence database performance because a commit cannot complete until the transaction information has been written to the logs by LGWR. You must place your redo log files on your fastest disks served by your fastest controllers. If possible, do not place any other database files on the same disks as your redo log files. Because only one group is written to at a given time, there is no performance impact in having members from several groups on the same disk.

4. Add another member to each redo log group. Name each member as redo*nn*b.log where *nn* represents the group number.

```
SQL> ALTER DATABASE ADD LOGFILE MEMBER
'/u03/app/oracle/fast_recovery_area/ORCL/redo01b.log' TO GROUP
1;

Database altered.

SQL> ALTER DATABASE ADD LOGFILE MEMBER
'/u03/app/oracle/fast_recovery_area/ORCL/redo02b.log' TO GROUP
2;

Database altered.

SQL> ALTER DATABASE ADD LOGFILE MEMBER
'/u03/app/oracle/fast_recovery_area/ORCL/redo03b.log' TO GROUP
3;

Database altered.

SQL>
```

5. Verify that the redo log files are now multiplexed. The query result shows that each group has two members, and therefore, the redo log files are multiplexed. Observe the INVALID status of the newly added redo log members. This status is expected because the new members have not yet been written to by LGWR. When a log switch occurs and the group containing the new member becomes CURRENT, the new member's status will change to null.

```
SQL> SELECT group#, status, member FROM v$logfile ORDER BY 1, 3;

 GROUP# STATUS  MEMBER
---------- --------- ------------------------------------------------------------
      1 INVALID
/u03/app/oracle/fast_recovery_area/ORCL/redo01b.log
      1         /u04/app/oracle/redo/redo01.log
      2 INVALID
/u03/app/oracle/fast_recovery_area/ORCL/redo02b.log
      2         /u04/app/oracle/redo/redo02.log
      3 INVALID
/u03/app/oracle/fast_recovery_area/ORCL/redo03b.log
      3         /u04/app/oracle/redo/redo03.log

6 rows selected.

SQL>
```

6. Switch the log files and observe the changes.

   a. Find out which log group is the current log group. In this example, the query result shows that group 3 is the current group. Your current group may be different.

```
SQL> SELECT group#, members, archived, status FROM v$log;

 GROUP#     MEMBERS ARC STATUS
-------- ---------- --- ----------------
      1           2 YES INACTIVE
      2           2 YES INACTIVE
      3           2 NO  CURRENT

SQL>
```

   b. Switch the log files three times.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL>
```

   c. Query the `V$LOGFILE` view again. Notice that the switch caused the new members' statuses to change to null.

```
SQL> SELECT group#, status, member FROM v$logfile ORDER BY 1, 3;

 GROUP# STATUS  MEMBER
------- ------- ----------------------------------------
      1         /u03/app/oracle/fast_recovery_area/ORCL/redo01b.l
                og
      1         /u04/app/oracle/redo/redo01.log
      2         /u03/app/oracle/fast_recovery_area/ORCL/redo02b.l
                og
      2         /u04/app/oracle/redo/redo02.log
      3         /u03/app/oracle/fast_recovery_area/ORCL/redo03b.l
                og
6 rows 3selected. /u04/app/oracle/redo/redo03.log

SQL>
```

d. Query the `V$LOG` view again to learn which log group is now the current group. In this example, the results show that the LGWR is writing to group 3. Your group may be different. Your statuses may be different too. An `INACTIVE` status means the log group is no longer needed for database instance recovery.

```
SQL> SELECT group#, members, archived, status FROM v$log;

  GROUP#     MEMBERS ARC STATUS
---------- ---------- --- ----------------
         1          2 YES INACTIVE
         2          2 YES INACTIVE
         3          2 NO  CURRENT

SQL>
```

e. Switch the log file.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL>
```

f. Query the `V$LOG` view again. The current group has changed to group 1, and the former current group's status is now `ACTIVE`. Your current group may be different. An `ACTIVE` status means that the log group is active, but it's not the current log group. It is needed for crash recovery. It may be in use for block recovery. It may or may not be archived.

```
SQL> SELECT group#, members, archived, status FROM v$log;

  GROUP#     MEMBERS ARC STATUS
---------- ---------- --- ----------------
         1          2 NO  CURRENT
         2          2 YES INACTIVE
         3          2 YES ACTIVE

SQL>
```

g. Switch the log file again.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL>
```

h. Query the `V$LOG` view again. The current group has changed again to group 2, and the status of both the other groups is now `ACTIVE`. Your current group may be different.

```
SQL> SELECT group#, members, archived, status FROM v$log;

 GROUP#    MEMBERS ARC STATUS
------- ---------- --- ----------------
      1          2 YES ACTIVE
      2          2 NO  CURRENT
      3          2 YES ACTIVE

SQL>
```

i. Question: Can the LGWR background process write to only one member of the `CURRENT` group in case the other members are missing or damaged?

   Answer: Yes, it can. As long as there is one member left in the `CURRENT` group, LGWR can work.

7. To save space in your course practice environment, drop the redo log file members you created in step 4.

   a. Determine which redo log group is current. You cannot drop a member of the current group.

