ORACLE
UNIVERSITY

Integrated Cloud Applications & Platform Services

Oracle Database 12*c* R2: New Features for 12*c* R1 Administrators

Activity Guide – Volume I

D93517GC10

Edition 1.0 | December 2016 | D98179

Learn more from Oracle University at **education.oracle.com**

ORACLE

Authors

Dominique Jeunot, Jean-François Verrier, Mark Fuller and James Spiller

Technical Contributors and Reviewers

Krishnanjani Chitta, Randall Richeson, Gerlinde Frenzen, Sailaja Pasupuleti, Anita Mukundan, Harald van Breederode, Jim Stenoish and Branislav Valny

This book was published using: Oracle Tutor

Table of Contents

Course Practice Environment: Security Credentials

Chapter I

Course Practice Environment: Security Credentials

For OS usernames and passwords, see the following:

- If you are attending a classroom-based or live virtual class, ask your instructor or LVC producer for OS credential information.
- If you are using a self-study format, refer to the communication that you received from Oracle University for this course.

For product-specific credentials used in this course, see the following table:

Product-Specific Credentials		
Product/Application	**Username**	**Password**
Enterprise Manager Database Express	SYS	oracle_4U
All CDBs and PDBs users	SYS, SYSTEM, OE …	oracle_4U
Enterprise Manager Cloud Control	sysman	Oracle123

Practices for Lesson 1: Introduction

Chapter 1

Practices for Lesson 1: Overview

Practices Overview

Your system currently has two VMs:

- VM1 dedicated to RDBMS: Both Oracle Database 12.1.0.2 and 12.2 are installed, with three pre-created databases.
  - The `ORCL` and `cdb2` databases are 12.2 CDBs. `ORCL` and `cdb2` either act as production or test databases.
  - The `cdb1` database is a production 12.1.0.2 database that will be upgraded to 12.2.
- VM2 dedicated to EM CC: The `cdbem` database is a 12.1.0.2 database. `cdbem` and more precisely the `pdbem` PDB is the repository database for Enterprise Manager Cloud Control (EM CC).

Net service names are already logged in the `tnsnames.ora` file for all new net services that will be created for CDBs and PDBs.

In the practices for this lesson, you will verify that Enterprise Manager Database Express is configured for `ORCL` database, and will configure EM CC to be aware of `ORCL` database target and set up the named credentials for `ORCL`, which will be the CDB used in most of the practices.

Note: The configuration for the `ORCL` database matches the pre-configured directories on an Oracle Cloud compute node associated with the pre-created `ORCL` database of a database deployment (or Database Cloud Service instance). This will be covered in a further lesson in the course.

- CDB datafiles in `/u02/app/oracle/oradata/`
- CDB root datafiles in `/u02/app/oracle/oradata/ORCL`
- PDB datafiles in `/u02/app/oracle/oradata/ORCL/pdb_orcl` (The pre-configured PDB on Oracle Cloud is `pdb1` and therefore the subdirectory is `pdb1`.)
- Controlfiles in `/u02/app/oracle/oradata/ORCL` and `/u03/app/oracle/fast_recovery_area/ORCL`
- All redo log files in `/u04/app/oracle/redo/ORCL` (The pre-configured directory for redo logs of the unique CDB on Oracle Cloud is `/u04/app/oracle/redo`.)
- All backup files in `/u03/app/oracle/fast_recovery_area/ORCL`
- Password and init files in `$ORACLE_HOME/dbs`
- Diagnostics files in `/u01/app/oracle/diag/rdbms/orcl/ORCL/…`
- TDE wallet in `/u01/app/oracle/admin/ORCL/tde_wallet`
- Net files in `$ORACLE_HOME/network/admin`

Practice 1-1: Starting Enterprise Manager Database Express

Overview

On VM1, you will check whether Enterprise Manager Database Express is configured for `ORCL` database. If this is not the case, you will configure it and connect to `ORCL` database through Enterprise Manager Database Express. You will also check if you can use Enterprise Manager Database Express to connect directly to `pdb_orcl`.

Tasks

1. Before starting the practice, execute the `$HOME/labs/admin/glogin_1.sh` shell script. It sets formatting for all columns selected in queries.

```
$ $HOME/labs/admin/glogin_1.sh
$
```

2. Check whether Enterprise Manager Database Express is configured for `ORCL`.
   a. Verify that the value of the `DISPATCHERS` instance parameter is set to `(PROTOCOL=TCP)(SERVICE=ORCLXDB)` in the `ORCL` instance.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> SHOW PARAMETER dispatchers
NAME                 TYPE       VALUE
-------------------- ---------- --------------------------------
dispatchers          string     (PROTOCOL=TCP)(SERVICE=ORCLXDB)
max_dispatchers      integer

SQL>
```

   b. Select the port number used for Enterprise Manager Database Express.

```
SQL> SELECT dbms_xdb_config.gethttpsport FROM DUAL;

GETHTTPSPORT
------------
        5500

SQL> EXIT
$
```

   c. Verify that the listener is running and listens to the localhost (*yourserver*) using TCP protocol, the port 5500 for `ORCL`, the http presentation with RAW session data.

```
$ lsnrctl status

…
Listening Endpoints Summary...
```

```
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=<Your
hostname>)(PORT=1521)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<Your
hostname>)(PORT=5502))(Security=(my_wallet_directory=/u01/app/or
acle/admin/cdb1/xdb_wallet))(Presentation=HTTP)(Session=RAW))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<Your
hostname>)(PORT=5500))(Security=(my_wallet_directory=/u01/app/or
acle/admin/ORCL/xdb_wallet))(Presentation=HTTP)(Session=RAW))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<Your
hostname>)(PORT=5501))(Security=(my_wallet_directory=/u01/app/or
acle/admin/cdb2/xdb_wallet))(Presentation=HTTP)(Session=RAW))


Services Summary...
…
The command completed successfully
$
```

d. Launch a browser and use the following URL https://localhost:5500/em.

e. Most probably, you receive a Secure Connection Failed message and you need to add a security exception. At the end of the alert box, click **I Understand the Risks**.

f. At the bottom of the page, click **Add Exception**.

   Confirm that "Permanently store this exception" is selected in your training environment and click **Confirm Security Exception**.

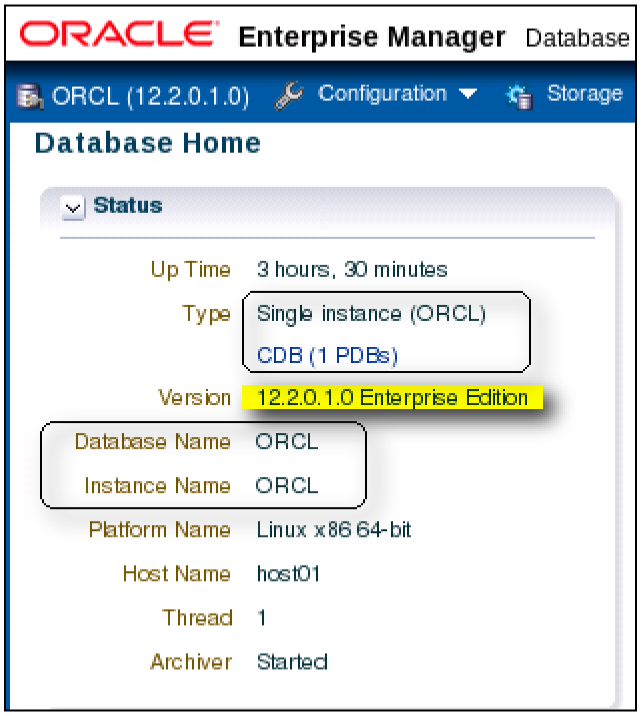
g. Enter **`sys`** in the User Name field. Enter the password in the Password field.

   Q/ What do you observe against previous versions of Enterprise Manager Database Express?

   A/ Login into a CDB allows you to define a container name, which means that you can connect to a PDB and not necessarily to the CDB root.

h. Leave the Containers field empty as you want to log in to the CDB root and not to any specific PDB. Check the `as sysdba` box. Then click **Login**.

Observe that the Oracle Database version for the database instance is 12.2.



Database Home

Status

Up Time	3 hours, 30 minutes
Type	Single instance (ORCL)
	CDB (1 PDBs)
Version	12.2.0.1.0 Enterprise Edition
Database Name	ORCL
Instance Name	ORCL
Platform Name	Linux x86 64-bit
Host Name	host01
Thread	1
Archiver	Started

3. Verify that you can use Enterprise Manager Database Express to connect directly to `pdb_orcl`. If `pdb_orcl` is not opened, open it.

   a. Connect to `pdb_orcl` and select the port number used for Enterprise Manager Database Express. If none is configured, use the port 5520.

```
$ sqlplus system@pdb_orcl
Enter password: ******

SQL> SELECT dbms_xdb_config.gethttpsport FROM DUAL;

GETHTTPSPORT
------------
           0

SQL> EXEC dbms_xdb_config.sethttpsport(5520)

PL/SQL procedure successfully completed.

SQL> SELECT dbms_xdb_config.gethttpsport FROM DUAL;

GETHTTPSPORT
------------
        5520

SQL> EXIT
$
```

b. Verify that the listener is running and listens to the localhost (*yourserver*) using TCP protocol, the port 5520 for `pdb_orcl`, the http presentation with RAW session data.

```
$ lsnrctl status

…
Listening Endpoints Summary...
     (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<Your
hostname>)(PORT=5500))(Security=(my_wallet_directory=/u01/app/or
acle/admin/ORCL/xdb_wallet))(Presentation=HTTP)(Session=RAW))
   (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<Your
hostname>)(PORT=5520))(Security=(my_wallet_directory=/u01/app/or
acle/admin/ORCL/xdb_wallet))(Presentation=HTTP)(Session=RAW))Sum
mary...
…
The command completed successfully
$
```

Q/ Why is the security wallet the same file for `ORCL` and `pdb_orcl`?

A/ The security wallet used is the same for the CDB and the PDBs because this is the one used for the database instance.

c. Launch a browser and use the following URL https://localhost:5520/em.

d. Most probably, you receive a Secure Connection Failed message and you need to add a security exception. At the end of the alert box, click **I Understand the Risks**.

e. At the bottom of the page, click **Add Exception**.

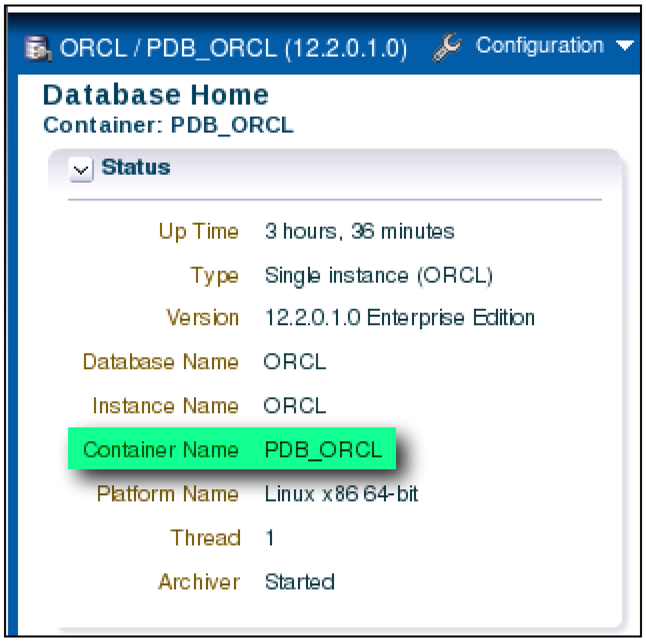
Confirm that "Permanently store this exception" is selected in your training environment and click **Confirm Security Exception**.

Q/ Do you notice any difference in the Login window?

A/ The Login window to connect to the CDB root suggests a Container Name field whereas the Login window to connect to the PDB does not suggest a Container Name field because the container is preselected in the URL.

f. Enter `sys` in the User Name field. Enter the password in the Password field. Check the `as sysdba` box. Then click **Login**.

Observe that the Oracle Database version for the database instance is 12.2.



Database Home

Container: PDB_ORCL

Status

Up Time	3 hours, 36 minutes
Type	Single instance (ORCL)
Version	12.2.0.1.0 Enterprise Edition
Database Name	ORCL
Instance Name	ORCL
Container Name	PDB_ORCL
Platform Name	Linux x86 64-bit
Thread	1
Archiver	Started

Practice 1-2: Configuring Enterprise Manager Cloud Control Targets

Overview

You act as an Enterprise Manager administrator. You access Oracle Enterprise Manager Cloud Control 13*c* as the `sysman` user and select **Summary** as your home page. You start exploring some of the Oracle Enterprise Manager Cloud Control 13*c* functionalities through the different menus and options. And lastly, you add the `ORCL` and `cdb2` databases as monitored targets.

The EM CC repository is a 12.1.0.2 PDB, `pdbem` in `cdbem` CDB.

Tasks

1. VM2 being dedicated to EM CC, login to VM2 as the `oracle` Unix user. Click the Firefox icon on the top panel (toolbar region) above the desktop to open a browser to access the Enterprise Manager Cloud Control console.
2. Enter the URL for Cloud Control: **Error! Hyperlink reference not valid.**. In the current setup, use `https://localhost:7802/em`. If an error appears, you must first start the OMS; else proceed directly with step 3.
   a. If the listener is not running, start the listener.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdbem
The Oracle base remains unchanged with value /u01/app/oracle
$ lsnrctl status
LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 05-OCT-
2016 06:25:34

Copyright (c) 1991, 2016, Oracle.  All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=edcdr19p1.us.oracle.co
m)(PORT=1521)))
TNS-12541: TNS:no listener
 TNS-12560: TNS:protocol adapter error
  TNS-00511: No listener
   Linux Error: 111: Connection refused
$ lsnrctl start
…
$
```

   b. Start the Enterprise Manager Repository Database `cdbem` if it is not already started.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdbem
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

Connected to an idle instance.
SQL> startup
```

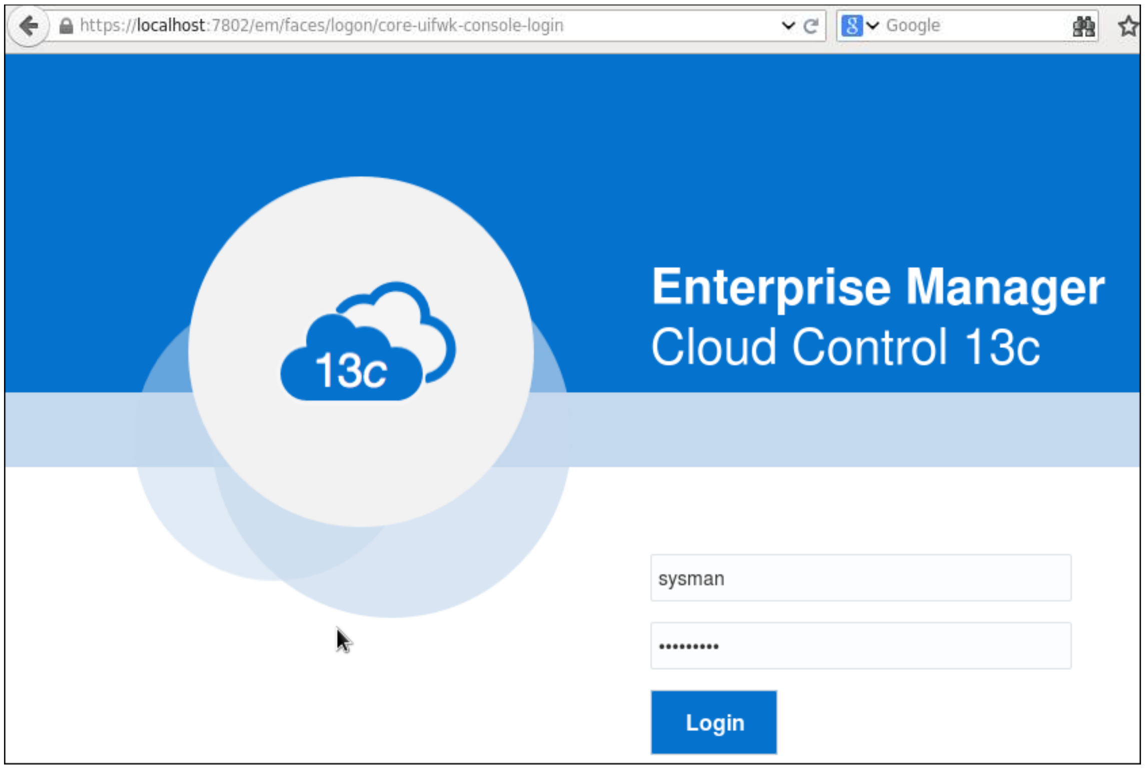
```
ORACLE instance started.

Total System Global Area  400846848 bytes
Fixed Size                  2271568 bytes
Variable Size             339740336 bytes
Database Buffers           50331648 bytes
Redo Buffers                8503296 bytes
Database mounted.
Database opened.
SQL> EXIT
$
```

c. Start the OMS.