```
SQL> SELECT group#, status FROM v$log;

    GROUP# STATUS
---------- ----------------------------
         1 INACTIVE
         2 CURRENT
         3 INACTIVE
SQL>
```

   b. Drop the member in the previous group and then perform a log switch. In this example, group 2 is current, so the command drops a member in group 1.

```
SQL> ALTER DATABASE DROP LOGFILE MEMBER
'/u03/app/oracle/fast_recovery_area/ORCL/redo01b.log';

Database altered.
SQL> alter system switch logfile;

System altered.

SQL>
```

   c. Drop the member in the next group and then perform a log switch.

```
SQL> ALTER DATABASE DROP LOGFILE MEMBER
'/u03/app/oracle/fast_recovery_area/ORCL/redo02b.log';

Database altered.
```

```
SQL> alter system switch logfile;

System altered.

SQL>
```

d. Drop the member in the final group and then perform a log switch.

```
SQL> ALTER DATABASE DROP LOGFILE MEMBER
'/u03/app/oracle/fast_recovery_area/ORCL/redo03b.log';

Database altered.
SQL> alter system switch logfile;

System altered.

SQL>
```

e. Verify that each group now has only one member.

```
SQL> SELECT group#, members, archived, status FROM v$log;

    GROUP#    MEMBERS ARC STATUS
---------- ---------- --- ----------------
         1          1 YES ACTIVE

         2          1 NO  CURRENT
         3          1 YES ACTIVE

SQL>
```

f. Exit from SQL*Plus.

```
SQL> exit
...
[oracle@MYDBCS ~]$
```

g. Remove the physical files from the operating system.

```
[oracle@MYDBCS ~]$ rm
/u03/app/oracle/fast_recovery_area/ORCL/redo*.log
```

h. Verify that the redo log files have been removed.

```
[oracle@MYDBCS ~]$ ls /u03/app/oracle/fast_recovery_area/ORCL
archivelog  autobackup  control02.ctl  flashback  onlinelog
```

Lab 20: Verifying that ARCHIVELOG Mode is Configured

Overview

In this practice, you verify that your database is in `ARCHIVELOG` mode so that redo logs are archived.

Assumptions

Tasks

1. Log in to SQL*Plus as the `SYS` user with the `SYSDBA` privilege.

```
[oracle@MYDBCS ~]$ sqlplus / AS SYSDBA
…
SQL
```

2. Issue the `ARCHIVE LOG LIST` command to verify that the database is in `ARCHIVELOG` mode. The result confirms that the database log mode is set to `ARCHIVELOG` mode.

```
SQL> archive log list
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     27
Next log sequence to archive   29
Current log sequence           29
SQL>
```

   In Oracle Database Cloud Service, the database is set to ARCHIVELOG mode by default. If you are creating your own Oracle database, you will need to place the database in ARCHIVELOG mode as described in this lesson.

3. Exit from SQL*Plus.

```
SQL> EXIT
…
[oracle@MYDBCS ~]$
```

4) Place the database in archivelog mode

5) $ mkdir /u01/fra

6) Sqlplus / as sysdba

```
sql> alter system set db_recovery_file_dest_size=10g;
 sql> alter system set db_recovery_file_dest=‘/u01/fra’
 sql> shutdown immediate
 sql> startup mount
 sql> alter database archivelog;
 sql> alter database open;
 sql> alter system switch logfile;
```

Lab 21: Verifying Automatic Backups of the Control File and SPFILE

Overview

In this practice, you use Recovery Manager (RMAN) to configure automatic backups of the control file and server parameter file (SPFILE) when a backup of the database is made and when there is a structural change to the database.

Assumptions

You are logged in to the compute node as the `oracle` user.

You completed the practices in Lesson 17.

Tasks

1. Start Recovery Manager and connect to the CDB root (target database) as the `SYS` user.

```
[oracle@MYDBCS ~]$ rman target /

Recovery Manager: Release 18.0.0.0.0 - Production on Wed Apr 4
20:42:53 2018
Version 18.1.0.0.0

Copyright (c) 1982, 2018, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: ORCL (DBID=2299035716)

RMAN>
```

2. Show all RMAN settings. Notice the `CONFIGURE CONTROLFILE AUTOBACKUP ON;` setting.

```
RMAN> SHOW ALL;

using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name
ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO
'%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO
BACKUPSET; # default
```

```
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #
default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT'
OPTIMIZE FOR LOAD TRUE ; # default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/u01/app/oracle/product/18.0.0/dbhome_1/dbs/snapcf_ORCL.f'; #
default

RMAN>
```

3. Question: In your configuration, does RMAN automatically back up the control file and server parameter file (SPFILE) with every backup and database structural change?

   Answer: Yes, it does because the `CONTROLFILE AUTOBACKUP` attribute is set to `ON`.

4. Question: Will a backup operation back up all control files or only one of the multiplexed control files?

   Answer: It will back up only one of the multiplexed control files because all control files in a database are identical.

5. Exit RMAN.

```
RMAN> EXIT

Recovery Manager complete.
[oracle@MYDBCS ~]$
```

Lab 22: Creating a Whole Database Backup

Overview

In this practice, you use Recovery Manager to back up your entire database, including the archived redo log files, the SPFILE, and the control files. The backup should be the base for an incremental backup strategy.

Assumptions

You are logged in to the compute node as the `oracle` user.

You completed the following practices:

- Practice 17-2 Configure the Size of the Fast Recovery Area
- Practice 18-2 Verifying Automatic Backups of the Control File and SPFILE
- Practice 18-5: Checking Storage Availability

Tasks

1. Start Oracle Recovery Manager (RMAN) and connect to the CDB root as the `SYS` user.

```
[oracle@MYDBCS ~]$ rman target /

Recovery Manager: Release 18.0.0.0.0 - Production on Thu Apr 5
13:43:04 2018
Version 18.1.0.0.0

Copyright (c) 1982, 2018, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: ORCL (DBID=2299035716)

RMAN>
```

2. View the structure of the CDB in terms of PDBs, tablespaces, and data files (permanent and temporary). Your file numbers will differ from those shown below.

```
RMAN> REPORT schema;

using target database control file instead of recovery catalog
Report of database schema for database with db_unique_name ORCL

List of Permanent Datafiles
===========================
File Size(MB) Tablespace           RB segs Datafile Name
---- -------- -------------------- ------- ------------------------
1    870      SYSTEM               YES
/u02/app/oracle/oradata/ORCL/system01.dbf
```

```
3    1480     SYSAUX               NO
/u02/app/oracle/oradata/ORCL/sysaux01.dbf
4    60       UNDOTBS1             YES
/u02/app/oracle/oradata/ORCL/undotbs01.dbf
5    340      PDB$SEED:SYSTEM      NO
/u02/app/oracle/oradata/ORCL/pdbseed/system01.dbf
6    620      PDB$SEED:SYSAUX      NO
/u02/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf
7    5        USERS                NO
/u02/app/oracle/oradata/ORCL/users01.dbf
8    200      PDB$SEED:UNDOTBS1    NO
/u02/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf
12   350      PDB1:SYSTEM       YES
/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf
13   790      PDB1:SYSAUX       NO
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf
14   200      PDB1:UNDOTBS1     YES
/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf
15   50       PDB1:USERS        NO
/u02/app/oracle/oradata/ORCL/PDB1/PDB1_users01.dbf
19   350      PDB2:SYSTEM       YES
/u02/app/oracle/oradata/ORCL/PDB2/system01.dbf
20   670      PDB2:SYSAUX       NO
/u02/app/oracle/oradata/ORCL/PDB2/sysaux01.dbf
21   200      PDB2:UNDOTBS1     YES
/u02/app/oracle/oradata/ORCL/PDB2/undotbs01.dbf
38   7        PDB1:INVENTORY    NO
/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF

List of Temporary Files
=======================
File Size(MB) Tablespace           Maxsize(MB) Tempfile Name
---- -------- -------------------- ----------- --------------------
1    130      TEMP                 32767
/u04/app/oracle/oradata/temp/temp01.dbf
2    62       PDB$SEED:TEMP        32767
/u04/app/oracle/oradata/temp/pdbseed_temp012018-02-08_13-49-27-
256-PM.dbf
3    62       PDB2:TEMP         32767
/u04/app/oracle/oradata/temp/PDB2/pdbseed_temp012018-02-08_13-
49-27-256-PM.dbf
4    62       PDB1:TEMP         32767
/u04/app/oracle/oradata/temp/temp012018-02-08_13-49-27-256-
PM.dbf


RMAN>
```

3. Back up the whole database. Your results will be different from the results shown below; for example, the piece handle names will be different.

```
RMAN> BACKUP DATABASE;

Starting backup at 09-APR-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=46 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001
name=/u02/app/oracle/oradata/ORCL/system01.dbf
input datafile file number=00003
name=/u02/app/oracle/oradata/ORCL/sysaux01.dbf
input datafile file number=00004
name=/u02/app/oracle/oradata/ORCL/undotbs01.dbf
input datafile file number=00007
name=/u02/app/oracle/oradata/ORCL/users01.dbf
channel ORA_DISK_1: starting piece 1 at 09-APR-18
channel ORA_DISK_1: finished piece 1 at 09-APR-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04
_09/o1_mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp
tag=TAG20180409T160501 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:01:25
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00013
name=/u02/app/oracle/oradata/ORCL/PDB1/sysaux01
…
channel ORA_DISK_1: finished piece 1 at 09-APR-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/69350B8874FA03C8E
053A23F160AC9F7/backupset/2018_04_09/o1_mf_nnndf_TAG20180409T160
501_fdql1xjr_.bkp tag=TAG20180409T160501 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:55
Finished backup at 09-APR-18

Starting Control File and SPFILE Autobackup at 09-APR-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_09/o1_mf_s_973008564_fdql3o2s_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 09-APR-18

RMAN>
```

4. Question: Do you have to shut down the database to back it up?

   Answer: No, as long as the database is in ARCHIVELOG mode, the backup can take place while the database is opened. This is a hot backup (or online backup). A cold backup (or offline backup) is a backup completed while the database is closed and is required if the database is in NOARCHIVELOG mode.

5. Question: Are hot backups consistent?

   Answer: Online backups are inconsistent because with the database opened, there is no guarantee that the data files are synchronized with the control files. However, offline backups taken while the database is not opened are consistent because, at the time of the backup, the system change number (SCN) in data file headers matches the SCN in the control files.

6. Question: What would allow hot backups (inconsistent backups) to perform a complete database recovery?

   Answer: During a complete recovery, restored online backups are recovered until the current SCN is matched, with the use of the archive log files and online redo log files.

7. Question: Did the backup include the SPFILE and control files?

   Answer: Yes. This is the last operation completed at the end of the backup command.