```
$ export OMS_HOME=/u01/app/oracle/product/middleware/oms
$ $OMS_HOME/bin/emctl start oms
Oracle Enterprise Manager Cloud Control 13c Release 1
Copyright (c) 1996, 2015 Oracle Corporation.  All rights
reserved.
Starting Oracle Management Server...
WebTier Successfully Started
Oracle Management Server Successfully Started
Oracle Management Server is Up
JVMD Engine is Up
Starting BI Publisher Server ...
BI Publisher Server Already Started
BI Publisher Server is Up
$
```

3. Retry `https://localhost:7802/em`. Most probably, you receive a Secure Connection Failed message and you need to add a security exception. Click **Or you can add an exception**.
   a. At the end of the alert box, click **I Understand the Risks**.
   b. At the bottom of the page, click **Add Exception**.
   c. In the Add Security Exception pop-up window, click **Get Certificate**.
   d. Confirm that “Permanently store this exception” is selected in your training environment and click **Confirm Security Exception**.

4. Enter **`sysman`** in the first field and the password in the second field. Then click **Login**.


https://localhost:7802/em/faces/logon/core-uifwk-console-login

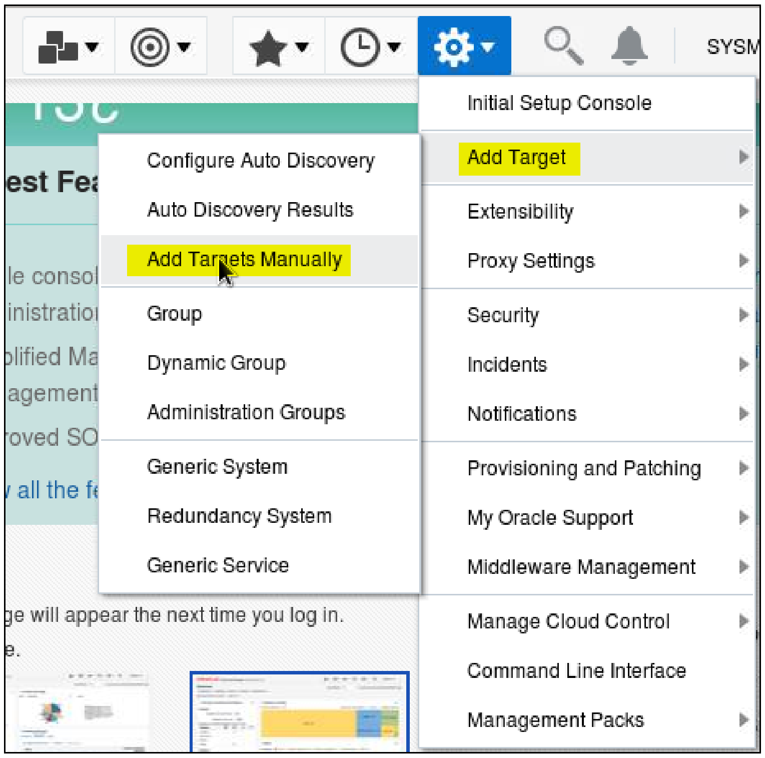
Enterprise Manager Cloud Control 13c

sysman

•••••••••

5. The Accessibility Preference page appears. The "Your accessibility preferences are presented because this is your first login. You can set these now, or at anytime by using Username menu." message appears. Click **I'll deal with this later**.
6. The first time a new user logs in to Enterprise Manager, a page asks you to accept the license agreement. You have to accept only once. Then each time you log in to Enterprise Manager, you do not get the license agreement page.
7. The "Welcome to Enterprise Manager Cloud Control 13*c*" page appears with choices, in the "Select Enterprise Manager Home page" section. Select "Databases" for the next practices, or any other choice according to your habits of EM CC usage.

8. Add the `ORCL` and `cdb2` Database Instances as new targets in Enterprise Manager Cloud Control.
   a. In the icons bar, click "Setup" (the wheel icon) > "Add Target" > "Add Targets Manually".



   b. In "Add Targets Manually", in "Overview", choose "Add Using Guided Process". Then, in "Guided Discovery", choose "Oracle Database, Listener and Automatic Storage Management". Click the "Add" button.
   c. On the "Database Discovery: Search Criteria" page, in "Specify Host or Cluster", click the magnifying glass to find your host. Select your host, and then click "Select". Then click Next.
      1) On the "Database Discovery: Results" page, in the "Databases" list, select the `ORCL (Container Database)` and `cdb2 (Container Database)` databases.
      2) Unlock the `DBSNMP` user. This user is the monitoring user used to test the connection once the target is being added. Open a terminal window.

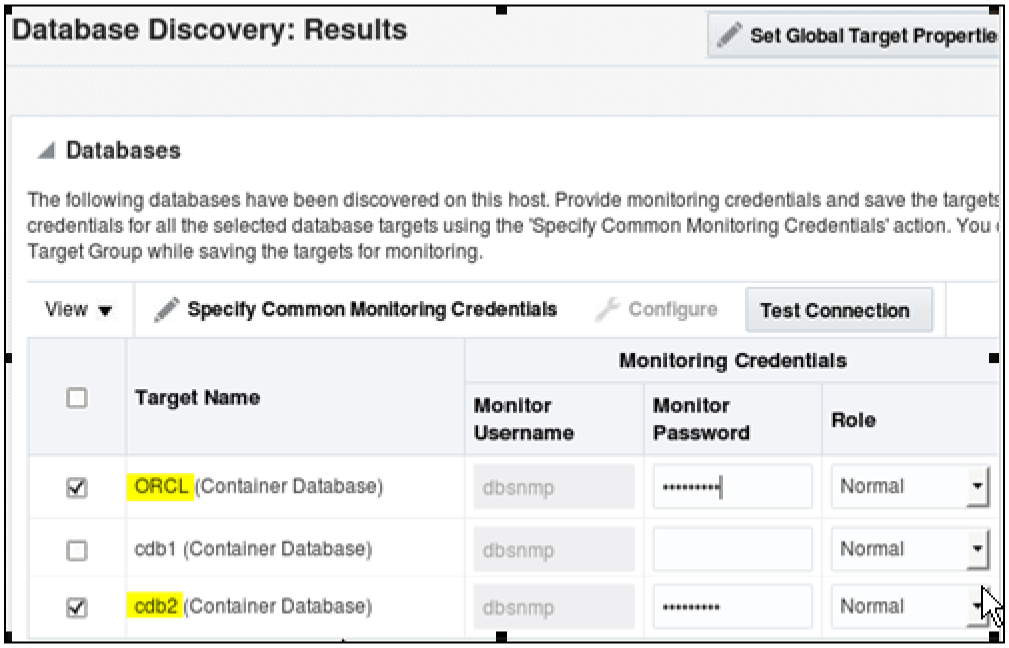
```
$ . oraenv
ORACLE_SID = [cdbem] ? ORCL
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.2.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER USER dbsnmp IDENTIFIED BY oracle_4U ACCOUNT UNLOCK
CONTAINER=ALL;


User altered.
```

```
SQL> EXIT
$
```

3) In `cdb2`, before unlocking the `DBSNMP` user, open `pdb2` if the PDB is not opened.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER USER dbsnmp IDENTIFIED BY oracle_4U ACCOUNT UNLOCK
CONTAINER=ALL;

User altered.

SQL> EXIT
$
```

4) Enter the password of `DBSNMP` for "Monitor Password" for both targets.


Database Discovery: Results
Set Global Target Propertie
Databases
View
Specify Common Monitoring Credentials
Configure
Test Connection

The following databases have been discovered on this host. Provide monitoring credentials and save the targets
credentials for all the selected database targets using the 'Specify Common Monitoring Credentials' action. You
Target Group while saving the targets for monitoring.

		Monitoring Credentials		
	Target Name	Monitor Username	Monitor Password	Role
☑	ORCL (Container Database)	dbsnmp	••••••••••	Normal
☐	cdb1 (Container Database)	dbsnmp		Normal
☑	cdb2 (Container Database)	dbsnmp	••••••••••	Normal

d. Click the "Test Connection" button. You should receive the confirmation message. Click the "OK" button, then "Next", and then "Save" to complete the operation, and finally click "Close".

Q/ Is the CDB root of each CDB the only target monitored in `ORCL` and in `cdb2`?

A/ No. All containers within `ORCL` and `cdb2` are monitored. The CDB root and `pdb_orcl`, respectively called `ORCL_CDBROOT` and `ORCL_PDB_ORCL`, are monitored in `ORCL`. The CDB root and `pdb2`, respectively called `cdb2_CDBROOT` and `cdb2_PDB2`, are monitored in `cdb2`.

Practice 1-3: Creating and Testing New Named Credentials

Overview

In this practice, still on VM2, you create and test the `ORCL_sys` credential used for any connection as `SYS` user sharable in the database instance `ORCL`.

Tasks

1. Navigate to Setup > Security > Named Credentials.
2. Click **Create**.
   a. Enter the following values:

Field	Choice or Value
General Properties	
Credential Name	**`ORCL_sys`**
Credential description	**`Credentials for Database`**
Authenticating Target Type	**`Database Instance`**
Credential type	**`Database Credentials`**
Scope	**`Target`**
Target type	**`Database Instance`**
Target Name	**`ORCL`** (Click the magnifying glass to find ORCL and select)
Credential Properties	
Username	**`SYS`**
Password	`*********`
Confirm Password	`*********`
Role	**`SYSDBA`**

   b. Test against the `ORCL` database instance. Click **Test and Save** until you get the following **Confirmation** message: **Credential Operation Successful**. This means that the credential was successful and saved.
3. Test if the named credential works when you connect to the `ORCL` target.
   Click **Targets** and then select **Databases**.
4. Click `ORCL`.
5. Click **Administration**, then **Storage**, and then **Tablespaces.** The named credential `ORCL_SYS` is displayed in the **Database Login** page.
6. Click Login to accept this named credential to log in to the `ORCL` database. You could choose New to define a new login username and password, if needed.
7. Under the `SYSMAN` menu at the top right, as soon as you click the Log Out button, different logout possibilities are suggested. Choose "Logout of Enterprise Manager and all targets" and click the Logout button.

8. Verify that the monitoring information stored in the Management Repository is stored in the `pdbem` PDB.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdbem
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus sysman@pdbem
Enter password: ******
Connected.
SQL> SELECT cred_name, cred_owner, entity_name
     FROM   em_nc_creds nc, mgmt$manageable_entities m
     WHERE  nc.target_guid  = m.entity_guid
     AND    cred_owner != '<SYSTEM>';
  2   3   4
CRED_NAME            CRED_OWNER           ENTITY_NAME
-------------------- -------------------- --------------
ORCL_SYS             SYSMAN               ORCL

SQL>
```

9. To release resources for next practices, shut down the `cdbem` database instance.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> EXIT
$
```

Practices for Lesson 2: Application Containers and Applications

Chapter 2

Practices for Lesson 2: Overview

Practices Overview

In these practices, you create the `toys_root` application container for both `robots` and `dolls` application PDBs, and install the `toys_app` application in the application container for both application PDBs.

- Create the application container and its application seed.
- Install the `toys_app` application in the `toys_root` application container.
- Create the `robots` application PDB associated to the `toys_root` application container.
- Create the `dolls` application PDB associated to the `toys_root` application container.

Then you upgrade the application and manage the application PDBs.

Practice 2-1: Installing an Application in an Application Container

Overview

In this practice, you install the `toys_app` application in the `toys_root` application container for both `robots` and `dolls` application PDBs so that both application PDBs can benefit from common entities such as application common users and schemas, application common roles, granted privileges, and application shared objects.

For this purpose, first define the application container, install the application in the application container, and finally create the two application PDBs in the application container.

Tasks

1. Before starting the practice, execute the `$HOME/labs/admin/glogin_2.sh` shell script. It sets formatting for all columns selected in queries.

```
$ $HOME/labs/admin/glogin_2.sh
$
```

2. Before creating application PDBs, you have to create an application root and optionally an application seed that can be used as a template for other future application PDBs needing the same application common entities as the application PDBs of this application container.
   a. Create the directories for the application root and application seed new containers.

```
$ . oraenv
ORACLE_SID = [cdbem] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ mkdir -p /u02/app/oracle/oradata/ORCL/toys_root/toys_SEED
$
```

   b. Create the application root.

```
$ sqlplus / AS SYSDBA

SQL> CREATE PLUGGABLE DATABASE toys_root
              AS APPLICATION CONTAINER
              ADMIN USER admin IDENTIFIED BY oracle_4U
              ROLES=(CONNECT)
         CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/toys_root';
  2    3    4

Pluggable database created.

SQL> SELECT name, con_id, application_root "APP_ROOT",
              application_seed "APP_Seed",
              application_pdb "APP_PDB",
              application_root_con_id "APP_ROOT_CONID"
         FROM   v$containers;
  2    3    4    5

```

```
NAME            CON_ID APP_ROOT APP_Seed APP_PDB APP_ROOT_CONID
--------------- ------ -------- -------- ------- --------------
CDB$ROOT             1 NO       NO       NO
PDB$SEED             2 NO       NO       NO
PDB_ORCL             3 NO       NO       NO
TOYS_ROOT            4 YES      NO       NO

SQL>
```

c. Open the application root.

```
SQL> ALTER PLUGGABLE DATABASE toys_root OPEN;

Pluggable database altered.

SQL>
```

d. Create the application seed for the application container.

Q/ Why and when would you create an application seed?

A/ An application seed is created for instantaneous provisioning of application PDBs. Synchronization of the application code in the application seed must be completed before application PDB creation so that the common tables container in the application seed can be replicated in the application PDBs.
Therefore, the application seed can be created only after the application installation in the application root.

1) Install the `toys_app` application including a table in the `toys_root` application root so that both the future `robots` and `dolls` application PDBs can benefit from the common tables.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                                          begin install '1.0';
  2
ALTER PLUGGABLE DATABASE APPLICATION toys_app
*
ERROR at line 1:
ORA-65046: operation not allowed from outside a pluggable
database
SQL>
```

Q/ To which container should you be connected to install the application?

A/ If future application PDBs associated to the `toys_root` application root should benefit from common objects like tables, the application objects should be created in the application root. Therefore, connect to the application root container.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
```

```
                                            BEGIN INSTALL '1.0';
  2
Pluggable database altered.

SQL> SELECT app_name, app_version, app_status
     FROM   dba_applications
     WHERE  app_name = 'TOYS_APP';
  2   3
APP_NAME       APP_VERSION  APP_STATUS
-------------- ------------ ------------
TOYS_APP                    INSTALLING

SQL>
```

2) Now execute the `$HOME/labs/APP/script_toys_app.sql` script to create a tablespace and a user, grant privileges and roles to users, and create a common table.

```
SQL> @$HOME/labs/APP/script_toys_app
…
SQL>
```

3) When the application script completes successfully, finish the application installation.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                                          END INSTALL '1.0';
  2
Pluggable database altered.

SQL> SELECT app_name, app_version, app_status
     FROM   dba_applications
     WHERE  app_name = 'TOYS_APP';
  2   3
APP_NAME       APP_VERSION  APP_STATUS
-------------- ------------ ------------
TOYS_APP       1.0          NORMAL

SQL>
```

4) Create the application seed for the `toys_root` application container.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> CREATE PLUGGABLE DATABASE AS SEED ADMIN USER admin
            IDENTIFIED BY oracle_4U ROLES=(CONNECT)
            CREATE_FILE_DEST =
    '/u02/app/oracle/oradata/ORCL/toys_root/toys_SEED';
```

```
  2   CREATE PLUGGABLE DATABASE AS SEED
*
ERROR at line 1:
ORA-65190: operation allowed only from within an application
root
SQL>
```

Q1/ Why does it fail?

A1/ An application seed exists as a template for future application PDBs within an application container. Therefore, it can only be created from the application root of the application container.

Q2/ Is it mandatory to create an application seed within an application container?

A2/ No.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> CREATE PLUGGABLE DATABASE AS SEED ADMIN USER admin
              IDENTIFIED BY oracle_4U ROLES=(CONNECT)
              CREATE_FILE_DEST =
    '/u02/app/oracle/oradata/ORCL/toys_root/toys_SEED';
  2    3    4
Pluggable database created.

SQL> SELECT name, con_id, application_root "APP_ROOT",
              application_seed "APP_Seed",
              application_pdb "APP_PDB",
              application_root_con_id "APP_ROOT_CONID"
     FROM    v$containers;
  2    3    4    5
NAME            CON_ID APP_ROOT APP_Seed APP_PDB APP_ROOT_CONID
--------------- ------ -------- -------- -------- -------------
TOYS_ROOT            4 YES      NO       NO
TOYS_ROOT$SEED       5 NO       YES      YES                 4

SQL>
```

Q3/ What is the name of the application seed within an application container?

A3/ The application seed's name is <`application_root`>$SEED.

5) Open the application seed.

```
SQL> ALTER PLUGGABLE DATABASE toys_root$seed OPEN;

Pluggable database altered.

SQL> SHOW pdbs
```

```
    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 READ WRITE NO

SQL>
```

3. Create the `robots` and `dolls` application PDBs associated to the `toys_root` application container.
   a. Create the directories for the application PDBs.

```
SQL> !mkdir /u02/app/oracle/oradata/ORCL/toys_root/robots

SQL> !mkdir /u02/app/oracle/oradata/ORCL/toys_root/dolls

SQL>
```

   b. Create and open the `robots` application PDB associated to the `toys_root` application container.

```
SQL> CREATE PLUGGABLE DATABASE robots
              ADMIN USER admin IDENTIFIED BY oracle_4U
              CREATE_FILE_DEST =
              '/u02/app/oracle/oradata/ORCL/toys_root/robots';
  2    3    4
CREATE PLUGGABLE DATABASE robots
*
ERROR at line 1:
ORA-65035: unable to create pluggable database from
TOYS_ROOT$SEED


SQL> !oerr ora 65035
65035, 00000, "unable to create pluggable database from %s"
// *Cause:  An attempt was made to clone a pluggable database
that did not have local undo enabled.
// *Action: Enable local undo for the PDB and retry the
operation.
//

SQL>
```

The concept of local undo has not yet been covered. Use the script `$HOME/labs/APP/script_undo.sql` to enable local undo for the PDB so that the creation of the new PDB from the seed PDB completes successfully. Then retest the creation of the new PDBs.

```
SQL> @$HOME/labs/APP/script_undo.sql
…
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> CREATE PLUGGABLE DATABASE robots
            ADMIN USER admin IDENTIFIED BY oracle_4U
            CREATE_FILE_DEST =
            '/u02/app/oracle/oradata/ORCL/toys_root/robots';
  2    3    4

Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE robots OPEN;

Pluggable database altered.

SQL>
```

c. Create and open the `dolls` application PDB associated to the `toys_root` application container.

```
SQL> CREATE PLUGGABLE DATABASE dolls
            ADMIN USER admin IDENTIFIED BY oracle_4U
            CREATE_FILE_DEST =
            '/u02/app/oracle/oradata/ORCL/toys_root/dolls';
  2    3    4
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE dolls OPEN;

Pluggable database altered.

SQL>
```

d. Connect to the `robots` application PDB and verify that the common `toys_owner.sales_data` table is replicated.

```
SQL> CONNECT toys_owner@robots
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied

Warning: You are no longer connected to Oracle.

SQL>
```

Q/ Why does the connection fail even though you created `toys_owner` as a common user in the `toys_app` application in the application root and even though the `toys_owner` common user was replicated in the application seed from which the application PDB is created?

A/ The application seed has not been synchronized with the application root.

```
SQL> CONNECT sys@toys_root$seed AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN READ ONLY;

Pluggable database altered.

SQL>
```

```
SQL> CONNECT toys_owner@robots
Enter password: ******
ERROR:
ORA-01017: invalid username/password; logon denied


Warning: You are no longer connected to Oracle.

SQL>
```

Q2/ Why does the connection still fail?

A2/ The application seed has been synchronized with the application root after the application PDBs had been created. There are two possibilities: either re-create the application PDBs from the now synchronized seed or synchronize the application PDBs with the application root. Let's try the first method for `ROBOTS` and the second method for `DOLLS`.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE robots CLOSE;

Pluggable database altered.
```

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

Pluggable database dropped.

SQL> CREATE PLUGGABLE DATABASE robots
            ADMIN USER admin IDENTIFIED BY oracle_4U
            CREATE_FILE_DEST =
            '/u02/app/oracle/oradata/ORCL/toys_root/robots';
 2    3    4
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE robots OPEN;

Pluggable database altered.

SQL>
```

```
SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL>
```

e. Reattempt to connect to the `robots` and `dolls` application PDB and verify that the common `toys_owner.sales_data` table is replicated in both application PDBs.

```
SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
SQL> DESC toys_owner.sales_data
 Name                                   Null?    Type
 ---------------------------------- -------- ------------------
 YEAR                                            NUMBER(4)
 REGION                                          VARCHAR2(10)
 QUARTER                                         VARCHAR2(4)
 REVENUE                                         NUMBER

SQL>
SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
SQL> DESC toys_owner.sales_data
 Name                                   Null?    Type
 ---------------------------------- -------- ------------------
 YEAR                                            NUMBER(4)
 REGION                                          VARCHAR2(10)
 QUARTER                                         VARCHAR2(4)
 REVENUE                                         NUMBER

SQL> EXIT
$
```

You will manage the data in common tables in application PDBs in another practice.

Practice 2-2: Upgrading an Application

Overview

In this practice, you will upgrade the `TOYS_APP` application to create the `toys_owner.new_tab` table shared by both application PDBs.

Tasks

1. Before starting the practice, execute the `$HOME/labs/APP/setup_apps.sh` shell script. The script recreates the `toys_root` application container and creates another application container, the `hr_root`.

```
$ $HOME/labs/APP/setup_apps.sh
…
$
```

2. Major changes to the application such as new common schemas, objects, and procedures need to be installed and constitute an application upgrade. The Oracle database is responsible for propagating the upgrade from the application root to all the application PDBs.

   a. Retrieve the `APP_VERSION` value that indicates the application version of an application installation at which the upgrade can be applied.

```
$ sqlplus / AS SYSDBA

SQL> SELECT app_name, app_version, app_status,con_id
     FROM   cdb_applications
     WHERE  app_name NOT LIKE 'APP$%';
  2   3
APP_NAME       APP_VERSION  APP_STATUS   CON_ID
-------------- ------------ ------------ ------
HR_APP         1.0          NORMAL            8
HR_APP         1.0          NORMAL            9
HR_APP         1.0          NORMAL           10

SQL>
```

Q/ You know that `toys_app` application has been installed successfully. Why is there no row for this application in the list?

A/ The application container is closed.