```
…
Starting Control File and SPFILE Autobackup at 09-APR-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_09/o1_mf_s_973008564_fdql3o2s_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 09-APR-18
```

8. Question: Does the complete operation create a single backup set?

   Answer: No. The operation creates multiple backup sets.

   - Four backup sets including data files (one for each of the containers): CDB root, PDB seed, PDB1, PDB2
   - One backup set for the SPFILE and control files.

9. List the backup sets. Look for Piece Name in the results for each backup set.

```
RMAN> LIST BACKUP;


List of Backup Sets
===================



BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
------- ----    -------------              ----------------
---

2       Full    17.95M     DISK        00:00:01     05-APR-18
        BP Key: 2   Status: AVAILABLE  Compressed: NO  Tag:
TAG20180405T170539
```

```
        Piece Name:
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_05/o1
_mf_s_972666339_fdf3x4bf_.bkp
  SPFILE Included: Modification time: 05-APR-18
  SPFILE db_unique_name: ORCL
  Control File Included: Ckp SCN: 2845204      Ckp time: 05-APR-
18
…

BS Key  Type LV Size       Device Type Elapsed Time Completion
Time
---------- ------ --- ------------- ----------- ------------ ----------
---
10      Full    18.23M     DISK        00:00:01     09-APR-18
        BP Key: 10   Status: AVAILABLE  Compressed: NO  Tag:
TAG20180409T160924
        Piece Name:
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_09/o1
_mf_s_973008564_fdql3o2s_.bkp
  SPFILE Included: Modification time: 09-APR-18
  SPFILE db_unique_name: ORCL
  Control File Included: Ckp SCN: 3108607      Ckp time: 09-APR-
18

RMAN>
```

10. Exit RMAN.

```
RMAN> EXIT
```

11. Verify that the files are stored on disk in theFRA.

```
$ cd /u03/app/oracle/fast_recovery_area/ORCL
$ ls -ltR
.:
total 18608
-rwxr-x--- 1 oracle oinstall 19021824 Apr  9 16:19 control02.ctl
drwxr-x--- 3 oracle oinstall     4096 Apr  9 16:08
69350B8874FA03C8E053A23F160AC9F7
drwxr-x--- 3 oracle oinstall     4096 Apr  9 16:07
64B4AE270F061AFDE0536604C40A9006
drwxr-x--- 3 oracle oinstall     4096 Apr  9 16:06
692121551ED82717E053E20D130AEB6C
drwxr-x--- 3 oracle oinstall     4096 Apr  9 16:05 backupset
drwxr-x--- 5 oracle oinstall     4096 Apr  9 15:01 autobackup
drwxr-x--- 5 oracle oinstall     4096 Apr  9 14:33 archivelog
drwxr-x--- 2 oracle oinstall     4096 Apr  5 16:51 flashback
drwxr-x--- 2 oracle oinstall     4096 Apr  5 16:49 onlinelog
```

```
./69350B8874FA03C8E053A23F160AC9F7:
total 4
drwxr-x--- 3 oracle oinstall 4096 Apr  9 16:08 backupset


…
./archivelog/2018_04_06:
total 0

./archivelog/2018_04_05:
total 0

./flashback:
total 2097176
-rwxr-x--- 1 oracle oinstall 1073750016 Apr  9 16:15
o1_mf_fdf30qbz_.flb
-rwxr-x--- 1 oracle oinstall 1073750016 Apr  5 16:52
o1_mf_fdf32pm6_.flb

./onlinelog:
total 0
[oracle@MYDBCS ORCL]$
```

a. Question: Where are the backups of control files and SPFILE located?

   Answer: They are created in the `autobackup` subdirectory.

b. Question: How are backups deleted?

   Answer: Space management in the FRA is governed by a backup retention policy. A retention policy determines when files are obsolete, which means that they are no longer needed to meet your data recovery objectives. The Oracle Database server automatically manages this storage by deleting files that are no longer needed.

12. View the backup retention policy.

    a. Start RMAN and connect to the CDB root as the `SYS` user.

```
$ rman target /
```

    b. Issue the `SHOW RETENTION POLICY` command. The policy is `REDUNDANCY 1`.