```
SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB_ORCL                       READ WRITE NO
         4 TOYS_ROOT                      MOUNTED
         5 TOYS_ROOT$SEED                 MOUNTED
         6 ROBOTS                         MOUNTED
         7 DOLLS                          MOUNTED
         8 HR_ROOT                        READ WRITE NO
         9 OPERATIONS                     READ WRITE NO
        10 RESEARCH                       READ WRITE NO
SQL>
```

b. Open the application container.

```
SQL> ALTER PLUGGABLE DATABASE toys_root OPEN;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> SELECT app_name, app_version, app_status,con_id
     FROM   cdb_applications
     WHERE  app_name NOT LIKE 'APP$%';
  2   3
APP_NAME       APP_VERSION  APP_STATUS   CON_ID
-------------- ------------ ------------ ------
TOYS_APP       1.0          NORMAL            4
TOYS_APP       1.0          NORMAL            6
TOYS_APP       1.0          NORMAL            7
HR_APP         1.0          NORMAL            8
HR_APP         1.0          NORMAL            9
HR_APP         1.0          NORMAL           10

6 rows selected.

SQL>
```

3. Start the application upgrade.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                    BEGIN UPGRADE '1.0' TO '1.1';

Pluggable database altered.
```

```
  2
ALTER PLUGGABLE DATABASE APPLICATION toys_app
*
ERROR at line 1:
ORA-65046: operation not allowed from outside a pluggable
database
```

```

SQL>
```

Q/ Why is the operation rejected?

A/ You have to connect to the application root.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                     BEGIN UPGRADE '1.0' TO '1.1';
  2
Pluggable database altered.

SQL> SELECT app_name, app_version, app_status
     FROM   dba_applications
     WHERE  app_name = 'TOYS_APP';
  2   3

APP_NAME       APP_VERSION  APP_STATUS
-------------- ------------ ------------
TOYS_APP       1.0          UPGRADING

SQL>
```

4. Execute the `@$HOME/labs/APP/script_upgrade.sql` application script.

```
SQL> @$HOME/labs/APP/script_upgrade
…
SQL>
```

5. Complete the application upgrade.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
           END UPGRADE TO '1.1';
  2
Pluggable database altered.

SQL> SELECT app_name, app_version, app_status
     FROM   dba_applications
     WHERE  app_name = 'TOYS_APP';
  2   3
APP_NAME       APP_VERSION  APP_STATUS
-------------- ------------ ------------
TOYS_APP       1.1          NORMAL

SQL> SHOW pdbs
```

```

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 READ WRITE NO
         6 ROBOTS                         READ WRITE NO
         7 DOLLS                          READ WRITE NO
SQL>
```

Q1/ What do you observe I you connect to the CDB root?

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB_ORCL                       READ WRITE NO
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 READ WRITE NO
         6 ROBOTS                         READ WRITE NO
         7 DOLLS                          READ WRITE NO
        11 F2719737412_3_1                READ ONLY  NO

SQL>
```

A1/ There is a new PDB, `F2719737412_3_1`. All PDBs are opened in READ WRITE mode except the application root clone. Application root clones are created when an application is upgraded or uninstalled in an application root.

They are meant primarily for the purpose of metadata lookup. When an application PDB needs to look up metadata of common objects after the application root has upgraded but before the application PDB has upgraded to the same version as the application root, the older definitions of common objects needed by the application PDB can be found in the application root clone.

Q2/ Is there only one application root clone for all application containers?

A2/ No. There are as many application root clones as applications upgraded or uninstalled in an application container. Consider two applications installed in `toys_root`. The first application was upgraded twice, so there would be two application root clones, and the second application was upgraded once, so there would be one more application root clone. Hence there would be three application root clones for the application container.

6. Test that both application PDBs share the shared `toys_owner.new_tab` table.

```
SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
SQL> DESC toys_owner.new_tab
```

```
ERROR:
ORA-04043: object toys_owner.new_tab does not exist

SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
SQL> DESC toys_owner.new_tab
ERROR:
ORA-04043: object toys_owner.new_tab does not exist

SQL>
```

Q/ Why is the common user recognized but not the shared table?

A/ The application PDBs had been synchronized with the application root after the application installation when the common user was created, but they have not been resynchronized after the application upgrade when the new shared table was created.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT app_name, app_version, app_status, con_id
     FROM   cdb_applications
     WHERE  app_name = 'TOYS_APP';
  2    3
APP_NAME       APP_VERSION  APP_STATUS   CON_ID
-------------- ------------ ------------ ------
TOYS_APP       1.1          NORMAL            4
TOYS_APP       1.0          NORMAL            7
TOYS_APP       1.0          NORMAL            6

SQL>
```

Observe that both application PDBs are still not at the application version level of the application root.

```
SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.
```

```
SQL> DESC toys_owner.new_tab
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
 COL1                                               NUMBER(4)
 COL2                                               NUMBER

SQL> CONNECT system@dolls
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> DESC toys_owner.new_tab
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
 COL1                                               NUMBER(4)
 COL2                                               NUMBER

SQL> EXIT
$
```

Practice 2-3: Opening and Closing Application PDBs

Overview

In this practice, you will open and close application PDBs in application containers.

Tasks

1. Before starting the practice, execute the **$HOME/labs/APP/setup_hr_app.sh** and **$HOME/labs/APP/setup_app.sh** shell scripts. The first one recreates the `hr_root` application container and the second one closes the `hr_root` application container.

```
$ $HOME/labs/APP/setup_hr_app.sh
…
$ $HOME/labs/APP/setup_app.sh
…
$
```

2. Today, the `HR_APP` application should only be available through the `operations` application PDB and not through the `research` application PDB. Open the `operations` application PDB.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE operations OPEN;
ALTER PLUGGABLE DATABASE operations OPEN
*
ERROR at line 1:
ORA-65238: operation cannot be performed when the application
root is not open

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB_ORCL                       READ WRITE NO
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 READ WRITE NO
         6 ROBOTS                         READ WRITE NO
         7 DOLLS                          READ WRITE NO
         8 HR_ROOT                        MOUNTED
         9 OPERATIONS                     MOUNTED
        10 RESEARCH                       MOUNTED
        11 F2719737412_3_1                READ ONLY  NO
```

```
SQL>
```

Q/ Which other behavior is this one comparable to?

A/ An application PDB cannot be opened when the application root is not opened like regular PDBs cannot be opened when the CDB root is not opened.

```
SQL> ALTER PLUGGABLE DATABASE hr_root OPEN;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE operations OPEN;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB_ORCL                       READ WRITE NO
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 READ WRITE NO
         6 ROBOTS                         READ WRITE NO
         7 DOLLS                          READ WRITE NO
         8 HR_ROOT                        READ WRITE NO
         9 OPERATIONS                     READ WRITE NO
        10 RESEARCH                       MOUNTED
        11 F2719737412_3_1                READ ONLY  NO
SQL>
```

3. Open all application PDBs of the `hr_app` application.

```
SQL> SELECT name, con_id, application_root "APP_ROOT",
            application_seed "APP_Seed",
            application_pdb "APP_PDB",
            application_root_con_id "APP_ROOT_CONID"
     FROM   v$containers
     WHERE  application_root = 'YES';
  2    3    4    5    6

NAME                CON_ID APP_ROOT APP_Seed APP_PDB  APP_ROOT_CONID
--------------- ------ -------- -------- -------- -------------
TOYS_ROOT                4 YES      NO       NO
HR_ROOT                  8 YES      NO       NO
F2719737412_3_1         11 YES      NO       YES                  4

```

```
SQL> SELECT name, con_id, application_root "APP_ROOT",
            application_seed "APP_Seed",
            application_pdb "APP_PDB",
            application_root_con_id "APP_ROOT_CONID"
     FROM   v$containers
     WHERE  application_root_con_id = 8;
  2    3    4    5    6
NAME            CON_ID APP_ROOT APP_Seed APP_PDB  APP_ROOT_CONID
--------------- ------ -------- -------- -------- -------------
OPERATIONS           9 NO       NO       YES                  8
RESEARCH            10 NO       NO       YES                  8

SQL> ALTER PLUGGABLE DATABASE research OPEN;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB_ORCL                       READ WRITE NO
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 READ WRITE NO
         6 ROBOTS                         READ WRITE NO
         7 DOLLS                          READ WRITE NO
         8 HR_ROOT                        READ WRITE NO
         9 OPERATIONS                     READ WRITE NO
        10 RESEARCH                       READ WRITE NO
        11 F2719737412_3_1                READ ONLY  NO
SQL>
```

Q1/ What happens if you close the application root?

```
SQL> ALTER PLUGGABLE DATABASE hr_root CLOSE;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB_ORCL                       READ WRITE NO
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 READ WRITE NO
         6 ROBOTS                         READ WRITE NO
         7 DOLLS                          READ WRITE NO
         8 HR_ROOT                        MOUNTED
         9 OPERATIONS                     MOUNTED
        10 RESEARCH                       MOUNTED
        11 F2719737412_3_1                READ ONLY  NO
SQL>
```

A1/ All application PDBs associated to the application root are automatically closed.

Q2/ Is the behavior similar when you open the application root?

```
SQL> ALTER PLUGGABLE DATABASE hr_root OPEN;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB_ORCL                       READ WRITE NO
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 READ WRITE NO
         6 ROBOTS                         READ WRITE NO
         7 DOLLS                          READ WRITE NO
         8 HR_ROOT                        READ WRITE NO
         9 OPERATIONS                     MOUNTED
        10 RESEARCH                       MOUNTED
        11 F2719737412_3_1                READ ONLY  NO
SQL>
```

A2/ All application PDBs associated to the application root are not automatically opened. If you want them to be automatically opened after the application root is opened, preserve the application PDB's open mode across application root reopening. After reopening an application root, the PDBs are by default kept in mounted mode. If you want the PDBs to automatically open whenever the application root reopens, use the `SAVE STATE` clause of the `ALTER PLUGGABLE DATABASE` command.

```
SQL> CONNECT SYS@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE all OPEN;

```

```
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE all SAVE STATE;

Pluggable database altered.

SQL> SELECT con_name, state FROM cdb_pdb_saved_states;

CON_NAME       STATE
-------------- --------------
OPERATIONS     OPEN
HR_ROOT        OPEN
RESEARCH       OPEN

SQL>
```

4. Close the application root of the `HR_APP` application and reopen it. Verify that all application PDBs of the `HR_APP` application are automatically opened.

```
SQL> ALTER PLUGGABLE DATABASE hr_root CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE hr_root OPEN;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         8 HR_ROOT                        READ WRITE NO
         9 OPERATIONS                     READ WRITE NO
        10 RESEARCH                       READ WRITE NO
SQL> EXIT
$
```

5. Clean up application containers by executing the `$HOME/labs/APP/cleanup_apps.sh` script.

```
$ $HOME/labs/APP/cleanup_apps.sh
…
$
```

Practices for Lesson 3: Security in CDB, Application Containers, and PDBs

Chapter 3

Practices for Lesson 3: Overview

Practices Overview

In this practice, you will manage the users, privileges, and roles in application roots and application PDBs, and you will create shared and local tables in application containers. Security also includes CDB and PDB level auditing, encryption, and protection with Database Vault common realms and command rules.

Practice 3-1: Managing Common Entities and Shared Data in Application Containers

Overview

In this practice, you will manage common and local users, common and local roles, and granted privileges in application containers.

Tasks

1. Before starting the practice, execute the `$HOME/labs/admin/glogin_3.sh` and `$HOME/labs/APP/setup_toys_app.sh` shell scripts. The first one sets formatting for all columns selected in queries and the second one recreates the `toys_root` application container.

```
$ $HOME/labs/admin/glogin_3.sh
$ $HOME/labs/APP/setup_toys_app.sh
…
$
```

2. Before managing application common users in application containers, you check common and local users in regular PDBs.
   a. Verify that `SYSTEM` is a common user in the CDB.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> SELECT name, con_id FROM v$containers ORDER BY 2;

NAME                 CON_ID
-------------------- ------
CDB$ROOT                  1
PDB$SEED                  2
PDB_ORCL                  3
TOYS_ROOT                 4
TOYS_ROOT$SEED            5
ROBOTS                    6
DOLLS                     7

7 rows selected.

SQL> SELECT username, common, con_id FROM cdb_users
     WHERE  username = 'SYSTEM' ORDER BY con_id;
  2
USERNAME               COMMON CON_ID
---------------------- ------ ------
```

```
SYSTEM                 YES       1
SYSTEM                 YES       3
SYSTEM                 YES       4
SYSTEM                 YES       6
SYSTEM                 YES       7

SQL>
```

Q1/ Does the `SYSTEM` common user exist only in the CDB root?

A1/ There are as many `SYSTEM` users replicated as containers because this is a common user.

Q2/ Why then are there only five `SYSTEM` users whereas there are seven containers?

A2/ There are as many `SYSTEM` users listed as containers opened except the CDB seed (`PDB$SEED`) and the application seeds (`toys_root$seed`).

b. List the local users and the respective PDB where they exist.

```
SQL> SELECT username, common, name, u.con_id cid
     FROM   cdb_users u, v$containers c
     WHERE  common = 'NO' AND  u.con_id = c.con_id
     ORDER BY 4;
  2    3    4
USERNAME               COMMON NAME                  CID
---------------------- ------ -------------------- ----
PDBADMIN               NO     PDB_ORCL                3
ADMIN                  NO     TOYS_ROOT               4
ADMIN                  NO     ROBOTS                  6
ADMIN                  NO     DOLLS                   7

SQL>
```

c. Create a common user `C##_USER` in the CDB root.

```
SQL> CREATE USER c##_user IDENTIFIED BY oracle_4U
                 CONTAINER=ALL;
  2
User created.

SQL>
```

d. Verify that the new common user has been replicated in each PDB, regular and application PDBs.

```
SQL> SELECT username, common, name, u.con_id cid
     FROM   cdb_users u, v$containers c
     WHERE  username = 'C##_USER' AND  u.con_id = c.con_id
     ORDER BY 4;
  2    3    4
USERNAME               COMMON NAME                  CID
```

```
---------------------- ------ -------------------- ----
C##_USER               YES    CDB$ROOT                1
C##_USER               YES    PDB_ORCL                3
C##_USER               YES    TOYS_ROOT               4
C##_USER               YES    ROBOTS                  6
C##_USER               YES    DOLLS                   7

SQL>
```

e. Create a local user `l_user` in the regular `pdb_orcl` and then in the application PDB `robots`.

```
SQL> CREATE USER l_user IDENTIFIED BY oracle_4U;
CREATE USER l_user IDENTIFIED BY oracle_4U
            *
ERROR at line 1:
ORA-65096: invalid common user or role name

SQL> CREATE USER l_user IDENTIFIED BY oracle_4U
                 CONTAINER=current;
  2  CREATE USER l_user IDENTIFIED BY oracle_4U
*
ERROR at line 1:
ORA-65049: Creation of local user or role is not allowed in this
container.

SQL>
```

Q/ What do the error messages attempt to say?

A/ The first error message means that Oracle expected a common user to be created because you are still connected to the CDB root and that the user name does not match the convention, which stipulates starting with `C##`. The second error message is explicit.

```
SQL> SHOW parameter common_user_prefix

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
common_user_prefix                   string      C##

SQL>
```

f. Connect to the regular `pdb_orcl` to create the local user.

```
SQL> CONNECT system@pdb_orcl
Enter password: ******
Connected.
SQL> CREATE USER l_user IDENTIFIED BY oracle_4U
            QUOTA 1M ON users;
```

```
   2
User created.

SQL> SELECT username, common, name, u.con_id cid
     FROM   cdb_users u, v$containers c
     WHERE  common = 'NO' AND  u.con_id = c.con_id
     ORDER BY 4;
  2    3    4
USERNAME                COMMON NAME                  CID
----------------------- ------ -------------------- ----
PDBADMIN                NO     PDB_ORCL                3
L_USER                  NO     PDB_ORCL                3

SQL>
```

g. Connect to the application PDB `robots` to create the local user.

```
SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> CREATE USER l_user IDENTIFIED BY oracle_4U;

User created.

SQL> SELECT username, common, name, u.con_id cid
     FROM   cdb_users u, v$containers c
     WHERE  common = 'NO' AND  u.con_id = c.con_id
     ORDER BY 4;
  2    3    4

USERNAME                COMMON NAME                  CID
----------------------- ------ -------------------- ----
ADMIN                   NO     ROBOTS                  6
L_USER                  NO     ROBOTS                  6

SQL>
```

Q/ Why doesn't the list display all local users of all PDBs?

A/ You are not connected to the CDB root.

h. How can you verify that `l_user` can be distinctly referenced in `pdb_orcl` and in `robots`?

```
SQL> CONNECT system@pdb_orcl
Enter password: ******
Connected.
SQL> GRANT create session TO l_user;
```

```

Grant succeeded.

SQL>
```

1) Connect to `pdb_orcl` as `l_user`.

```
SQL> CONNECT l_user@pdb_orcl
Enter password: ******
Connected.
SQL> SHOW con_name

CON_NAME
------------------------------
PDB_ORCL
SQL> SHOW user
USER is "L_USER"
SQL>
```

2) Connect to `robots` as `l_user`.

```
SQL> CONNECT l_user@robots
Enter password: ******
ERROR:
ORA-01045: user L_USER lacks CREATE SESSION privilege; logon
denied

SQL>
```

3) Grant the appropriate privilege to the `l_user` user in `robots`.

```
SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> GRANT create session TO l_user;

Grant succeeded.

SQL> CONNECT l_user@robots
Enter password: ******
Connected.
SQL> SHOW con_name

CON_NAME
------------------------------
ROBOTS
SQL> SHOW user
USER is "L_USER"
```

```
SQL>
```

4) Attempt to connect as `l_user` in `dolls`, the other application PDB of the `toys_root` application container.

```
SQL> CONNECT l_user@dolls
Enter password: ******
ERROR:
ORA-01017: invalid username/password; logon denied

Warning: You are no longer connected to ORACLE.
SQL>
```

Q/ Why does the connection fail?

A/ It fails because `l_user` does not exist in the `dolls` application PDB***.***

i. Provide an overview of common and local users from `pdb_orcl` and `robots`.

```
SQL> CONNECT system@pdb_orcl
Enter password: ******
Connected.
SQL> SELECT username, common, con_id
     FROM   cdb_users ORDER BY 2;
```

```
  2

USERNAME                COMMON CON_ID
----------------------- ------ ------
…
L_USER                  NO          3
PDBADMIN                NO          3
……
SYS                     YES         3
SYSTEM                  YES         3

48 rows selected.

SQL>
```

Note: The number of rows may differ according to the features and options installed.

```
SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> SELECT username, common, con_id
     FROM   cdb_users ORDER BY 2;
  2

USERNAME                COMMON CON_ID
```

```
--------------------- ------ ------
L_USER                NO          6
ADMIN                 NO          6
XS$NULL               YES         6
GSMCATUSER            YES         6…
…
SYS                   YES         6
SYSTEM                YES         6

43 rows selected.

SQL>
```

3. You learned that the same concept of commonality exists for users at the application container level. You will now manage the creation of application common users within application containers.
   Creating application common users within the `toys_app` application in the `toys_root` application container means that the application common users will be replicated in all containers of the `toys_root` application container, namely the `toys_root` application root, the application seed, and the two application PDBs, `robots` and `dolls`.
   a. Before creating application common users in the `toys_root` application root, check whether there are existing application common users at the `toys_root` application root level.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT username, common, con_id
     FROM   cdb_users
     WHERE  username IN ('C##_USER','TOYS_OWNER','L_USER')
     ORDER BY 2, 3;
```

```
  2    3    4
USERNAME               COMMON CON_ID
---------------------- ------ ------
L_USER                 NO          6
C##_USER               YES         4
TOYS_OWNER             YES         4
C##_USER               YES         6
TOYS_OWNER             YES         6
C##_USER               YES         7
TOYS_OWNER             YES         7

7 rows selected.