```
RMAN> SHOW RETENTION POLICY;


using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name
ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default


RMAN>
```

13. Question: How does Oracle determine when files are obsolete?

    Answer: There are two retention policy parameters that are mutually exclusive:

    - If a retention policy is enabled with `RECOVERY WINDOW OF 5 DAYS`, the window stretches from the current time (`SYSDATE`) to the point of recoverability, which is the earliest date to which you want to recover. The point of recoverability is `SYSDATE - integer days in the past`.
    - If a retention policy is enabled with `REDUNDANCY r`, then RMAN skips backups only if at least *n* backups of an identical file exist on the specified device, where `n=r+1` (default is 1).

    RMAN automatically deletes obsolete backup sets and copies in the FRA when space is needed.

14. Manually delete obsolete files by issuing the `DELETE OBSOLETE` command.

```
RMAN> delete obsolete;

using target database control file instead of recovery catalog
RMAN retention policy will be applied to the command
RMAN retention policy is set to redundancy 1
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=4 device type=DISK
Deleting the following obsolete backups and copies:
Type                 Key    Completion Time    Filename/Handle
-------------------- ------ -------------------------------------
---
Backup Set           2      05-APR-18
  Backup Piece       2      05-APR-18
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_05/o1
_mf_s_972666339_fdf3x4bf_.bkp
Backup Set           3      06-APR-18
  Backup Piece       3      06-APR-18
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_06/o1
_mf_s_972751516_fdhq2y5d_.bkp
Backup Set           4      09-APR-18
  Backup Piece       4      09-APR-18
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_09/o1
_mf_s_973004494_fdqg4gqp_.bkp
Backup Set           5      09-APR-18
  Backup Piece       5      09-APR-18
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_09/o1
_mf_s_973005697_fdqhb2cd_.bkp

Do you really want to delete the above objects (enter YES or
NO)? yes
deleted backup piece
```

```
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_05/o1_mf_s_972666339_fdf3x4bf_.bkp RECID=2 STAMP=972666340
deleted backup piece
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_06/o1_mf_s_972751516_fdhq2y5d_.bkp RECID=3 STAMP=972751517
deleted backup piece
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_09/o1_mf_s_973004494_fdqg4gqp_.bkp RECID=4 STAMP=973004494
deleted backup piece
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_09/o1_mf_s_973005697_fdqhb2cd_.bkp RECID=5 STAMP=973005698
Deleted 4 objects



RMAN>
```

15. Back up the database and archive logs as image copies. At the same time, free space in the FRA by deleting the archive log files once they are backed up.

    a. Perform the backup.

```
RMAN> BACKUP AS COPY DATABASE PLUS ARCHIVELOG DELETE INPUT;

Starting backup at 06-JUN-18
current log archived
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=284 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=15 RECID=15 STAMP=978029619
output file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_06_
06/o1_mf_1_15_fkj1fpws_.arc RECID=21 STAMP=978105797
…
Finished backup at 06-JUN-18

Starting backup at 06-JUN-18
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
input datafile file number=00003
name=/u02/app/oracle/oradata/ORCL/sysaux01.dbf
…
Starting Control File and SPFILE Autobackup at 06-JUN-18
```

```
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
6_06/o1_mf_s_978106455_fkj22r3p_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 06-JUN-18


RMAN>
```

b. Question: What would you do if an error such as the following occurs?

```
RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
===============
RMAN-00571:
===========================================================
RMAN-03002: failure of backup plus archivelog command at
01/23/2017 11:05:08
ORA-19809: limit exceeded for recovery files
ORA-19804: cannot reclaim 67108864 bytes disk space from
19327352832 bytes limit
```

Answer: Increase the `DB_RECOVERY_FILE_DEST_SIZE` parameter value to `30G` by issuing the following command:

```
RMAN> ALTER SYSTEM SET db_recovery_file_dest_size = 30G
SCOPE=both;
```

c. Question: What is the advantage of creating backups as image copies?

Answer: The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy, only the file or files need to be retrieved from your backup location. With backup sets, the entire backup set must be retrieved from your backup location before you extract the file or files that are needed.

d. Question: What is the advantage of creating backups as backup sets?

Answer: The advantage of creating backups as backup sets is better space usage. In most databases, 20% or more of the data blocks are empty blocks. Image copies back up every data block, even if the data block is empty. Backup sets significantly reduce the space required by the backup. In most systems, the advantages of backup sets outweigh the advantages of image copies.

e. Question: How many image copies of the data files are created?

Answer: There are 15 image copies, one image copy for each data file in the CDB, PDBs included.

16. Exit RMAN.

```
RMAN> EXIT
```

Lab 23: Creating Partial Database Backups

Overview

In this practice, you use Recovery Manager to back up PDB1, including the archived redo log files. This is a partial database backup.

Assumptions

You are logged in to the compute node as the `oracle` user.

Tasks

1. Start Recovery Manager (RMAN) and connect to the CDB root as the SYS user.

```
[oracle@MYDBCS ORCL]$ rman target /

Recovery Manager: Release 18.0.0.0.0 - Production on Wed Jun 6
16:48:45 2018
Version 18.1.0.0.0

Copyright (c) 1982, 2018, Oracle and/or its affiliates.  All
rights reserved.

connected to target database: ORCL (DBID=1505229725)

RMAN>
```

2. Back up PDB1, including the archived redo log files.