SQL>
```

Q1/ Can the `toys_owner` user be a common user at the CDB level?

A1/ No. Any user created as common at the CDB root level must be created with the predefined prefix `c##`.

Q2/ Which column describes whether the existence of an object is inherited from the CDB root or from an application root?

Find the new column in the `CDB_USERS` view.

```
SQL> DESC cdb_users
 Name                                  Null?    Type
 ---------------------------------- -------- ------------------
 USERNAME                              NOT NULL VARCHAR2(128)
 USER_ID                               NOT NULL NUMBER
 PASSWORD                                       VARCHAR2(4000)
 ACCOUNT_STATUS                        NOT NULL VARCHAR2(32)
 LOCK_DATE                                      DATE
 EXPIRY_DATE                                    DATE
 DEFAULT_TABLESPACE                    NOT NULL VARCHAR2(30)
 TEMPORARY_TABLESPACE                  NOT NULL VARCHAR2(30)
 LOCAL_TEMP_TABLESPACE                          VARCHAR2(30)
 CREATED                               NOT NULL DATE
 PROFILE                               NOT NULL VARCHAR2(128)
 INITIAL_RSRC_CONSUMER_GROUP                    VARCHAR2(128)
 EXTERNAL_NAME                                  VARCHAR2(4000)
 PASSWORD_VERSIONS                              VARCHAR2(17)
 EDITIONS_ENABLED                               VARCHAR2(1)
 AUTHENTICATION_TYPE                            VARCHAR2(8)
 PROXY_ONLY_CONNECT                             VARCHAR2(1)
 COMMON                                         VARCHAR2(3)
 LAST_LOGIN                                            TIMESTAMP(9) WITH
TIME
                                                       ZONE
 ORACLE_MAINTAINED                                     VARCHAR2(1)
 INHERITED                                             VARCHAR2(3)
 DEFAULT_COLLATION                                     VARCHAR2(100)
 IMPLICIT                                              VARCHAR2(3)
 ALL_SHARD                                             VARCHAR2(3)
 CON_ID                                                NUMBER

SQL>
```

A2/ The `COMMON` column value defines if the object is a common object.
A common object can be common to all PDBs when created from the CDB root, or common to all application PDBs within an application container when created from the application root.

The `INHERITED` column describes whether an object (like a user) is created by replication from the CDB root or from the application root to which it belongs.
A `YES` value means that the common object (`COMMON = YES`) was created by replication from the CDB root, which is the case for the `SYSTEM` and `c##_user` users.
A `NO` value means that the common object was created in the application root and could be replicated in application PDBs, which is the case for the `toys_owner` user.

```
SQL> SELECT username, common, inherited, con_id
     FROM   cdb_users
     WHERE  username IN ('TOYS_OWNER','L_USER')
     ORDER BY 2, 3;
  2    3    4
USERNAME                COMMON INH CON_ID
----------------------- ------ --- ------
L_USER                  NO     NO       6
TOYS_OWNER              YES    NO       4
TOYS_OWNER              YES    YES      7
TOYS_OWNER              YES    YES      6

SQL>
```

Q3/ Why does the application common `toys_owner` user in application PDBs (`robots` and `dolls`) show with an `INHERITED` column value of YES?

A3/ The `toys_owner` application common user has been created in the application PDBs (`robots` and `dolls`) by replication from the `toys_root` application root to which they belong.

b. Create a common user `toys_test` in the `toys_root` application root.

```
SQL> SHOW con_name

CON_NAME
------------------------------
TOYS_ROOT
SQL> CREATE USER toys_test IDENTIFIED BY oracle_4U
                CONTAINER = ALL;

User created.

SQL> SELECT username, common, inherited, con_id
     FROM   cdb_users
     WHERE  username like 'TOYS_TEST%' ORDER BY 2, 3;
  2    3
USERNAME               COMMON INH CON_ID
---------------------- ------ --- ------
TOYS_TEST              YES    NO       4

SQL>
```

c. You forgot that the creation of application common entities within an application container requires you to first upgrade the application existing in the application root. You will observe the difference between a user created commonly in an application root outside an application and within an application. You will now create an application common user `toys_test2` in the `toys_root` application root in the `toys_app` application.

```
SQL> SELECT app_name, app_version, app_status
     FROM   dba_applications
     WHERE  app_name NOT LIKE 'APP%' ;
  2    3
APP_NAME   APP_VERS APP_STATUS
---------- -------- ------------
TOYS_APP   1.0      NORMAL

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                    BEGIN UPGRADE '1.0' TO '1.1';
 2
Pluggable database altered.

SQL> CREATE USER toys_test2 IDENTIFIED BY oracle_4U
                CONTAINER = ALL;
  2
User created.
```

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                    END UPGRADE TO '1.1';
 2
Pluggable database altered.

SQL> SELECT username, common, inherited, con_id
     FROM   cdb_users
     WHERE  username like 'TOYS_TEST%' ORDER BY 2, 3;
  2    3
USERNAME               COMMON INH CON_ID
---------------------- ------ --- ------
TOYS_TEST              YES    NO       4
TOYS_TEST2             YES    NO       4

SQL>
```

Q/ Why are `toys_test` and `toys_test2` created only in `toys_root`, the application root container, and not in application PDBs?

A/ The application PDBs need to be synchronized with the application root before the `toys_test` and `toys_test2` users are replicated in the application PDBs.

d. Before completing the application common users creation in application PDBs, test whether the `toys_test2` user can connect to the application root and then to the application PDBs.

   1) Connect as `toys_test2` to `toys_root`, then `robots`, and finally to `dolls`.

```
SQL> CONNECT toys_test2@toys_root
Enter password:
ERROR:
ORA-01045: user TOYS_TEST2 lacks CREATE SESSION privilege; logon
denied


Warning: You are no longer connected to ORACLE.
SQL> CONNECT toys_test2@robots
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied

SQL> CONNECT toys_test2@dolls
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied

SQL>
```

Q/ Why is `toys_test2` recognized to connect to the application root and not to the application PDBs?

A/ The application PDBs are still not synchronized with the application root.

2) Synchronize the application PDBs.

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL>
```

3) Connect as `toys_test2` and then as `toys_test` to `robots` and to `dolls`.

```
SQL> CONNECT toys_test2@robots
Enter password:
ERROR:
ORA-01045: user TOYS_TEST2 lacks CREATE SESSION privilege; logon
denied

SQL> CONNECT toys_test2@dolls
Enter password:
ERROR:
ORA-01045: user TOYS_TEST2 lacks CREATE SESSION privilege; logon
denied

SQL> CONNECT toys_test@robots
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied

SQL> CONNECT toys_test@dolls
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied
```

```
SQL>
```

Q/ Why is `toys_test2` recognized to connect to the application PDBs, whereas `toys_test` is not?

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT username, common, inherited, con_id
     FROM   cdb_users
     WHERE  username like 'TOYS_TEST%' ORDER BY 2, 3;
  2    3
USERNAME               COMMON INH CON_ID
---------------------- ------ --- ------
TOYS_TEST              YES    NO       4
TOYS_TEST2             YES    NO       4
TOYS_TEST2             YES    YES      7
TOYS_TEST2             YES    YES      6

SQL>
```

A/ `toys_test` cannot be synchronized with the application root because it has not been created within the `toys_app` application. It does not show in `CDB_USERS` for the application PDBs.
`toys_test2` is synchronized with the application root because it has been created within the `toys_app` application. The `NO` value of the `INHERITED` column means that the application common `toys_test2` users (`COMMON = YES`) were created as common in the `toys_app` application root (`CON_ID=4`), and not by replication from the CDB root. The `YES` value for the same users in `robots` and `dolls` means that the same common user `toys_test2` was created in the application PDBs by replication from the `toys_app` application root.

4. You learned that the same concept of commonality exists for roles at the application container level. You will now manage the creation of application common roles and privileges granted within application containers.
   Creating application common roles within the `toys_app` application in the `toys_root` application container means that the application common roles will be replicated in all containers of the `toys_root` application container, namely the `toys_root` application root, the application seed, and the two application PDBs, `robots` and `dolls`.
   a. Before creating application common roles in the `toys_root` application root, check whether there are application common roles at the `toys_root` application root level. Just like `CDB_USERS`, `CDB_ROLES` now has a new `INHERITED` column.

```
SQL> SELECT role, common, inherited, con_id FROM cdb_roles
     WHERE  role IN ('DBA' ,'CONNECT') ORDER BY 4;
  2
ROLE                           COMMON INH CON_ID
------------------------------ ------ --- ------
DBA                            YES    YES      4
```

```
CONNECT                         YES    YES       4
DBA                             YES    YES       6
CONNECT                         YES    YES       6
DBA                             YES    YES       7
CONNECT                         YES    YES       7

6 rows selected.

SQL>
```

Observe that all predefined common roles are inherited from the CDB root level.

b. You will now create two application common roles, `mgr_role` and `emp_role`, within the `toys_root` application container so that the application common roles within the `toys_app` application in the application container will be replicated in all containers of the `toys_root` application container, namely the `toys_root` application root, the application seed, and the two application PDBs, `robots` and `dolls`.

c. Create the `mgr_role` role and `emp_role` role as common in the application container.

```
SQL> SELECT app_name, app_version, app_status
     FROM   dba_applications WHERE app_name NOT LIKE 'APP$%';
  2
APP_NAME   APP_VERS APP_STATUS
---------- -------- ------------
TOYS_APP   1.1      NORMAL

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                     BEGIN UPGRADE '1.1' TO '1.2';
 2
Pluggable database altered.

SQL> CREATE ROLE mgr_role CONTAINER = ALL;

Role created.

SQL> CREATE ROLE emp_role CONTAINER = ALL;

Role created.

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                     END UPGRADE TO '1.2';
 2
Pluggable database altered.

SQL> SELECT role, common, inherited, con_id FROM cdb_roles
```

```
     WHERE  role IN ('DBA','MGR_ROLE','EMP_ROLE') ORDER BY 1,4;
  2
ROLE                           COMMON INH CON_ID
------------------------------ ------ --- ------
DBA                            YES    YES      4
DBA                            YES    YES      6
DBA                            YES    YES      7
EMP_ROLE                       YES    NO       4
MGR_ROLE                       YES    NO       4

SQL>
```

Q/ Why are `mgr_role` and `emp_role` created only in `toys_root`, the application root container?

A/ The application PDBs need to be synchronized with the application root before the `mgr_role` and `emp_role` roles are replicated in the application PDBs.

d. Synchronize the application PDBs.

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT role, common, inherited, con_id FROM cdb_roles
     WHERE  role IN ('MGR_ROLE','EMP_ROLE') ORDER BY 1,4;
  2
ROLE                           COMMON INH CON_ID
------------------------------ ------ --- ------
EMP_ROLE                       YES    NO       4
EMP_ROLE                       YES    YES      6
EMP_ROLE                       YES    YES      7
MGR_ROLE                       YES    NO       4
MGR_ROLE                      YES    YES      6
MGR_ROLE                      YES    YES      7

6 rows selected.

SQL>
```

Q1/ Now that an application common user and application common roles are created in the application root, would these common entities be automatically replicated in a brand new application PDB that is created?

A1/ Yes, if the application seed is synchronized first.

Q2/ How can you determine if the application seed is synchronized with the application root?

A2/ Connect to the application seed to display the `CDB_APPLICATIONS` view. The `app_capture_service` column names the application PDB from which the synchronization was completed. The application seed referenced in both `app_capture_service` and `con_id` columns means that the PDB was synchronized manually.

```
SQL> CONNECT sys@toys_root$seed AS SYSDBA
Enter password: ******
Connected.
SQL> COL app_name FORMAT A8
SQL> SELECT app_name, app_version APP_Vers, app_status Status,
            app_capture_service, a.con_id, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  app_name NOT LIKE 'APP$%'
     AND    a.con_id = p.pdb_id;
  2    3    4    5
APP_NAME   APP_VERS STATUS   APP_CAPTURE_SE CON_ID PDB_NAME
---------- -------- -------- -------------- ------ ------------
TOYS_APP   1.0      NORMAL   toys_root$seed      5
TOYS_ROOT$SEED

SQL>
```

e. Synchronize the application seed.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;
ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC
*
ERROR at line 1:
ORA-16000: database or pluggable database open for read-only
access

SQL> ALTER PLUGGABLE DATABASE CLOSE;

```

```
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN READ ONLY;

Pluggable database altered.
SQL> SELECT app_name, app_version APP_Vers, app_status Status,
            app_capture_service, a.con_id, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  app_name NOT LIKE 'APP$%'
     AND    a.con_id = p.pdb_id;
  2    3    4    5
APP_NAME   APP_VERS STATUS   APP_CAPTURE_SE CON_ID PDB_NAME
---------- -------- -------- -------------- ------ ------------
TOYS_APP   1.2      NORMAL   toys_root$seed      5
TOYS_ROOT$SEED

SQL>
```

f. Create the new `doodles` application PDB.

```
SQL> !mkdir  /u02/app/oracle/oradata/ORCL/toys_root/doodles
SQL>
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SESSION SET db_create_file_dest =
           '/u02/app/oracle/oradata/ORCL/toys_root/doodles';
  2
System altered.

SQL> CREATE PLUGGABLE DATABASE doodles
            ADMIN USER admin IDENTIFIED BY oracle_4U;
  2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE doodles OPEN;

Pluggable database altered.

SQL>
```

g. Check whether the application common users and application common roles are automatically replicated.

```
SQL> CONNECT system@doodles
Enter password: ******
Connected.
SQL> SELECT username, common, inherited, con_id
     FROM   cdb_users
     WHERE  username like 'TOYS_TEST%' ORDER BY 2, 3;
  2    3
USERNAME               COMMON INH CON_ID
---------------------- ------ --- ------
TOYS_TEST2             YES    YES     10

SQL> SELECT role, common, inherited, con_id FROM cdb_roles
     WHERE  role IN ('MGR_ROLE','EMP_ROLE') ORDER BY 1,4;
  2
ROLE                           COMMON INH CON_ID
------------------------------ ------ --- ------
EMP_ROLE                       YES    YES     10
MGR_ROLE                       YES    YES     10

SQL>
```

Q/ How do you know that the new application PDB has been automatically synchronized with the application root?

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: *******
Connected.
SQL> SELECT app_name, app_version, app_status,
            app_capture_service, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  app_name NOT LIKE 'APP$%'
     AND    a.con_id = p.pdb_id;
  2    3    4    5

APP_NAME   APP_VERS APP_STATUS   APP_CAPTURE_SE PDB_NAME
---------- -------- ------------ -------------- --------------
TOYS_APP   1.2      NORMAL       toys_root      TOYS_ROOT
TOYS_APP   1.2      NORMAL       toys_root$seed DOODLES
TOYS_APP   1.2      NORMAL       dolls          DOLLS
TOYS_APP   1.2      NORMAL       robots         ROBOTS

SQL>
```

A/ The `app_capture_service` column names the application PDB from which the synchronization could be completed. `robots` and `dolls` were manually synchronized whereas `doodles` was synchronized automatically because it was created from the seed application that was manually synchronized.

5. Create tables common to all application PDBs within the `toys_app` application in the `toys_root` application container.
   a. Check whether there are application common tables owned by `toys_owner` in the `toys_app` application in the `toys_root` application container.

```
SQL> SELECT owner, object_name, sharing, application
     FROM   dba_objects
     WHERE  owner = 'TOYS_OWNER';
  2    3
OWNER          OBJECT_NAM SHARING       A
-------------- ---------- ------------- -
TOYS_OWNER     SALES_DATA METADATA LINK Y
TOYS_OWNER     CODES      DATA LINK     Y

SQL>
```

Q1/ How can you differentiate application common tables from Oracle-supplied common tables?

A1/ The new `application` column defines whether the object is an application-defined object or an Oracle-supplied object.

```
SQL> SELECT owner, object_name, sharing, application
     FROM   dba_objects
     WHERE  object_name= 'TAB$';
  2    3
OWNER          OBJECT_NAM SHARING       A
-------------- ---------- ------------- -
SYS            TAB$       METADATA LINK N

SQL>
```

Q2/ What is the difference between metadata-linked and data-linked common objects?

A2/ All references to metadata-linked objects get resolved in the context of the container in which a reference is made, unlike data-linked objects for which all references get treated as if they were made in the context of the application root. Referenced tables and dimensions tables are good candidates for data-linked tables, whereas referencing tables and fact tables are better candidates for metadata-linked tables.

b. You will create a metadata-linked common table and grant object privileges on the table to `emp_role`.

1) First begin the application upgrade.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                    BEGIN UPGRADE '1.2' TO '1.3';
 2
Pluggable database altered.

SQL>
```

2) Create a metadata-linked common table and grant object privileges on the table to `emp_role`, the role that you grant to the common user `toys_test2`.

```
SQL> ALTER SESSION SET default_sharing = metadata;

Session altered.

SQL> CREATE TABLE toys_owner.tab1 (c1 number, c2 number);

Table created.

SQL> SELECT owner, object_name, sharing, application
```

```
     FROM   dba_objects
     WHERE  object_name = 'TAB1';
  2    3
OWNER          OBJECT_NAM SHARING       A
-------------- ---------- ------------- -
TOYS_OWNER     TAB1       METADATA LINK Y

SQL> INSERT INTO toys_owner.tab1 VALUES (1,1);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> GRANT SELECT, INSERT on toys_owner.tab1
                   TO emp_role, mgr_role CONTAINER=ALL;
  2
Grant succeeded.

SQL> GRANT emp_role, create session TO toys_test2
                     CONTAINER=ALL;
     2
Grant succeeded.

SQL>
```

Note: You can either set your session to the type of common table you want to create or use the `SHARING` clause in the CREATE TABLE statement.

3) Complete the application upgrade.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                     END UPGRADE TO '1.3';
 2
Pluggable database altered.

SQL>
```

4) Verify now that the common metadata-linked table can be read from application PDBs within the application.

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> DESC toys_owner.tab1
ERROR:
```

```
ORA-04043: object toys_owner.tab1 does not exist

SQL>
```

Q1/ Why is the table not visible?