```
RMAN> BACKUP PLUGGABLE DATABASE PDB1 PLUS ARCHIVELOG;

Starting backup at 06-JUN-18
current log archived
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=41 device type=DISK
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=14 RECID=22 STAMP=978105812
…
Starting Control File and SPFILE Autobackup at 06-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
6_06/o1_mf_s_978108946_fkj4jm29_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 06-JUN-18

RMAN>
```

3. Exit RMAN.

```
RMAN> EXIT
Recovery Manager complete.
[oracle@MYDBCS ORCL]$
```

4. Question: Did the partial backup automatically include the SPFILE and control files?

   Answer: Yes. The setting verified in Practice 18-2 Verifying Automatic Backups of the Control File and SPFILE is also valid for partial backups.

5. Question: How many backup sets are created?

   Answer: Four backup sets: one for the PDB data files, one for the SPFILE and control file, one for the archived log files before the data file backup set, and one for the archived log files after the data file backup set.

6. Question: Can you connect in RMAN directly to the PDB to perform the same backup?

   Answer: Yes. In this case, you do not have to specify that you want to back up a PDB. Instead, you can use the simple BACKUP DATABASE command.

7. Perform a partial database backup in `PDB1` directly.

   a. Start RMAN and connect to `PDB1` as the `SYS` user.

```
[oracle@MYDBCS ORCL]$ rman target SYS/password@PDB1
…
connected to target database: ORCL:PDB1 (DBID=2133829969)

RMAN>
```

   b. Execute the `BACKUP DATABASE` command. Notice that the SPFILE and control file are not backed up.

```
RMAN> BACKUP DATABASE;
Starting backup at 06-JUN-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=41 device type=DISK
…
channel ORA_DISK_1: starting piece 1 at 06-JUN-18
channel ORA_DISK_1: finished piece 1 at 06-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6D5DA61701C1285EE
0533E89810ACAA5/backupset/2018_06_06/o1_mf_nnndf_TAG20180606T165
917_fkj4q7w6_.bkp tag=TAG20180606T165917 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:01:25
Finished backup at 06-JUN-18

RMAN>
```

8. Try to configure the recovery setting for the PDB so that the SPFILE and control file are backed up too. You get an error message because you must be connected to the CDB root to configure any recovery settings.

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
===============
RMAN-00571:
===========================================================
RMAN-03002: failure of configure command at 06/06/2018 17:02:27
RMAN-07536: command not allowed when connected to a Pluggable
Database

RMAN>
```

9. Exit RMAN.

```
RMAN> EXIT
```

10. Back up the `TBS_APP`tablespace in `PDB2`.

   a. Connect to `PDB2` as the `SYS` user.

```
$ rman target SYS/password@PDB2
…
connected to target database: ORCL:PDB2 (DBID=3237529478)

RMAN>
```

   b. Back up the tablespace.

```
RMAN> BACKUP TABLESPACE tbs_app;
Starting backup at 06-JUN-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=55 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00059
name=/u02/app/oracle/oradata/ORCL/PDB2/tbs_app01.dbf
channel ORA_DISK_1: starting piece 1 at 06-JUN-18
channel ORA_DISK_1: finished piece 1 at 06-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6D975E8B80B85F14E
0537A051D0A3C0D/backupset/2018_06_06/o1_mf_nnndf_TAG20180606T170
407_fkj5091o_.bkp tag=TAG20180606T170407 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:04
Finished backup at 06-JUN-18
```

```
RMAN>
```

c. Exit RMAN.

```
RMAN> EXIT
Recovery Manager complete.
[oracle@DKKDBCS ORCL]$
```

11. Can you connect to the CDB root and perform the same operation?

a. Start RMAN and connect to the CDB root as the `SYS` user.

```
$ rman target /
…
connected to target database: ORCL (DBID=1505229725)

RMAN>
```

b. Back up the `TBS_APP` tablespace in `PDB2`. You must specify the PDB in which the tablespace exists.

```
RMAN> BACKUP TABLESPACE PDB2:tbs_app;
Starting backup at 06-JUN-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=271 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00059
name=/u02/app/oracle/oradata/ORCL/PDB2/tbs_app01.dbf
channel ORA_DISK_1: starting piece 1 at 06-JUN-18
channel ORA_DISK_1: finished piece 1 at 06-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6D975E8B80B85F14E
0537A051D0A3C0D/backupset/2018_06_06/o1_mf_nnndf_TAG20180606T170
547_fkj53f1h_.bkp tag=TAG20180606T170547 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:03
Finished backup at 06-JUN-18

Starting Control File and SPFILE Autobackup at 06-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
6_06/o1_mf_s_978109551_fkj53l2p_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 06-JUN-18

RMAN>
```

c. Question: Did the operation back up only the tablespace data files?

Lab 24: Recovering from the Loss of a System-Critical Data File

Overview

In this practice, you recover your CDB after the data file for the `SYSTEM` tablespace (in the CDB root) has been inadvertently removed.

Tip

Because you use several windows at the same time in this practice, you may find it helpful to change the name of each of them in their banner at the top.