A1/ Application PDB synchronization has not been performed.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> DESC toys_owner.tab1
 Name                              Null?    Type
 --------------------------------- -------- -----------------
 C1                                         NUMBER
 C2                                         NUMBER

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT toys_test2@robots
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.tab1;

        C1         C2
---------- ----------
         1          1

SQL> INSERT INTO toys_owner.tab1 VALUES (3, 3);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM toys_owner.tab1;

        C1         C2
```

```
---------- ----------
         1          1
         3          3

SQL> CONNECT toys_test2@dolls
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.tab1;

        C1         C2
---------- ----------
         1          1

SQL> INSERT INTO toys_owner.tab1 VALUES (4, 4);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM toys_owner.tab1;

        C1         C2
---------- ----------
         1          1
         4          4

SQL>
```

Q2/ What is your conclusion about the rows displayed in either application PDB?

A2/ ***Because the table is a metadata-linked object, the table definition created in the application root is sharable in all application PDBs associated to the*** ***`toys_root`*** ***application root. Nevertheless data inserted, updated, or deleted during application install or upgrade is visible only to the application PDB where data has been manipulated.***

c. You will now create a data-linked common table and grant object privileges on the table to `emp_role`.

   1) First begin the application upgrade.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: *******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                        BEGIN UPGRADE '1.3' TO '1.4';
```

```
 2
Pluggable database altered.

SQL>
```

2) Create a data-linked common table and grant object privileges on the table to `emp_role`.

```
SQL> CREATE TABLE toys_owner.labels SHARING = OBJECT
                 (code number, label varchar2(10));
  2
Table created.

SQL> INSERT INTO toys_owner.labels VALUES (1,'Label1');

1 row created.

SQL> INSERT INTO toys_owner.labels VALUES (2,'Label2');

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT owner, object_name, sharing, application
     FROM   dba_objects
     WHERE  object_name IN ('TAB1', 'LABELS');
  2    3
OWNER          OBJECT_NAM SHARING       A
-------------- ---------- ------------- -
TOYS_OWNER     TAB1       METADATA LINK Y
TOYS_OWNER     LABELS     DATA LINK     Y

SQL> GRANT SELECT, INSERT ON toys_owner.labels TO emp_role
                             CONTAINER=ALL;
  2
Grant succeeded.

SQL>
```

3) Complete the application upgrade.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                      END UPGRADE TO '1.4';
 2
Pluggable database altered.
```

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL>
```

4) Verify now that the data-linked common table can be read from application PDBs within the application.

```
SQL> CONNECT toys_test2@robots
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.labels;

     CODE LABLE
---------- ----------
         1 Label1
         2 Label2

SQL> INSERT INTO toys_owner.labels VALUES (3, 'Label3');
INSERT INTO toys_owner.labels VALUES (3,'Label3')
                       *
ERROR at line 1:
ORA-65097: DML into a data link table is outside an application
action

SQL>
```

Q/ What data can be read but not inserted in the table?

A/ Because the table is a data-linked object, the table definition and data created in the application root is sharable in all application PDBs associated to the `toys_root` application root. Both definition and data reside in the application root.

```
SQL> CONNECT toys_test2@dolls
Enter password: ******
```

```
Connected.
SQL> SELECT * FROM toys_owner.labels;

      CODE LABLE
---------- ----------
         1 Label1
         2 Label2

SQL>
```

d. Create local tables and grant privileges locally to local users in `robots` and `dolls`.

```
SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> CREATE USER user1 IDENTIFIED BY oracle_4U
            QUOTA unlimited ON system;
  2
User created.

SQL> CREATE TABLE user1.tab_robots ( c number);

Table created.

SQL> INSERT INTO user1.tab_robots values (1);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> GRANT create session TO l_user;

Grant succeeded.

SQL> GRANT select, delete ON user1.tab_robots TO l_user;

Grant succeeded.

SQL> CONNECT system@dolls
Enter password: ******
Connected.
SQL> CREATE USER user1 IDENTIFIED BY oracle_4U
```

```
             QUOTA unlimited ON system;
  2
User created.

SQL> CREATE USER l_user IDENTIFIED BY oracle_4U;

User created.

SQL> CREATE TABLE user1.tab_dolls (c number);

Table created.

SQL> INSERT INTO user1.tab_dolls values (2);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> GRANT create session TO l_user;

Grant succeeded.

SQL> GRANT select, update ON user1.tab_dolls TO l_user;

Grant succeeded.

SQL>
```

```
SQL> CONNECT l_user@robots
Enter password: ******
Connected.
SQL> SELECT * FROM user1.tab_robots;

         C
----------
         1

SQL> DELETE FROM user1.tab_robots;

1 row deleted.
```

```
SQL> SELECT * FROM user1.tab_dolls;
SELECT * FROM user1.tab_dolls
                    *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> CONNECT l_user@dolls
Enter password: ******
Connected.
SQL> SELECT * FROM user1.tab_dolls;

         C
----------
         2

SQL> SELECT * FROM user1.tab_robots;
SELECT * FROM user1.tab_robots
                    *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> DELETE FROM user1.tab_dolls;
DELETE FROM user1.tab_dolls
                  *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> UPDATE user1.tab_dolls SET c=3;

1 row updated.

SQL>
```

Q/ Why didn't you complete these operations before the application upgrade completion?

A/ These are operations on local users and local tables. Only common entities created within the application root need to be replicated in the application PDBs and therefore require an application installation or upgrade or patch BEGIN/END block with further application PDBs synchronization.

6. Manage system administrative privileges such `SYSDBA` in application root and application PDBs.
   a. Within `toys_root`, grant the `SYSDBA` privilege to `toys_owner`.

```
SQL> CONNECT sys@toys_root AS SYSDBA
```

```
Enter password: *******
Connected.
SQL> SELECT username, sysdba, con_id FROM v$pwfile_users;

USERNAME               SYSDB CON_ID
---------------------- ----- ------
SYS                    TRUE       0

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
           BEGIN UPGRADE '1.4' TO '1.5';
  2
Pluggable database altered.

SQL> GRANT sysdba TO toys_owner CONTAINER = all;

Grant succeeded.

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
           END UPGRADE  TO '1.5';
  2
Pluggable database altered.

SQL> CONNECT toys_owner@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SHOW user
USER is "SYS"
SQL>
```

Q/ Is SYSDBA granted to the common application user granted in all application PDBs?

```
SQL> SELECT username, sysdba, common, con_id
     FROM v$pwfile_users;

USERNAME               SYSDB COMMON CON_ID
---------------------- ----- ------ ------
SYS                    TRUE  YES         0
TOYS_OWNER             TRUE  YES         4

SQL> CONNECT toys_owner@robots AS SYSDBA
Enter password: ******
ERROR:
ORA-01017: invalid username/password; logon denied
```

```
Warning: You are no longer connected to ORACLE.
SQL>
```

A/ Not yet. Synchronization needs to be performed IF granting SYSDBA to `toys_owner` is required in application PDBs. If you do not want to grant SYSDBA to the common user in all application PDBs, a better solution is to grant the privilege directly to the user within the PDB and not through an application upgrade.

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT username, sysdba, common, con_id
     FROM v$pwfile_users;
  2
USERNAME               SYSDB COMMON CON_ID
---------------------- ----- ------ ------
SYS                    TRUE  YES         0

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> SELECT username, sysdba, common, con_id
     FROM v$pwfile_users;
  2
USERNAME               SYSDB COMMON CON_ID
---------------------- ----- ------ ------
SYS                    TRUE  YES         0
TOYS_OWNER             TRUE  YES         6

SQL> CONNECT toys_owner@robots AS SYSDBA
Enter password: ******
Connected.
SQL> SHOW user
USER is "SYS"
SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT toys_owner@dolls AS SYSDBA
```

```
Enter password: ******
Connected.
SQL> SELECT username, sysdba, common, con_id
     FROM v$pwfile_users;
  2
USERNAME                SYSDB COMMON CON_ID
----------------------- ----- ------ ------
SYS                     TRUE  YES         0
TOYS_OWNER              TRUE  YES         7

SQL> EXIT
$
```

Practice 3-2: Querying Data Across Application PDBs in CDB

Overview

In this practice, you will configure application PDBs in `ORCL` so as to be able to query data from application PDBs within the same CDB using container maps.

Tasks

1. Container maps allow equi-partitioning of application data into PDBs based on value in a particular column. This column should be one of the most frequently occurring columns in the application queries.
   You will use the `scott.emp` and `scott.dept` tables and container maps to spread the data across four application PDBs in the `hr_root` application container. Then you will aggregate all the data spread across the four application PDBs by running the query from the application root.
   Before starting the practice, execute the `$HOME/labs/APP/setup3_hr_app.sh` shell script. The script sets up the `hr_root` application container with three application PDBs and the `hr_app` application containing the metadata-linked `scott.emp` and `scott.dept` tables.

```
$ $HOME/labs/APP/setup3_hr_app.sh
…
$
```

2. Use the `$HOME/labs/APP/script_insert.sql` SQL script to insert rows into the metadata-linked `scott.emp` and `scott.dept` tables.
   a. Insert different rows in the metadata-linked `scott.emp` and `scott.dept` tables in each PDB.

```
$ sqlplus / AS SYSDBA

SQL> @$HOME/labs/APP/script_insert.sql
…
SQL>
```

   b. Query data from the tables.

```
SQL> CONNECT scott@hr_root
Enter password: ******
Connected.
SQL> SELECT * FROM scott.emp;

no rows selected

SQL> CONNECT scott@sales
Enter password: ******
Connected.
SQL> SELECT deptno, dname FROM dept;

    DEPTNO DNAME
```

```
---------- --------------
        30 SALES

SQL> SELECT empno, ename, deptno FROM emp;

     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7499 ALLEN              30
      7521 WARD               30
      7654 MARTIN             30
      7698 BLAKE              30
      7844 TURNER             30
      7900 JAMES              30

6 rows selected.

SQL> CONNECT scott@accounting
Enter password: ******
Connected.
SQL> SELECT deptno, dname FROM dept;

    DEPTNO DNAME
---------- --------------
        10 ACCOUNTING

SQL> SELECT empno, ename, deptno FROM emp;

     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7782 CLARK              10
      7839 KING               10
      7934 MILLER             10

SQL>
```

Q1/ What is missing to aggregate all rows from the four partitions stored in the four PDBs when you are connected to the application root?

A1/ Use the `CONTAINERS` clause.

```
SQL> CONNECT scott@hr_root
Enter password: ******
Connected.
SQL> SELECT empno, ename, deptno FROM containers(emp);
```

```
     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7782 CLARK              10
      7839 KING               10
      7934 MILLER             10
      7369 SMITH              20
      7566 JONES              20
      7788 SCOTT              20
      7876 ADAMS              20
      7902 FORD               20
      7499 ALLEN              30
      7521 WARD               30
      7654 MARTIN             30
      7698 BLAKE              30
      7844 TURNER             30
      7900 JAMES              30

14 rows selected.

SQL> SELECT deptno, dname FROM containers(dept);

    DEPTNO DNAME
---------- --------------
        10 ACCOUNTING
        20 RESEARCH
        30 SALES

SQL>
```

Q2/ When data is segregated into separate application PDBs of the application container, as in the case of the `scott.emp` and `scott.dept` tables, how would you configure the application container so that a query would be appropriately routed to the relevant application PDB?

A2/ You have to create a container map definition. The container map is a table definition where each partition name corresponds to the PDB name. You only need one column in this table that you have decided to use for partitioning the data. Then set the database property `container_map` for the application container.

```
SQL> CREATE TABLE scott.maptable
      (deptno number, name varchar2(30))
      PARTITION BY LIST (deptno)
       ( PARTITION accounting VALUES (10),
         PARTITION research VALUES (20),
         PARTITION sales VALUES (30));
```

```
  2    3    4    5    6
Table created.

SQL> ALTER DATABASE SET container_map='scott.maptable';

Database altered.

SQL> SELECT container_map, container_map_object, table_name
     FROM   dba_tables WHERE table_name = 'MAPTABLE';
  2
CONTAINER_MAP CONTAINER_MAP_OBJECT TABLE_NAME
------------- -------------------- --------------------
NO            YES                  MAPTABLE

SQL>
```

```
SQL> SELECT * FROM emp where deptno = 10;

no rows selected

SQL>
```

Q3/ Why does the query again return 'no rows selected'?

A3/ You created a container map table, but the tables of the query are not enabled to collaborate with the container map table.

```
SQL> SELECT container_map, container_map_object, table_name
     FROM   dba_tables WHERE owner='SCOTT';
  2
CONTAINER_MAP CONTAINER_MAP_OBJECT TABLE_NAME
------------- -------------------- --------------------
NO            NO                   DEPT
NO            NO                   EMP
NO            YES                  MAPTABLE

SQL>
```

```
SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app
              BEGIN UPGRADE '1.0' TO '1.1';
  2
Pluggable database altered.
```

```
SQL> ALTER TABLE scott.dept ENABLE container_map;

Table altered.

SQL> ALTER TABLE scott.emp ENABLE container_map;

Table altered.

SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app
              END UPGRADE TO '1.1';
  2
Pluggable database altered.

SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@research AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@accounting AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT container_map, container_map_object, table_name
     FROM   dba_tables WHERE owner='SCOTT';
  2
```

```
CONTAINER_MAP CONTAINER_MAP_OBJECT TABLE_NAME
------------- -------------------- --------------------
YES           NO                   DEPT
YES           NO                   EMP
NO            YES                  MAPTABLE

SQL> SELECT * FROM scott.emp where deptno = 10;

no rows selected

SQL>
```

Q4/ Why does the query still not return any rows?

A4/ You enabled `scott.emp` and `scott.dept` tables to collaborate with the container map table but did not enable them to be used without the `CONTAINERS` clause.

```
SQL> SELECT containers_default, container_map,
            container_map_object, table_name "TABLE"
     FROM   dba_tables WHERE owner='SCOTT';
  2    3
CONTAINERS_DEFAULT CONTAINER_MAP CONTAINER_MAP_OBJECT TABLE
------------------ ------------- -------------------- ---------
NO                 YES           NO                   DEPT
NO                 YES           NO                   EMP
NO                 NO            YES                  MAPTABLE

SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app
              BEGIN UPGRADE '1.1' TO '1.2';
  2
Pluggable database altered.

SQL> ALTER TABLE scott.dept ENABLE containers_default;

Table altered.

SQL> ALTER TABLE scott.emp ENABLE containers_default;

Table altered.

SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app
              END UPGRADE TO '1.2';
  2
Pluggable database altered.
```

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@research AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@accounting AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT containers_default, container_map,
            container_map_object, table_name "TABLE"
     FROM   dba_tables WHERE owner='SCOTT';
  2    3
CONTAINERS_DEFAULT CONTAINER_MAP CONTAINER_MAP_OBJECT TABLE
------------------ ------------- -------------------- ---------
YES                YES           NO                   DEPT
YES                YES           NO                   EMP
NO                 NO            YES                  MAPTABLE

SQL> CONNECT scott@hr_root
Enter password: ******
Connected.
SQL> SELECT deptno, dname, loc
     FROM   scott.dept where deptno = 20;
  2
    DEPTNO DNAME          LOC
---------- -------------- -------------
```

```
        20 RESEARCH       DALLAS

SQL> SET autot traceo explain
SQL> SET linesize 200
SQL> SELECT deptno, dname, loc
     FROM scott.dept where deptno = 20;
  2
Execution Plan
----------------------------------------------------------
Plan hash value: 4285497843


---------------------------------------------------------------
| Id  | Operation             | Name | Rows  | Bytes | Cost
(%CPU)| Time     | Pstart| Pstop |
---------------------------------------------------------------
|   0 | SELECT STATEMENT      |      |  1900 | 57000 |     5
(100)| 00:00:01 |       |       |
|   1 |  PARTITION LIST SINGLE|      |  1900 | 57000 |     5
(100)| 00:00:01 |     2 |     2 |
|   2 |   CONTAINERS FULL     | DEPT |  1900 | 57000 |     5
(100)| 00:00:01 |       |       |
---------------------------------------------------------------
SQL>
```

*Notice in the execution plan the **Pstart** and **Pstop** columns. The data is selected from only one partition of the container map table and therefore the query is routed to the appropriate PDB for the data.*

```
SQL> SET autot off
SQL> SELECT empno, ename, deptno FROM scott.emp
     WHERE  deptno=30;
  2
     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7499 ALLEN              30
      7521 WARD               30
      7654 MARTIN             30
      7698 BLAKE              30
      7844 TURNER             30
      7900 JAMES              30

6 rows selected.

SQL> SET autot traceo explain
SQL> /
```

```
Execution Plan
----------------------------------------------------------
```

```
Plan hash value: 1523017463


---------------------------------------------------------------
| Id  | Operation                | Name | Rows  | Bytes | Cost
(%CPU)| Time      | Pstart| Pstop |
---------------------------------------------------------------
|   0 | SELECT STATEMENT         |      |  1900 | 62700 |     5
(100)| 00:00:01 |       |       |
|   1 |  PARTITION LIST SINGLE|      |  1900 | 62700 |     5
(100)| 00:00:01 |     3 |     3 |
|   2 |   CONTAINERS FULL        | EMP  |  1900 | 62700 |     5
(100)| 00:00:01 |       |       |
---------------------------------------------------------------
SQL> EXIT
$
```

3. Drop the `hr_root` application container.

```
$ $HOME/labs/APP/cleanup_hr_app.sh
…
$
```

Practice 3-3: Applying Recorded Statements in Application PDBs

Overview

In this practice, you will observe how DDL and DML statements on data-linked objects can be reapplied on closed application PDBs and newly created PDBs.

Tasks

1. Connect to the `toys_root` application container.

```
$ sqlplus sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT app_name, app_version APP_Vers, app_status Status,
            app_capture_service, a.con_id, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  app_name NOT LIKE 'APP$%'
     AND    a.con_id = p.pdb_id;
  2    3    4    5
APP_NAME   APP_VERS STATUS   APP_CAPTURE_SE CON_ID PDB_NAME
---------- -------- -------- -------------- ------ ------------
TOYS_APP   1.2      NORMAL   toys_root$seed     10 DOODLES
TOYS_APP   1.5      NORMAL   dolls               7 DOLLS
TOYS_APP   1.5      NORMAL   robots              6 ROBOTS
TOYS_APP   1.5      NORMAL   toys_root           4 TOYS_ROOT

SQL>
```

Q/ Why is there an application PDB not in sync with the application root?

A/ This is the `doodles` application PDB. This application PDB was created from the application seed that was synchronized when the application was upgraded to version `1.5`. It is not required that the application seed is in sync with the application root. But in this case, any newly created application PDB will be in an earlier version than the current application version.

2. Upgrade the application to add a new column and insert a new row in the data-linked `toys_owner.codes` table while the application PDBs are closed.

```
SQL> ALTER PLUGGABLE DATABASE ALL CLOSE;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         4 TOYS_ROOT                      READ WRITE NO
         5 TOYS_ROOT$SEED                 MOUNTED
         6 ROBOTS                         MOUNTED
         7 DOLLS                          MOUNTED
        10 DOODLES                        MOUNTED
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                     BEGIN UPGRADE '1.5' TO '1.6';
 2
Pluggable database altered.

SQL> ALTER TABLE toys_owner.codes ADD (c3 CHAR(1) default 'Y');

Table altered.

SQL> INSERT INTO toys_owner.codes VALUES (3, 'Game', 'N');

1 row updated.

SQL> COMMIT;

Commit complete.

SQL> SELECT * FROM toys_owner.codes;

      CODE LABEL      C
---------- ---------- -
         1 Puppet     Y
         2 Car        Y
         3 Game       N

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                     END UPGRADE TO '1.6';
 2
Pluggable database altered.

SQL>
```

3. Open the application PDBs and check whether the updates have been applied.

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> ALTER SYSTEM FLUSH buffer_cache;

System altered.
```

```
SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> SELECT code, label FROM toys_owner.codes;

      CODE LABEL
---------- ----------
         1 Puppet
         2 Car

SQL>
```

Q/ Which operation was required to retrieve a synchronized table definition and synchronized data?

A/ Synchronization of the application PDBs is required.

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> SELECT code, label FROM toys_owner.codes;

      CODE LABEL
---------- ----------
         1 Puppet
         2 Car
         3 Game

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> SELECT code, label, c3 FROM toys_owner.codes;

      CODE LABEL      C
---------- ---------- -
         1 Puppet     Y
         2 Car        Y
         3 Game       N

SQL>
```

4. Create a new application PDB and check whether the updates have been applied.
   a. First synchronize the application seed.

```
SQL> CONNECT sys@toys_root$seed AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> SELECT code, label, c3 FROM toys_owner.codes;

      CODE LABEL      C
---------- ---------- -
         1 Puppet     Y
         2 Car        Y
         3 Game       N

SQL>
```

   b. Create the new application PDB.

```
SQL> ! mkdir /u02/app/oracle/oradata/ORCL/toys_root/newPDB

SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SESSION SET db_create_file_dest =
           '/u02/app/oracle/oradata/ORCL/toys_root/newPDB';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE newpdb
```

```
            ADMIN USER admin IDENTIFIED BY oracle_4U;
  2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE newpdb OPEN;

Pluggable database altered.

SQL> CONNECT sys@newpdb AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT code, label, c3 FROM toys_owner.codes;

      CODE LABEL      C
---------- ---------- -
         1 Puppet     Y
         2 Car        Y
         3 Game       N

SQL>
```

c. Drop the new PDB.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE newpdb CLOSE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE newpdb INCLUDING DATAFILES;

Pluggable database dropped.