To set a title for a terminal window:

1. In the terminal window's menu, select **Terminal** and then **Set Title**. A Set Title dialog box is displayed.
2. In the Title box, enter the window number.
3. Click **OK**.

Assumptions

You are logged in as the `oracle` user.

You completed the following practices:

- Practice 18-2 Verifying Automatic Backups of the Control File and SPFILE
- Practice 18-4 Creating a Whole Database Backup

Tasks

Create a Loss of a System-Critical Data File

Window 1

1. Open a new terminal window and connect to the compute node. This window will be referred to as Window 1.

2) Identify the system datafile under the root container
3) Select * from dba_data_files where tablespace_name = ‘SYSTEM’ and con_id = 1
4) From the OS rm the datafile associated to that tablespace

Recover the Database by Using the `RESTORE` and `RECOVER` Commands

Window 1

1. Start Recovery Manager (RMAN) and connect to the target database (CDB root).

```
[oracle@MYDBCS ~]$ rman target /
…
connected to target database: ORCL (DBID=1500451933)

RMAN>
```

2. Issue the `RESTORE` command. You must provide the number for the missing data file.

```
RMAN> RESTORE DATAFILE 1;

Starting restore at 13-JUN-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=46 device type=DISK

channel ORA_DISK_1: restoring datafile 00001
input datafile copy RECID=2 STAMP=978678712 file
name=/u03/app/oracle/fast_recovery_area/ORCL/datafile/o1_mf_syst
em_fl1jwb73_.dbf
destination for restore of datafile 00001:
/u02/app/oracle/oradata/ORCL/system01.dbf
RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
===============
RMAN-00571:
===========================================================
RMAN-03002: failure of restore command at 06/13/2018 08:08:12
ORA-19573: cannot obtain exclusive enqueue for datafile 1
```

```
ORA-45909: restore, recover or block media recovery may be in
progress
ORA-19600: input file is datafile-copy 2
(/u03/app/oracle/fast_recovery_area/ORCL/datafile/o1_mf_system_f
l1jwb73_.dbf)
ORA-19601: output file is datafile 1
(/u02/app/oracle/oradata/ORCL/system01.dbf)
RMAN>
```

3. Question: What does the error message "cannot obtain exclusive enqueue for datafile 1" mean?

   Answer: The restore operation requires an exclusive enqueue lock on the data file 1 that is not obtainable. In this case, you have to open the database instance in `MOUNT` mode. This means that you have to shut down the database instance if it did not already abort.

4. Question: What does a database instance shut down do?

   Answer: This closes all PDBs and therefore prevents all users from working during the recovery operation.

5. Try to shut down the database instance in `IMMEDIATE` mode. You get an error.

```
RMAN> SHUTDOWN IMMEDIATE

RMAN-00571:
===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS
===============
RMAN-00571:
===========================================================
RMAN-03002: failure of shutdown command at 06/13/2018 08:10:09
ORA-01116: error in opening database file 1
ORA-01110: data file 1:
'/u02/app/oracle/oradata/ORCL/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3

RMAN>
```

6. Question: Why does the `SHUTDOWN IMMEDIATE` command fail?

   Answer: RMAN needs to close all data files cleanly by writing the current SCN to all data file headers. This cannot be done because data file 1 is missing.

7. Reconnect to the database.

8. Try to shut down the database instance in `ABORT` mode.

```
RMAN> SHUTDOWN ABORT

Oracle instance shut down
```

```

RMAN>
```

9. Start the database instance in `MOUNT` mode.

```
RMAN> STARTUP MOUNT

Oracle instance started
database mounted

Total System Global Area    2768239832 bytes

Fixed Size                     8899800 bytes
Variable Size                704643072 bytes
Database Buffers            1979711488 bytes
Redo Buffers                  74985472 bytes

RMAN>
```

10. Restore the missing data file.

```
RMAN> RESTORE DATAFILE 1;

Starting restore at 10-APR-18
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=25 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00001 to
/u02/app/oracle/oradata/ORCL/system01.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04_09/o1_
mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04
_09/o1_mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp
tag=TAG20180409T160501
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15
Finished restore at 10-APR-18

RMAN>
```

11. Recover the missing data file.

```
RMAN> RECOVER DATAFILE 1;
```

```
Starting recover at 10-APR-18
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 14 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_10/o1
_mf_1_14_fdskgq34_.arc
archived log for thread 1 with sequence 15 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_10/o1
_mf_1_15_fdsmhyob_.arc
archived log for thread 1 with sequence 16 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_10/o1
_mf_1_16_fdsmhyqz_.arc
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_
10/o1_mf_1_14_fdskgq34_.arc thread=1 sequence=14
media recovery complete, elapsed time: 00:00:01
Finished recover at 10-APR-18

RMAN>
```

12. Open the CDB root.

```
RMAN> ALTER DATABASE OPEN;

Statement processed

RMAN>
```

13. Open all PDBs.

```
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;

Statement processed

RMAN>
```

14. Exit RMAN.

```
RMAN> EXIT

Recovery Manager complete.
[oracle@MYDBCS ~]$
```

15. Start SQL*Plus and connect to the CDB root as the `SYSTEM` user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
SQL> CREATE USER c##test IDENTIFIED BY DBAdmin_1;

User created.