SQL>
```

Q/ What happens if the data-linked table is dropped?

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                     BEGIN UPGRADE '1.6' TO '1.7';
 2
Pluggable database altered.

SQL> DROP TABLE toys_owner.codes;

Table dropped.
```

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                    END UPGRADE TO '1.7';
 2
Pluggable database altered.

SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.codes;
```

```
      CODE LABEL      C
---------- ---------- -
         1 Puppet     Y
         2 Car        Y
         3 Game       N
```

```


SQL>
```

A/ Application PDBs still see the table. Application PDBs are not synchronized with the application root.

```
SQL> SELECT app_name, app_version APP_Vers, app_status Status,
            a.con_id, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  app_name NOT LIKE 'APP$%'
     AND    a.con_id = p.pdb_id;
  2    3    4    5

APP_NAME   APP_VERS STATUS   CON_ID PDB_NAME
---------- -------- -------- ------ --------------
TOYS_APP   1.6      NORMAL        6 ROBOTS

```

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> SELECT app_name, app_version APP_Vers, app_status Status,
            a.con_id, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  app_name NOT LIKE 'APP$%'
     AND    a.con_id = p.pdb_id;
  2    3    4    5
```

```
APP_NAME   APP_VERS STATUS   CON_ID PDB_NAME
---------- -------- -------- ------ --------------
TOYS_APP   1.7      NORMAL        6 ROBOTS

SQL> SELECT * FROM toys_owner.codes;
SELECT * FROM toys_owner.codes
                         *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@doodles AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL>
```

d. Check that all `toys_root` application PDBs are in sync with the application root.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT app_name, app_version APP_Vers, app_status Status,
            a.con_id, p.pdb_name
     FROM   cdb_applications a, cdb_pdbs p
     WHERE  app_name NOT LIKE 'APP$%'
     AND    a.con_id = p.pdb_id;
  2    3    4    5
APP_NAME   APP_VERS STATUS   CON_ID PDB_NAME
---------- -------- -------- ------ --------------
TOYS_APP   1.7      NORMAL        6 ROBOTS
TOYS_APP   1.7      NORMAL        4 TOYS_ROOT
TOYS_APP   1.7      NORMAL       10 DOODLES
TOYS_APP   1.7      NORMAL        7 DOLLS

SQL> EXIT
$
```

Practice 3-4: Managing PDB Lockdown Profiles

Overview

In this practice, you will configure PDB lockdown profiles so that in `robots`, the users will not be allowed to use the `ALTER SYSTEM` command except by using the clause `FLUSH SHARED POOL`, and in `dolls`, the users will not be allowed to use the `ALTER SYSTEM` command except by using the clause `SET`.

Tasks

1. Create the PDB lockdown profiles.

```
$ sqlplus / AS SYSDBA
Enter password: ******
Connected.
SQL> CREATE LOCKDOWN PROFILE alter_flush;

Lockdown Profile created.

SQL> ALTER LOCKDOWN PROFILE alter_flush
           DISABLE STATEMENT = ('alter system');
  2
Lockdown Profile altered.

SQL> ALTER LOCKDOWN PROFILE alter_flush
           ENABLE STATEMENT = ('alter system')
           CLAUSE = ('flush shared_pool');
  2    3
Lockdown Profile altered.

SQL> CREATE LOCKDOWN PROFILE alter_set;

Lockdown Profile created.

SQL> ALTER LOCKDOWN PROFILE alter_set
           DISABLE STATEMENT = ('alter system');
  2
Lockdown Profile altered.

SQL> ALTER LOCKDOWN PROFILE alter_set
           ENABLE STATEMENT = ('alter system')
           CLAUSE = ('set');
  2    3
Lockdown Profile altered.
```

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SYSTEM SET PDB_LOCKDOWN = alter_flush SCOPE = both;

System altered.

SQL>
```

2. Test by connecting to `robots` as a DBA.

```
SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> ALTER SYSTEM switch logfile;
ALTER SYSTEM switch logfile
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> ALTER SYSTEM FLUSH shared_pool;

System altered.

SQL> ALTER SYSTEM checkpoint;
ALTER SYSTEM checkpoint
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> ALTER SYSTEM SET ddl_lock_timeout=30 scope=both;
ALTER SYSTEM SET ddl_lock_timeout=10 scope=both
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

3. Configure the `dolls` setting in the `ALTER_SET` PDB lockdown profile and then test.

```
SQL> CONNECT sys@dolls as sysdba
Enter password: ******
Connected.
SQL> ALTER SYSTEM SET PDB_LOCKDOWN = alter_set SCOPE = both;
```

```
System altered.

SQL> CONNECT system@dolls
Enter password: ******
Connected.
SQL> ALTER SYSTEM switch logfile;
ALTER SYSTEM switch logfile
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> ALTER SYSTEM flush shared_pool;
ALTER SYSTEM FLUSH shared_pool
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> ALTER SYSTEM SET ddl_lock_timeout=10 scope=both;

System altered.

SQL>
```

4. Clean up the PDB lockdown profiles. Before dropping the PDB lockdown profiles, find the CDB_*xxx* view that lists the PDB lockdown profiles and reset the `PDB_LOCKDOWN` parameter to null.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT profile_name, rule, clause, status
     FROM   cdb_lockdown_profiles
     WHERE  rule IS NOT NULL;
  2    3
PROFILE_NAME   RULE           CLAUSE               STATUS
-------------- -------------- -------------------- --------
ALTER_FLUSH    ALTER SYSTEM                        DISABLE
ALTER_FLUSH    ALTER SYSTEM   FLUSH SHARED_POOL    ENABLE
ALTER_SET      ALTER SYSTEM                        DISABLE
ALTER_SET      ALTER SYSTEM   SET                  ENABLE

SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SYSTEM SET PDB_LOCKDOWN='' scope=both;
```

```
ALTER SYSTEM SET PDB_LOCKDOWN='' scope=both
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER LOCKDOWN PROFILE alter_flush
           ENABLE STATEMENT = ('alter system');
  2
Lockdown Profile altered.

SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SYSTEM SET PDB_LOCKDOWN='' scope=both;

System altered.

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SYSTEM SET PDB_LOCKDOWN='' scope=both;

System altered.

SQL> CONNECT / AS SYSDBA
Connected.
SQL> DROP LOCKDOWN PROFILE alter_flush;

Lockdown Profile dropped.

SQL> DROP LOCKDOWN PROFILE alter_set;

Lockdown Profile dropped.

SQL> SELECT distinct PROFILE_NAME, RULE, CLAUSE, STATUS
     FROM   cdb_lockdown_profiles;
  2
PROFILE_NAME   RULE           CLAUSE            STATUS
-------------- -------------- ----------------- --------
SAAS                                            EMPTY
PRIVATE_DBAAS                                   EMPTY
```

```
PUBLIC_DBAAS                                                EMPTY

SQL> EXIT
$
```

The remaining PDB lockdown profiles are pre-defined PDB lockdown profiles for Oracle Public Cloud customers usage.

Practice 3-5: Auditing Operations in PDBs

Overview

In this practice, you will audit operations performed in PDBs by using Unified Auditing.

Tasks

1. Ensure that Unified Auditing is enabled in `ORCL`.

```
$ sqlplus / AS SYSDBA

SQL> SELECT * FROM v$option
     WHERE parameter = 'Unified Auditing';
  2
PARAMETER              VALUE                CON_ID
---------------------- -------------------- ------
Unified Auditing       FALSE                     0

SQL> EXIT
$
```

Q/ How do you detect that Unified Auditing is not enabled?

A/ `V$OPTION` view does not mention it as enabled.

2. Enable Unified Auditing by using the following shell script:

```
$ $HOME/labs/APP/enable_unified_auditing.sh
…
/usr/bin/ar cr
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/libknlopt.a
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/kzaiang.o
chmod 755 /u01/app/oracle/product/12.2.0/dbhome_1/bin


 - Linking Oracle
rm -f /u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/oracle
/u01/app/oracle/product/12.2.0/dbhome_1/bin/orald  -o
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/oracle -m64 -z
noexecstack -Wl,--disable-new-dtags -
L/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/ -
L/u01/app/oracle/product/12.2.0/dbhome_1/lib/ -
L/u01/app/oracle/product/12.2.0/dbhome_1/lib/stubs/   -Wl,-E
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/opimai.o
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/ssoraed.o
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/ttcsoi.o -Wl,-
-whole-archive -lperfsrv12 -Wl,--no-whole-archive
/u01/app/oracle/product/12.2.0/dbhome_1/lib/nautab.o
/u01/app/oracle/product/12.2.0/dbhome_1/lib/naeet.o
/u01/app/oracle/product/12.2.0/dbhome_1/lib/naect.o
/u01/app/oracle/product/12.2.0/dbhome_1/lib/naedhs.o
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/config.o  -
```

```
lserver12 -lodm12 -lofs -lcell12 -lnnet12 -lskgxp12 -lsnls12 -
lnls12  -lcore12 -lsnls12 -lnls12 -lcore12 -lsnls12 -lnls12 -
lxml12 -lcore12 -lunls12 -lsnls12 -lnls12 -lcore12 -lnls12 -
lclient12  -lvsn12 -lcommon12 -lgeneric12 -lknlopt `if
/usr/bin/ar tv
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/libknlopt.a |
grep xsyeolap.o > /dev/null 2>&1 ; then echo "-loraolap12" ; fi`
-lskjcx12 -lslax12 -lpls12  -lrt -lplp12 -lserver12 -lclient12
-lvsn12 -lcommon12 -lgeneric12 `if [ -f
/u01/app/oracle/product/12.2.0/dbhome_1/lib/libavserver12.a ] ;
then echo "-lavserver12" ; else echo "-lavstub12"; fi` `if [ -f
/u01/app/oracle/product/12.2.0/dbhome_1/lib/libavclient12.a ] ;
then echo "-lavclient12" ; fi` -lknlopt -lslax12 -lpls12  -lrt -
lplp12 -ljavavm12 -lserver12  -lwwg  `cat
/u01/app/oracle/product/12.2.0/dbhome_1/lib/ldflags`    -
lncrypt12 -lnsgr12 -lnzjs12 -ln12 -lnl12 -lngsmshd12 -lnro12
`cat /u01/app/oracle/product/12.2.0/dbhome_1/lib/ldflags`    -
lncrypt12 -lnsgr12 -lnzjs12 -ln12 -lnl12 -lngsmshd12 -lnnzst12 -
lzt12 -lztkg12 -lmm -lsnls12 -lnls12  -lcore12 -lsnls12 -lnls12
-lcore12 -lsnls12 -lnls12 -lxml12 -lcore12 -lunls12 -lsnls12 -
lnls12 -lcore12 -lnls12 -lztkg12 `cat
/u01/app/oracle/product/12.2.0/dbhome_1/lib/ldflags`    -
lncrypt12 -lnsgr12 -lnzjs12 -ln12 -lnl12 -lngsmshd12 -lnro12
`cat /u01/app/oracle/product/12.2.0/dbhome_1/lib/ldflags`    -
lncrypt12 -lnsgr12 -lnzjs12 -ln12 -lnl12 -lngsmshd12 -lnnzst12 -
lzt12 -lztkg12   -lsnls12 -lnls12  -lcore12 -lsnls12 -lnls12 -
lcore12 -lsnls12 -lnls12 -lxml12 -lcore12 -lunls12 -lsnls12 -
lnls12 -lcore12 -lnls12 `if /usr/bin/ar tv
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/libknlopt.a |
grep "kxmnsd.o" > /dev/null 2>&1 ; then echo " " ; else echo "-
lordsdo12 -lserver12"; fi` -
L/u01/app/oracle/product/12.2.0/dbhome_1/ctx/lib/ -lctxc12 -
lctx12 -lzx12 -lgx12 -lctx12 -lzx12 -lgx12 -lordimt12 -lclsra12
-ldbcfg12 -lhasgen12 -lskgxn2 -lnnzst12 -lzt12 -lxml12 -locr12 -
locrb12 -locrutl12 -lhasgen12 -lskgxn2 -lnnzst12 -lzt12 -lxml12
-lgeneric12 -lorazip -loraz -llzopro -lorabz2 -lipp_z -lipp_bz2
-lippdcemerged -lippsemerged -lippdcmerged  -lippsmerged -
lippcore  -lippcpemerged -lippcpmerged  -lsnls12 -lnls12  -
lcore12 -lsnls12 -lnls12 -lcore12 -lsnls12 -lnls12 -lxml12 -
lcore12 -lunls12 -lsnls12 -lnls12 -lcore12 -lnls12 -lsnls12 -
lunls12  -lsnls12 -lnls12  -lcore12 -lsnls12 -lnls12 -lcore12 -
lsnls12 -lnls12 -lxml12 -lcore12 -lunls12 -lsnls12 -lnls12 -
lcore12 -lnls12 -lasmclnt12 -lcommon12 -lcore12  -laio -lons  -
lfthread12   `cat
/u01/app/oracle/product/12.2.0/dbhome_1/lib/sysliblist` -Wl,-
rpath,/u01/app/oracle/product/12.2.0/dbhome_1/lib -lm    `cat
/u01/app/oracle/product/12.2.0/dbhome_1/lib/sysliblist` -ldl -lm
-L/u01/app/oracle/product/12.2.0/dbhome_1/lib
test ! -f /u01/app/oracle/product/12.2.0/dbhome_1/bin/oracle ||\
       mv -f /u01/app/oracle/product/12.2.0/dbhome_1/bin/oracle
/u01/app/oracle/product/12.2.0/dbhome_1/bin/oracleO
```

```
mv /u01/app/oracle/product/12.2.0/dbhome_1/rdbms/lib/oracle
/u01/app/oracle/product/12.2.0/dbhome_1/bin/oracle
chmod 6751 /u01/app/oracle/product/12.2.0/dbhome_1/bin/oracle
…
$
```

3. Ensure that Unified Auditing is now enabled in ORCL.

```
$ sqlplus / AS SYSDBA

SQL> SELECT * FROM v$option
     WHERE parameter = 'Unified Auditing';
  2
PARAMETER              VALUE                CON_ID
---------------------- -------------------- ------
Unified Auditing       TRUE                      0

SQL>
```

4. You will audit the SELECT ANY TABLE system privilege and LOGON action for any user connected in application PDBs in the toys_root application container.

   a. Grant the SELECT ANY TABLE system privilege to the toys_test2 application common user in the toys_root application container. Use the $HOME/labs/APP/script_grant.sql SQL script.
   Note: If the connections in the script fail, edit $ORACLE_HOME/network/admin/listener.ora to update all localhost occurrences to your server name.
   Note: Edit the script to make the application version number match your own version number.

```
SQL> @$HOME/labs/APP/script_grant.sql
…
SQL>
```

   b. Create the audit policy at the application root level.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> CREATE AUDIT POLICY user_toys_pol
       PRIVILEGES select any table ACTIONS logon CONTAINER=all;
  2
Audit policy created.

SQL> AUDIT POLICY user_toys_pol;

Audit succeeded.

SQL> SELECT user_name, enabled_option
```

```
     FROM   audit_unified_enabled_policies
     WHERE  policy_name = 'USER_TOYS_POL';
  2    3
USER_NAME ENABLED_OPTION
--------- ----------------
ALL USERS BY USER

SQL> SELECT policy_name, common, inherited, audit_option
     FROM   audit_unified_policies
     WHERE  policy_name = 'USER_TOYS_POL';
  2    3
POLICY_NAME      COMMON INH AUDIT_OPTION
---------------- ------ --- ----------------
USER_TOYS_POL    YES     NO  SELECT ANY TABLE
USER_TOYS_POL    YES     NO  LOGON

SQL>
```

Q1/ Is the `user_toys_pol` audit policy common in the `toys_root` application container?

A1/ The `YES` value in the `COMMON` column in the `audit_unified_policies` view means that the policy is a common audit policy. The `NO` value in the `INHERITED` column in the `audit_unified_policies` view means that the policy is not inherited from the CDB root and therefore an application common audit policy.

Q2/ Did you have to explicitly declare an application BEGIN-END block to create the application common audit policy in the `toys_root` application container?

A2/ An implicit application BEGIN-END block for application common unified audit policies is automatically added when the end user does not create them inside an explicit application BEGIN-END block. Therefore, it is not mandatory to create application common unified audit policies within an explicit application BEGIN-END block.

c. Let application common users `toys_owner` and `toys_test2` perform audited operations in the `robots` and `dolls` application PDBs.

```
SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
SQL> EXEC dbms_audit_mgmt.clean_audit_trail(-
          dbms_audit_mgmt.audit_trail_unified, false,-
          dbms_audit_mgmt.container_current)
> >
PL/SQL procedure successfully completed.

SQL> SELECT count(*) FROM toys_owner.sales_data;

  COUNT(*)
```

```
----------
         0

SQL> CONNECT toys_test2@robots
Enter password: ******
Connected.
SQL> SELECT count(*) FROM toys_owner.sales_data;

  COUNT(*)
----------
         0

SQL> SELECT count(*) FROM sys.tab$;
SELECT count(*) FROM sys.tab$
                         *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
```

```
SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
SQL> EXEC dbms_audit_mgmt.clean_audit_trail(-
          dbms_audit_mgmt.audit_trail_unified, false,-
          dbms_audit_mgmt.container_current)
> >
PL/SQL procedure successfully completed.

SQL> SELECT count(*) FROM toys_owner.sales_data;

  COUNT(*)
----------
         0

SQL> SELECT count(*) FROM sys.obj$;

  COUNT(*)
----------
     75928

SQL> CONNECT toys_test2@dolls
```

```
Enter password: ******
Connected.
SQL> SELECT count(*) FROM toys_owner.sales_data;

  COUNT(*)
----------
         0

SQL> SELECT count(*) FROM sys.obj$;
SELECT count(*) FROM sys.obj$
                         *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
```

d. Check whether `LOGON` actions and use of the `SELECT ANY TABLE` privilege are audited.

```
SQL> CONNECT system@toys_root
Enter password: ******
Connected.
SQL> SELECT dbusername, action_name, object_name,
            system_privilege_used, pdb_name
     FROM   cdb_unified_audit_trail u, cdb_pdbs p
     WHERE  u.dbid = p.dbid
     AND    UNIFIED_AUDIT_POLICIES ='USER_TOYS_POL';
  2    3    4    5
DBUSERNAME ACTION_NAME  OBJECT_NAM SYSTEM_PRIVILEGE PDB_NAME
---------- ------------ ---------- ---------------- -----------
SYSTEM     LOGON                   CREATE SESSION   TOYS_ROOT

SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> SELECT dbusername, action_name, object_name,
            system_privilege_used, pdb_name
     FROM   cdb_unified_audit_trail u, cdb_pdbs p
     WHERE  u.dbid = p.dbid
     AND    UNIFIED_AUDIT_POLICIES ='USER_TOYS_POL';
  2    3    4    5
DBUSERNAME ACTION_NAME  OBJECT_NAM SYSTEM_PRIVILEGE PDB_NAME
---------- ------------ ---------- ---------------- -----------
TOYS_TEST2 SELECT       SALES_DATA SELECT ANY TABLE ROBOTS
SYSTEM     LOGON                   CREATE SESSION   ROBOTS
```

```
TOYS_TEST2 LOGON                        CREATE SESSION   ROBOTS

SQL> CONNECT system@dolls
Enter password: ******
Connected.
SQL> SELECT dbusername, action_name, object_name,
            system_privilege_used, pdb_name
     FROM   cdb_unified_audit_trail u, cdb_pdbs p
     WHERE  u.dbid = p.dbid
     AND    UNIFIED_AUDIT_POLICIES ='USER_TOYS_POL';
  2    3    4    5
DBUSERNAME ACTION_NAME  OBJECT_NAM SYSTEM_PRIVILEGE PDB_NAME
---------- ------------ ---------- ---------------- -----------
TOYS_TEST2 SELECT       SALES_DATA SELECT ANY TABLE DOLLS
SYSTEM     LOGON                   CREATE SESSION   DOLLS
TOYS_TEST2 LOGON                   CREATE SESSION   DOLLS

SQL> CONNECT system@ORCL
Enter password: ******
Connected.
SQL> SELECT dbusername, action_name, object_name,
            system_privilege_used, pdb_name
     FROM   cdb_unified_audit_trail u, cdb_pdbs p
     WHERE  u.dbid = p.dbid
     AND    UNIFIED_AUDIT_POLICIES ='USER_TOYS_POL';
  2    3    4    5
DBUSERNAME ACTION_NAME  OBJECT_NAM SYSTEM_PRIVILEGE PDB_NAME
---------- ------------ ---------- ---------------- -----------
SYSTEM     LOGON                   CREATE SESSION   TOYS_ROOT
TOYS_TEST2 SELECT       SALES_DATA SELECT ANY TABLE ROBOTS
SYSTEM     LOGON                   CREATE SESSION   ROBOTS
TOYS_TEST2 LOGON                   CREATE SESSION   ROBOTS
TOYS_TEST2 SELECT       SALES_DATA SELECT ANY TABLE DOLLS
SYSTEM     LOGON                   CREATE SESSION   DOLLS
TOYS_TEST2 LOGON                   CREATE SESSION   DOLLS

7 rows selected.

SQL>
```

e. Drop the audit policy.

```
SQL> NOAUDIT POLICY user_toys_pol;
NOAUDIT POLICY user_toys_pol
*
ERROR at line 1:
ORA-46357: Audit policy USER_TOYS_POL not found.
SQL>
```

Q/ The policy exists. Why isn't it recognized?

A/ The policy was created at the application root level.

```
SQL> CONNECT system@toys_root
Enter password: ******
Connected.
SQL> NOAUDIT POLICY user_toys_pol;

Noaudit succeeded.

SQL> DROP AUDIT POLICY user_toys_pol;

Audit Policy dropped.

SQL>
```

5. You will audit the `SELECT` object privilege for any user connected in application PDBs in the `toys_root` application container.

   a. Grant the `SELECT` object privilege on the shared `sales_data` table to the `toys_test2` application common user in the `toys_root` application container. Use the `$HOME/labs/APP/script_grant2.sql` SQL script.
   Note: Edit the script to make the application version number match your own version number.

```
SQL> @$HOME/labs/APP/script_grant2.sql
…
SQL>
```

   b. Create the audit policy at the application root level.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> CREATE AUDIT POLICY user_toys_pol2
       ACTIONS select ON toys_owner.sales_data CONTAINER=all;
  2
Audit policy created.

SQL> AUDIT POLICY user_toys_pol2;

```

```
Audit succeeded.

SQL> SELECT user_name, enabled_option
     FROM   audit_unified_enabled_policies
     WHERE  policy_name = 'USER_TOYS_POL2';
  2    3
USER_NAME  ENABLED_OPTION
---------- ---------------
ALL USERS  BY USER

SQL> SELECT policy_name, common, inherited, audit_option
     FROM   audit_unified_policies
     WHERE  policy_name = 'USER_TOYS_POL2';
  2    3
POLICY_NAME      COMMON INH AUDIT_OPTION
---------------- ------ --- ----------------
USER_TOYS_POL2   YES     NO  SELECT

SQL>
```

c. Let application common users `toys_owner` and `toys_test2` perform audited operations in the `robots` and `dolls` application PDBs.

```
SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
SQL> EXEC dbms_audit_mgmt.clean_audit_trail(-
          dbms_audit_mgmt.audit_trail_unified, false, -
          dbms_audit_mgmt.container_current)

PL/SQL procedure successfully completed.

SQL> SELECT count(*) FROM toys_owner.sales_data;

  COUNT(*)
----------
         0

SQL> CONNECT toys_test2@robots
Enter password: ******
Connected.
SQL> SELECT count(*) FROM toys_owner.sales_data;

  COUNT(*)
```

```
----------
         0

SQL>
```

```
SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
SQL> EXEC dbms_audit_mgmt.clean_audit_trail(-
          dbms_audit_mgmt.audit_trail_unified, false, -
          dbms_audit_mgmt.container_current)

PL/SQL procedure successfully completed.

SQL> SELECT count(*) FROM toys_owner.sales_data;

  COUNT(*)
----------
         0

SQL> CONNECT toys_test2@dolls
Enter password: ******
Connected.
SQL> SELECT count(*) FROM toys_owner.sales_data;

  COUNT(*)
----------
         0

SQL>
```

d. Check whether `SELECT` on the `toys_owner.sales_data` table is audited.

```
SQL> CONNECT system@toys_root
Enter password: ******
Connected.
SQL> SELECT dbusername, action_name, object_name,
            object_privileges, pdb_name
```

```
     FROM   unified_audit_trail u, cdb_pdbs p
     WHERE  u.dbid = p.dbid
     AND    unified_audit_policies = 'USER_TOYS_POL2';
  2    3    4   5


no rows selected
```

```
SQL> SELECT dbusername, action_name, object_name,
            object_privileges, pdb_name
     FROM   cdb_unified_audit_trail u, cdb_pdbs p
     WHERE  u.dbid = p.dbid
     AND    unified_audit_policies ='USER_TOYS_POL2';
  2    3    4    5

no rows selected

SQL> CONNECT system@robots
Enter password: ******
Connected.
SQL> SELECT dbusername, action_name, object_name, pdb_name
     FROM   unified_audit_trail u, cdb_pdbs p
     WHERE  u.dbid = p.dbid
     AND    unified_audit_policies = 'USER_TOYS_POL2';
  2    3    4
DBUSERNAME ACTION_NAME  OBJECT_NAM PDB_NAME
---------- ------------ ---------- ----------
TOYS_TEST2 SELECT       SALES_DATA  ROBOTS
TOYS_OWNER SELECT       SALES_DATA  ROBOTS

SQL> CONNECT system@ORCL
Enter password: ******
Connected.
SQL> SELECT dbusername, action_name, object_name, pdb_name
     FROM   cdb_unified_audit_trail u, cdb_pdbs p
     WHERE  u.dbid = p.dbid
     AND    unified_audit_policies ='USER_TOYS_POL2';
  2    3    4    5
DBUSERNAME ACTION_NAME  OBJECT_NAM PDB_NAME
---------- ------------ ---------- ----------
TOYS_TEST2 SELECT        SALES_DATA DOLLS
TOYS_OWNER SELECT        SALES_DATA DOLLS
TOYS_TEST2 SELECT        SALES_DATA ROBOTS
TOYS_OWNER SELECT        SALES_DATA ROBOTS

SQL>
```

e. Drop the audit policy.

```
SQL> CONNECT system@toys_root
Enter password: ******
Connected.
SQL> NOAUDIT POLICY user_toys_pol2;

Noaudit succeeded.

SQL> DROP AUDIT POLICY user_toys_pol2;

Audit Policy dropped.

SQL> EXIT
$
```

Practice 3-6: Protecting Application Common Objects with DV Common Realms *(Optional)*

Overview

In this practice, you will protect the `toys_owner.sales_data` and `toys_owner.codes` shared application tables in the `toys_root` application container so that enforcement applies to all application PDBs in the application container that have Database Vault (DV) enabled. Instead of configuring and managing the same realm in every PDB, the common protection can be configured and managed in the application root once and its enforcement will apply to the application PDBs that have DV enabled.

Tasks

1. To protect the `toys_owner.sales_data` and `toys_owner.codes` common tables in the `toys_root` application container, first configure and enable DV in the `CDB` root. Then you will be able to configure and enable DV at the PDB level.
   a. Execute the `$HOME/labs/APP/setup_DV_CDB.sh` shell script. For facilitating the application versions management, the script recreates the `toys_app` application in the `toys_root` application container with a new application version.

```
$ $HOME/labs/APP/setup_DV_CDB.sh
…
$
```

   b. Now, you can configure and enable DV in the `toys_root` application root.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus sys@toys_root AS SYSDBA
Enter password: ******

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
           BEGIN UPGRADE '1.0' to '1.1';
  2
Pluggable database altered.

SQL> CREATE USER sec_admin IDENTIFIED BY oracle_4U
                 CONTAINER = all;
  2
User created.

SQL> CREATE USER accts_admin IDENTIFIED BY oracle_4U
                 CONTAINER = all;
  2
User created.
```

```
SQL> GRANT create session TO sec_admin, accts_admin
                         CONTAINER=ALL;
  2
Grant succeeded.

SQL> GRANT restricted session TO sec_admin CONTAINER=ALL;

Grant succeeded.

SQL> GRANT select ON dba_dv_status TO sec_admin CONTAINER=ALL;

Grant succeeded.

SQL> exec DVSYS.CONFIGURE_DV(-
          dvowner_uname =>'sec_admin',-
          dvacctmgr_uname =>'accts_admin')
> >

PL/SQL procedure successfully completed.

SQL> CONNECT sec_admin@toys_root
Enter password: ******
Connected.
SQL> exec DVSYS.DBMS_MACADM.ENABLE_DV (strict_mode => 'Y')

PL/SQL procedure successfully completed.

SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
           END UPGRADE to '1.1';
  2
Pluggable database altered.

SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;
ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC
*
ERROR at line 1:
```

```
ORA-47503: Database Vault is not enabled in CDB$ROOT or
application root.
ORA-06512: at "SYS.CONFIGURE_DV", line 24
ORA-06512: at "SYS.CONFIGURE_DV", line 73
ORA-06512: at line 1

SQL>
```

Q1/ Which operation is required to complete the DV enforcement?

A1/ Like the instance was restarted to enforce DV configuration and enablement at the CDB level, the application root should be restarted to enforce DV configuration and enablement at the application root level.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT * FROM dba_dv_status;

NAME                 STATUS
-------------------- --------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     FALSE

SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Warning: PDB altered with errors.

SQL>
```

Q2/ Why does the PDB open with errors?

A2/ Find the reason in the `PDB_PLUG_IN_VIOLATIONS` view.

```
SQL> SELECT * FROM dba_dv_status;

NAME                 STATUS
-------------------- --------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     TRUE

SQL> CONNECT sys@robots AS SYSDBA
```

```
Enter password: ******
Connected.
```

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;
ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC
*
ERROR at line 1:
ORA-65290: Application may not be altered.

SQL>
```

Q3/ Which operation is required to allow synchronization?

A3/ As usual, first use the OERR command:
$ oerr ora 65290

65290, 00000, "Application may not be altered."

// *Cause: An attempt was made to alter an application when the application

//   pluggable database was not open.

// *Action: Open the application pluggable database and retry.

//

```
SQL> ALTER PLUGGABLE DATABASE OPEN;

Warning: PDB altered with errors.

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE OPEN;

Warning: PDB altered with errors.

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL>
```

c. DV is now enabled in the application root. Check if it is enabled in the `robots` and `dolls` application PDBs too.

```
SQL> CONNECT sec_admin@robots
Enter password: ******
Connected.
SQL> SELECT * FROM dba_dv_status;

```

```
NAME                 STATUS
-------------------- --------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     FALSE

SQL> exec DVSYS.DBMS_MACADM.ENABLE_DV

PL/SQL procedure successfully completed.

SQL> SELECT * FROM dba_dv_status;

NAME                 STATUS
-------------------- --------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     FALSE

SQL>
```

Q/ Which operation is required to complete the DV enforcement?

A/ Like the instance was restarted to enforce DV configuration and enablement at the CDB level and the application root was restarted to enforce DV configuration and enablement at the application root level, the application PDBs need to be restarted to enable enforcement.

```
SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> SELECT * FROM dba_dv_status;

NAME                 STATUS
-------------------- --------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     TRUE



SQL>
```

d. You also enable DV in the `dolls` application PDB.

```
SQL> CONNECT sec_admin@dolls
Enter password: ******
Connected.
SQL> exec DVSYS.DBMS_MACADM.ENABLE_DV

PL/SQL procedure successfully completed.

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> SELECT * FROM dba_dv_status;

NAME                 STATUS
-------------------- --------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     TRUE



SQL>
```

2. Use the `$HOME/labs/APP/script_populate.sql` SQL script to insert rows in the `toys_owner.sales_data` table in `robots`. Then, use `$HOME/labs/APP/script_populate2.sql` to insert rows in the `toys_owner.sales_data` table in `dolls`.

```
SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
SQL> @$HOME/labs/APP/script_populate.sql

PL/SQL procedure successfully completed.

SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
```

```
SQL> @$HOME/labs/APP/script_populate2.sql

PL/SQL procedure successfully completed.

SQL>
```

3. Create a common realm that will be enabled in simulation mode and will write violations to a designated log table.
   a. Create the `TOYS Application` common realm to protect the metadata-linked `toys_owner.sales_data` and the data-linked `toys_owner.codes` common tables, even against the schema owner (*`realm_type => 1`*).
      1) Create the common realm.

```
SQL> CONNECT sec_admin@toys_root
Enter password: ******
Connected.
SQL> EXEC DVSYS.DBMS_MACADM.DELETE_REALM(-
                 realm_name => 'TOYS Application')
BEGIN DVSYS.DBMS_MACADM.DELETE_REALM(realm_name => 'TOYS
Application'); END;

*
ERROR at line 1:
ORA-47241: Realm TOYS Application not found
ORA-06512: at "DVSYS.DBMS_MACADM", line 1085
ORA-06512: at line 1

SQL> EXEC DVSYS.DBMS_MACADM.CREATE_REALM (-
  realm_name    => 'TOYS Application', -
  description   => 'Realm to protect the TOYS application', -
  enabled       => DBMS_MACUTL.G_SIMULATION, -
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL +
DBMS_MACUTL.G_REALM_AUDIT_SUCCESS,-
  realm_type    => 1,-
  realm_scope   => DBMS_MACUTL.G_SCOPE_COMMON)
> > > > > >
PL/SQL procedure successfully completed.

SQL> SELECT name, realm_type, enabled, common, inherited
     FROM   dvsys.dba_dv_realm WHERE name LIKE 'TOYS%';
  2
NAME                 REALM_TYP E COMMON INH
-------------------- --------- - ------ ---
TOYS Application     MANDATORY S YES    NO
```

```
SQL>
```

Observe the new parameter `realm_scope` with two possible values `dbms_macutl.g_scope_common` and `dbms_macutl.g_scope_local`.

The realm type is created as `MANDATORY` (`realm_type => 1`), ***which means that the realm prevents users from accessing realm-secured objects even by using object privileges.***

Observe the `S` value in the `ENABLED` column, which means that the realm is enabled in simulation mode.

2) Add the `toys_owner.sales_data` and `toys_owner.codes` shared tables to the `TOYS Application` realm to be protected against any users being granted any select object or system privilege.

```
SQL> EXEC DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM (-
           realm_name   => 'TOYS Application', -
           object_owner => 'TOYS_OWNER', -
           object_name  => 'SALES_DATA', -
           object_type  => 'TABLE')
> > > >
PL/SQL procedure successfully completed.

SQL> EXEC DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM (-
           realm_name   => 'TOYS Application', -
           object_owner => 'TOYS_OWNER', -
           object_name  => 'CODES', -
           object_type  => 'TABLE')
> > > >
PL/SQL procedure successfully completed.

SQL> SELECT realm_name, common_realm, inherited_realm,
            o.owner, o.object_name, o.object_type, o.sharing
     FROM   dvsys.dba_dv_realm_object r, dba_objects o
     WHERE  r.object_name = o.object_name
     AND    o.owner = 'TOYS_OWNER';
  2    3    4    5
REALM_NAME       COM INH OWNER          OBJECT_NAM OBJECT_TYPE
---------------- --- --- -------------- ---------- ------------
SHARING
-------------
TOYS Application YES NO  TOYS_OWNER     CODES      TABLE
DATA LINK

TOYS Application YES NO  TOYS_OWNER     SALES_DATA TABLE
METADATA LINK
```

```
SQL>
```

4. Test if the two common tables are protected in the application container. Connect as `toys_owner` (the owner) and then as `test` to the application root and to the application PDBs. Prepare other terminal windows, connected as `sec_admin` to `toys_root` (*`Sec_admin window`*), connected as `sec_admin` to `robots` (*`Sec_admin robots window`*), and connected as `sec_admin` to `dolls` (*`Sec_admin dolls window`*).

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus sec_admin@toys_root
Enter password: ******

SQL>
```

Q1/ Start the verification with the first test on both tables connected as `toys_owner` and then as `test` to `toys_root`. Who can access the tables?

CONNECT toys_owner@toys_root	CONNECT test@toys_root	SELECT * FROM
OK	OK	sales_data
OK	OK	codes

A1/ It seems that both `toys_owner` (the owner) and `test` users can access both tables in `toys_root`.

Q2/ Is it the expected behavior?

A2/ No. The expected behavior is that the data in protected common tables in the application root is not visible to users even the schema owner.

Q3/ What is the root cause of the unexpected behavior? How was the realm created?

A3/ The realm was created and enabled in simulation mode. Therefore violations that occurred to the security controls are ONLY logged to the simulation log file. They are not enforced nor is access denied to the user.
Switch to the `Sec_admin window`.

```
SQL> SELECT username, violation_type, object_owner,
            object_name, returncode
     FROM   DVSYS.DBA_DV_SIMULATION_LOG
     WHERE  realm_name = 'TOYS Application';
  2   3
USERNAME   VIOLATION_TYPE   OBJECT_OWN OBJECT_NAM RETURNCODE
---------- ---------------- ---------- ---------- ----------
TOYS_OWNER Realm Violation  TOYS_OWNER SALES_DATA       1031
TOYS_OWNER Realm Violation  TOYS_OWNER CODES            1031
TEST       Realm Violation  TOYS_OWNER SALES_DATA       1031
TEST       Realm Violation  TOYS_OWNER CODES            1031

```

```
SQL>
```

Observe that the violations occurred as expected.

CONNECT toys_owner@toys_root	CONNECT test@toys_root	SELECT * FROM
ORA-01031	ORA-01031	sales_data
ORA-01031	ORA-01031	codes

```
SQL> DELETE FROM dvsys.simulation_log$;

4 rows deleted.

SQL> COMMIT;

Commit complete.

SQL> SELECT username, violation_type, object_name, returncode
     FROM   DVSYS.DBA_DV_SIMULATION_LOG
     WHERE  realm_name = 'TOYS Application';
  2    3
no rows selected

SQL> EXIT
$
```

5. Re-create the common realm in real mode.

```
SQL> CONNECT sec_admin@toys_root
Enter password: ******
Connected.
SQL> EXEC DVSYS.DBMS_MACADM.DELETE_REALM(-
                realm_name => 'TOYS Application')
>
PL/SQL procedure successfully completed.

SQL> EXEC DVSYS.DBMS_MACADM.CREATE_REALM (-
  realm_name    => 'TOYS Application', -
  description   => 'Realm to protect the TOYS application', -
  enabled       => DBMS_MACUTL.G_YES, -
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL +
DBMS_MACUTL.G_REALM_AUDIT_SUCCESS,-
  realm_type    => 1,-
  realm_scope   => DBMS_MACUTL.G_SCOPE_COMMON)
> > > > > >
PL/SQL procedure successfully completed.
```

```
SQL> SELECT name, realm_type, enabled, common, inherited
     FROM   dvsys.dba_dv_realm WHERE name LIKE 'TOYS%';
  2
NAME                 REALM_TYP E COMMON INH
-------------------- --------- - ------ ---
TOYS Application     MANDATORY Y YES    NO

SQL>
```

Observe the Y value in the ENABLED column, which means that the realm is enabled in real mode.