SQL>
```

```
$ sqlplus SYSTEM/password
…
SQL>
```

16. Try creating the `c##test` user again. This time the user is created.

17. Keep Window 1 open for the next section.

Use the Data Recovery Advisor to Recover the Database

1. **Window 2:** Open a new terminal window and execute the `RMAN_crash.sh` script to create a failure. This window will be referred to as Window 2.

Answer: Remember that the DBWR background process does not necessarily write immediately into the data files.

4. **Window 1:** Attempt to resize datafile 1. If it completes, execute the `ALTER SYSTEM SWITCH LOGFILE` command. You should receive an error message about the missing data file.

```
SQL> ALTER DATABASE DATAFILE 1 RESIZE 1G;
ALTER DATABASE DATAFILE 1 RESIZE 1G
*
ERROR at line 1:
ORA-01565: error in identifying file
'/u02/app/oracle/oradata/ORCL/system01.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 7


SQL>
```

5. **Window 1:** Exit SQL*Plus.

```
SQL> EXIT
…
[oracle@MYDBCS ~]$
```

6. **Window 1:** Start RMAN and connect to the target database.

```
$ rman target /
…
connected to target database (not started)

RMAN>
```

7. **Window 1:** Start the database instance in `MOUNT` mode.

```
RMAN> STARTUP MOUNT;

Oracle instance started
database mounted

Total System Global Area    2768239832 bytes

Fixed Size                     8899800 bytes
Variable Size                704643072 bytes
Database Buffers            1979711488 bytes
Redo Buffers                  74985472 bytes

RMAN>
```

8. **Window 1:** Use the `LIST FAILURE` command to determine the error. The value in the `Summary` column tells you that `system01.dbf` is missing.

```
RMAN> LIST FAILURE;

using target database control file instead of recovery catalog
Database Role: PRIMARY

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
262        CRITICAL OPEN      10-APR-18     System datafile 1:
'/u02/app/oracle/oradata/ORCL/system01.dbf' is missing

RMAN>
```

9. **Window 1:** Display repair options. At the very end of the results, a repair script is listed.

```
RMAN> ADVISE FAILURE;

Database Role: PRIMARY

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
262        CRITICAL OPEN      10-APR-18     System datafile 1:
'/u02/app/oracle/oradata/ORCL/system01.dbf' is missing

analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=25 device type=DISK
analyzing automatic repair options complete

Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
=======================
1. If file /u02/app/oracle/oradata/ORCL/system01.dbf was
unintentionally renamed or moved, restore it
```

```
Automated Repair Options
========================
Option Repair Description
-------- -----------------------
1      Restore and recover datafile 1
  Strategy: The repair includes complete media recovery with no
data loss
  Repair script:
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_2771153169.hm

RMAN>
```

10. **Window 1:** Use the `REPAIR FAILURE PREVIEW` command to generate a script with all repair actions and comments.

```
RMAN> REPAIR FAILURE PREVIEW;

Strategy: The repair includes complete media recovery with no
data loss
Repair script:
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_2771153169.hm

contents of repair script:
   # restore and recover datafile
   restore ( datafile 1 );
   recover datafile 1;
   sql 'alter database datafile 1 online';

RMAN>
```

11. **Window 1:** Use the `REPAIR FAILURE` command to repair database failures identified by the Data Recovery Advisor. When prompted, enter `YES` to execute the repair. When prompted to open the database, enter `YES`.

```
RMAN> REPAIR FAILURE;

Strategy: The repair includes complete media recovery with no
data loss
Repair script:
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_2771153169.hm

contents of repair script:
   # restore and recover datafile
   restore ( datafile 1 );
   recover datafile 1;
   sql 'alter database datafile 1 online';
```

```
Do you really want to execute the above repair (enter YES or
NO)? YES
executing repair script

Starting restore at 10-APR-18
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00001 to
/u02/app/oracle/oradata/ORCL/system01.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04_09/o1_
mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04
_09/o1_mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp
tag=TAG20180409T160501
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:07
Finished restore at 10-APR-18

Starting recover at 10-APR-18
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 14 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_10/o1
_mf_1_14_fdskgq34_.arc
archived log for thread 1 with sequence 15 is already on disk as
file
…
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_
10/o1_mf_1_16_fdsmhyqz_.arc thread=1 sequence=16
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_
10/o1_mf_1_17_fdsn3sf0_.arc thread=1 sequence=17
media recovery complete, elapsed time: 00:00:01
Finished recover at 10-APR-18

sql statement: alter database datafile 1 online
```

```
repair failure complete

Do you want to open the database (enter YES or NO)? YES
database opened


RMAN>
```

12. **Window 1:** Open all the PDBs.

```
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;

Statement processed


RMAN>
```

13. **Window 1:** Exit RMAN.

```
RMAN> EXIT


Recovery Manager complete.
[oracle@MYDBCS ~]$
```

14. **Window 2:** Close the terminal window.