6. Add the `toys_owner.sales_data` and `toys_owner.codes` shared tables to the `TOYS Application` realm to be protected against any users being granted any select object or system privilege.

```
SQL> EXEC DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM (-
           realm_name   => 'TOYS Application', -
           object_owner => 'TOYS_OWNER', -
           object_name  => 'SALES_DATA', -
           object_type  => 'TABLE')
> > > >
PL/SQL procedure successfully completed.

SQL> EXEC DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM (-
           realm_name   => 'TOYS Application', -
           object_owner => 'TOYS_OWNER', -
           object_name  => 'CODES', -
           object_type  => 'TABLE')
> > > >
PL/SQL procedure successfully completed.

SQL> SELECT * FROM dvsys.dba_dv_realm_object
     WHERE  owner = 'TOYS_OWNER';
   2
REALM_NAME       COM INH OWNER          OBJECT_NAM OBJECT_TYPE
---------------- --- --- -------------- ---------- ------------
TOYS Application YES NO  TOYS_OWNER     CODES      TABLE
TOYS Application YES NO  TOYS_OWNER     SALES_DATA TABLE

SQL> EXIT
$
```

Q/ Continue the verification on both tables, connected as `toys_owner` and then as `test` to `robots` and then to `dolls`. Which data is inaccessible?

CONNECT to robots	CONNECT to dolls	SELECT * FROM
OK	OK	sales_data
ORA-01031	ORA-01031	codes

A/ Remember that a Database Vault common realm protects only objects within the application root, not local data in metadata-linked objects. If there is shared data in metadata-linked objects, this type of data is protected. If you carefully observe the data retrieved from `toys_owner.sales_data` in either application PDB, you will find that only the local data is displayed and not the rows inserted into the table at the application root level (check rows inserted by the `$HOME/labs/APP/setup_DV_CDB.sh` shell script). The common data-linked objects are fully protected.

7. Execute the `disable_DV.sh` shell script to stop enforcing protection for common objects in the `toys_root` application container and the CDB root.

```
$ $HOME/labs/APP/disable_DV.sh
…
$ sqlplus / AS SYSDBA
Enter password: ******

SQL> SELECT * FROM dba_dv_status;

NAME                 STATUS
-------------------- --------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     FALSE


SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT * FROM dba_dv_status;

NAME                 STATUS
-------------------- --------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     FALSE

SQL> EXIT
$
```

Practice 3-7: Unplugging and Plugging Encrypted PDBs *(Optional)*

Overview

In this practice, you will unplug an encrypted PDB in a one-step operation and then plug the encrypted PDB in a one-step operation in another CDB.

Tasks

1. Execute the `$HOME/labs/APP/enable_TDE_in_ORCL.sh` shell script to set up transparent data encryption (TDE) at the CDB root level and to create the master encryption key for `pdb_orcl`.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ $HOME/labs/APP/enable_TDE_in_ORCL.sh
…
$
```

2. Then verify that the master encryption key for `pdb_orcl` exists.

```
$ sqlplus sys@pdb_orcl AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT KEY_ID, KEY_USE, ACTIVATING_DBNAME,
            ACTIVATING_PDBNAME
     FROM   V$ENCRYPTION_KEYS;
  2    3
KEY_ID                                                KEY_USE
----------------------------------------------------- ----------
ACTIVATING_DBNAME ACTIVATING_PDBNAME
----------------- ------------------
AW/Z7WQhP0+2v/56MlDdkfYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA TDE IN PDB
ORCL              PDB_ORCL

SQL>
```

3. Create a table in `pdb_orcl` with an encrypted column.

```
SQL> CREATE TABLE l_user.test (c number encrypt);

Table created.

SQL> INSERT INTO l_user.test VALUES (1);

1 row created.

SQL> COMMIT;
```

```
Commit complete.

SQL>
```

4. Unplug and drop `pdb_orcl`.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE pdb_orcl CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb_orcl
           UNPLUG INTO '/tmp/pdb_orcl.xml' ENCRYPT USING "tpwd";

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE pdb_orcl KEEP DATAFILES;

Pluggable database dropped.

SQL> EXIT
$
```

5. Plug the unplugged PDB into `cdb2`.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> CREATE PLUGGABLE DATABASE pdb_encrypt
            USING '/tmp/pdb_orcl.xml' NOCOPY
            KEYSTORE IDENTIFIED BY oracle_4U
            DECRYPT USING "tpwd";
  2    3    4  CREATE PLUGGABLE DATABASE pdb_encrypt
*
ERROR at line 1:
ORA-46661: keystore not open in root container

SQL> EXIT
$
```

Q/ You plug in a PDB into another CDB. What is missing to decrypt the implicitly exported master keys?

A/ To decrypt any data, a master key is required and therefore a keystore <u>is required</u> to store the master keys.

```
$ $HOME/labs/APP/enable_TDE_in_CDB2.sh
…
$
$ sqlplus / AS SYSDBA

SQL> CREATE PLUGGABLE DATABASE pdb_encrypt
           USING '/tmp/pdb_orcl.xml' NOCOPY
           KEYSTORE IDENTIFIED BY oracle_4U
           DECRYPT USING "tpwd";
  2    3    4
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb_encrypt OPEN;

Pluggable database created.

SQL> SELECT key_id, key_use, activating_dbname,
           activating_pdbname
     FROM   v$encryption_keys;
  2    3
KEY_ID                                                     KEY_USE
---------------------------------------------------------- ----------
ACTIVATING_DBNAME ACTIVATING_PDBNAME
----------------- ------------------
AW/Z7WQhP0+2v/56MlDdkfYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA TDE IN PDB
cdb2              PDB_ENCRYPT

AUoi7UqsNk9Xv1ssxio/P6IAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA TDE IN PDB
cdb2              PDB2

AQ3gQyp+rU+av02VGIVpykQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA TDE IN PDB
cdb2              CDB$ROOT



SQL> SELECT wrl_parameter, status, wallet_type
     FROM   v$encryption_wallet;
  2
WRL_PARAMETER                          STATUS   WALLET_T
-------------------------------------- -------- --------
/u01/app/oracle/admin/cdb2/tde_wallet/ OPEN     PASSWORD
```

```
SQL>
```

6. Verify that the encrypted data is readable.

```
SQL> CONNECT system@pdb_encrypt
Enter password: ******
Connected.
SQL> SELECT * FROM l_user.test;
SELECT * FROM l_user.test
                     *
ERROR at line 1:
ORA-28365: wallet is not open

SQL>
```

Q/ The keystore is opened. Why does the error message say that it is not opened?

```
SQL> SELECT wrl_parameter, status, wallet_type wallet
     FROM v$encryption_wallet;
  2
WRL_PARAMETER                          STATUS   WALLET
-------------------------------------- -------- --------
                                       CLOSED   UNKNOWN

SQL>
```

A/ The keystore is opened for the CDB root and not for `pdb_orcl`. The keystore for the CDB was opened before the PDB was plugged in.

```
SQL> CONNECT sys@pdb_encrypt AS SYSDBA
Enter password: ******
Connected.
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
                IDENTIFIED BY oracle_4U CONTAINER = current;
  2
keystore altered.

SQL> SELECT wrl_parameter, status, wallet_type
     FROM v$encryption_wallet;

WRL_PARAMETER                          STATUS   WALLET_T
-------------------------------------- -------- --------
                                       OPEN     PASSWORD

SQL> CONNECT system@pdb_encrypt
Connected.
SQL> SELECT * FROM l_user.test;
```

```
         C
----------
         1

SQL> EXIT
$
```

7. Drop the `pdb_encrypt` PDB using the `$HOME/labs/APP/cleanup_TDE_ENCRYPT.sh` script.

```
$ $HOME/labs/APP/cleanup_TDE_ENCRYPT.sh
…
$
```

Practices for Lesson 4: Creation of PDBs Using New Methods

Chapter 4

Practices for Lesson 4: Overview

Practices Overview

In these practices, you will create PDBs by using new methods.

- Unplug and plug an application container using Archive Files.
- Convert a regular PDB to an application PDB.
- Clone a remote PDB in hot mode and with automatic refreshing.
- Relocate a PDB.
- Create and use a proxy PDB to query data across CDBs.

You will also learn about new features like renaming services and configuring durable location transparency. Additionally, you will drop application containers.

Practice 4-1: Unplugging and Plugging Application Containers Using Archive File (*Optional*)

Overview

In this practice, you will create a new application container by unplugging and plugging using archive files.

Tasks

1. Before starting the practice, execute the `$HOME/labs/admin/glogin_4.sh`, `$HOME/labs/APP/cleanup_hr_app.sh`, `$HOME/labs/APP/setup_toys_app.sh`, and `$HOME/labs/APP/setup_pdb_orcl.sh` shell scripts. The first one sets formatting for all columns selected in queries, the second one drops the `hr_root` application container, the third one sets up the `toys_root` application container and the last one creates the `pdb_orcl` regular PDB.

```
$ $HOME/labs/admin/glogin_4.sh
$ $HOME/labs/APP/cleanup_hr_app.sh
…
$ $HOME/labs/APP/setup_toys_app.sh
…
$ $HOME/labs/APP/setup_pdb_orcl.sh
…
$
```

2. Suppose you use the new production `cdb2` instance and need to move the production `toys_root` application container from `ORCL` to `cdb2`. Unplug `toys_root` using archive files for each container (the application root, the application seed, and the application PDBs) and plug the whole application container into `cdb2`.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ rm /tmp/*pdb
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE toys_root CLOSE;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 TOYS_ROOT                      MOUNTED
         4 TOYS_ROOT$SEED                 MOUNTED
         5 ROBOTS                         MOUNTED
         6 DOLLS                          MOUNTED
         7 PDB_ORCL                       READ WRITE

SQL> ALTER PLUGGABLE DATABASE dolls
           UNPLUG INTO '/tmp/dolls.pdb';
  2
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE robots
           UNPLUG INTO '/tmp/robots.pdb';
  2
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE toys_root
           UNPLUG INTO '/tmp/toys_root.pdb';
  2
ALTER PLUGGABLE DATABASE toys_root
*
ERROR at line 1:
ORA-65265: PDB cannot be dropped or unplugged.

SQL>
```

Q1/ Why can this container not be dropped?

```
SQL> !oerr ora 65265
65265, 00000, "PDB cannot be dropped or unplugged."
// *Cause:   An attempt was made to drop or unplug the root of
an application container to which one or more application
pluggable databases (PDBs) belong.
// *Action:  Drop the application PDBs belonging to the
application container before attempting to drop or unplug the
application root.
//
```

A1/ Follow the logical order to unplug an application container: the application PDBs dependent on the application root, then if it exists, the application seed, and finally the application root.

```
SQL> ALTER PLUGGABLE DATABASE toys_root$SEED
           UNPLUG INTO '/tmp/toys_seed.pdb';
  2
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE toys_root
           UNPLUG INTO '/tmp/toys_root.pdb';
  2
```

```
Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE dolls INCLUDING DATAFILES;

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE robots INCLUDING DATAFILES;

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE toys_root$SEED INCLUDING DATAFILES;

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE toys_root INCLUDING DATAFILES;

Pluggable database dropped.

SQL> EXIT
$
```

Q2/ If you dropped the PDB and the datafiles, how can it be plugged into another CDB?

A2/ The new .pdb extension of the compressed file used for the unplug operation created a file containing the PDB manifest AND all of the data files of the PDB.

Q3/ Can you run the PDB plug-in compatibility test without extracting the PDB manifest file from the archive?

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ mkdir -p /u02/app/oracle/oradata/cdb2/toys_root/robots
$ mkdir /u02/app/oracle/oradata/cdb2/toys_root/dolls
$ mkdir /u02/app/oracle/oradata/cdb2/toys_root/toys_seed
$ sqlplus / AS SYSDBA

SQL> SET serveroutput on
SQL> DECLARE
  compat BOOLEAN := FALSE;
  BEGIN
  compat := DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
            pdb_descr_file => '/tmp/toys_root.pdb',
            pdb_name => 'toys_root');
  if compat then
  DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? YES');
  else DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? NO');
  end if;
end;
/
2    3    4    5    6    7    8    9   10   11
Is pluggable compatible? NO

PL/SQL procedure successfully completed.

SQL
```

A3/ No. Query the `PDB_PLUG_IN_VIOLATIONS` view and verify that the PDB will be compatible once it will be created as an application root.

```
SQL> SELECT name, message, action FROM pdb_plug_in_violations;

NAME
--------------
MESSAGE
----------------------------------------------------------------
ACTION
----------------------------------------------------------------
TOYS_ROOT
CDB parameter pga_aggregate_target mismatch: Previous 450M
Current 300M
Please check the parameter in the current CDB

TOYS_ROOT
Undo mode mismatch: PDB using LOCAL undo.  CDB using SHARED
undo.
Either create an undo tablespace in the PDB or be aware that the
CDB will not look at undo in the PDB.

TOYS_ROOT
Application Root plugged in as an Non-Application PDB, requires
approot_to_pdb.sql be run.
Run approot_to_pdb.sql.

SQL>
```

3. Plug the `toys_root` application container into `cdb2`.

   Be careful to follow the logical reverse sequence to plug the application container: the application root first, then if it exists, the application seed, and finally the application PDBs.

```
SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/cdb2/toys_root';
```

```
Session altered.

SQL> CREATE PLUGGABLE DATABASE toys_root
            AS APPLICATION CONTAINER AS CLONE
            USING '/tmp/toys_root.pdb';
  2    3
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE toys_root OPEN;

Pluggable database altered.

SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/cdb2/toys_root/toys
_seed';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE AS SEED
                      AS CLONE USING '/tmp/toys_seed.pdb';
  2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE toys_root$seed OPEN;

Pluggable database altered.

SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/cdb2/toys_root/robo
ts';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE robots
                      AS CLONE USING '/tmp/robots.pdb';
  2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE robots OPEN;
```

```

Pluggable database altered.

SQ> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/cdb2/toys_root/doll
s';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE dolls
                      AS CLONE USING '/tmp/dolls.pdb';
  2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE dolls OPEN;

Pluggable database altered.

```

```
SQL> SELECT name, con_id, application_root "APP_ROOT",
            application_seed "APP_Seed",
            application_pdb "APP_PDB",
            application_root_con_id "APP_ROOT_CONID"
     FROM   v$containers;
  2    3    4    5
```

```
NAME           CON_ID APP_ROOT APP_Seed APP_PDB  APP_ROOT_CONID
-------------- ------ -------- -------- -------- --------------
TOYS_ROOT           4 YES      NO       NO
TOYS_ROOT$SEED      5 NO       YES      YES                   4
ROBOTS              6 NO       NO       YES                   4
DOLLS               7 NO       NO       YES                   4

SQL>
```

4. Verify that the common tables are accessible from application PDBs.

```
SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
SQL> SELECT * FROM codes;

      CODE LABEL
---------- ----------
         1 Puppet
         2 Car
```

```
SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
SQL> SELECT * FROM sales_data;

no rows selected

SQL> EXIT
$
```

5. Execute the `$HOME/labs/APP/cleanup_toys_app.sh` shell script to drop the `toys_root` application container in `cdb2`.

```
$ $HOME/labs/APP/cleanup_toys_app.sh
…
$
```

Practice 4-2: Renaming Services *(Optional)*

Overview

In this practice, you will manage possible conflicts with names of existing services in CDBs.

Tasks

1. For testing purposes, clone `pdb2` into `ORCL` from the source `cdb2` and open the new PDB.
   a. If `pdb2` in `cdb2` does not exist, execute the `$HOME/labs/APP/setup_pdb2.sh` shell script to create `pdb2` in `cdb2`.

```
$ $HOME/labs/APP/setup_pdb2.sh
…
$
```

   b. Because the `SERVICE_NAME_CONVERT` feature works only on managed services and not the default service, create a new service for `pdb2` in the source `cdb2`.

```
$ . oraenv
ORACLE_SID = [cdb2] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus sys@pdb2 AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE OPEN;
ALTER PLUGGABLE DATABASE OPEN
*
ERROR at line 1:
ORA-65019: pluggable database PDB2 already open
SQL>
SQL> EXEC DBMS_SERVICE.delete_SERVICE('pdb2_node1')
BEGIN DBMS_SERVICE.delete_SERVICE('pdb2_node1'); END;

*
ERROR at line 1:
ORA-44304: service pdb2_node1 does not exist
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 86
ORA-06512: at "SYS.DBMS_SERVICE_ERR", line 23
ORA-06512: at "SYS.DBMS_SERVICE", line 453
ORA-06512: at line 1

SQL> EXEC DBMS_SERVICE.CREATE_SERVICE('pdb2_node1','pdb2_node1')

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_SERVICE.START_SERVICE('pdb2_node1')
```

```

PL/SQL procedure successfully completed.

SQL> SELECT name, network_name FROM dba_services;

NAME           NETWORK_NAME
-------------- --------------
pdb2           pdb2
pdb2_node1     pdb2_node1

SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN READ ONLY;

Pluggable database altered.

SQL> EXIT
$
```

c. Clone `pdb2` into `ORCL` from the source `cdb2` and open the cloned PDB.

```
$ mkdir /u02/app/oracle/oradata/ORCL/pdb2
$
$ . oraenv
ORACLE_SID = [cdb2] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> DROP DATABASE LINK link_cdb2;
DROP DATABASE LINK link_cdb2
                   *
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE DATABASE LINK link_cdb2
           CONNECT TO system IDENTIFIED BY oracle_4U
           USING 'cdb2';
  2    3
Database link created.

SQL> ALTER SESSION SET
     db_create_file_dest='/u02/app/oracle/oradata/ORCL/pdb2';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE pdb2 FROM pdb2@link_cdb2;

Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb2 OPEN;

Pluggable database altered.

SQL> HOST
$
```

Q/ Because the new PDB inherits the service name from the source, how can you avoid conflicts with names of the same service existing in both `ORCL` and `cdb2`?

```
$ lsnrctl status
…
Service "pdb2" has 2 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
  Instance "cdb2", status READY, has 1 handler(s) for this
service...
…
$ EXIT
SQL>
```

A/ Rename the `pdb2` service to another name during the PDB creation. First drop the PDB to recreate it.

```
SQL> ALTER PLUGGABLE DATABASE pdb2 CLOSE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE pdb2 INCLUDING DATAFILES;

Pluggable database dropped.

SQL> CREATE PLUGGABLE DATABASE pdb2 FROM pdb2@link_cdb2
     SERVICE_NAME_CONVERT = ('pdb2_node1', 'pdb2_node2');
  2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb2 OPEN
                    SERVICES = ('pdb2_node2');
  2
Pluggable database altered.

SQL> HOST
$ lsnrctl status
…
Service "pdb2" has 2 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
  Instance "cdb2", status READY, has 1 handler(s) for this
service...
Service "pdb2_node2" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
…
$ exit
SQL> CONNECT sys@pdb2_node2 AS SYSDBA
Enter password: ******
Connected.
SQL> SHOW con_name

CON_NAME
------------------------------
PDB2
SQL> EXIT
$
```

2. Drop `pdb2` from `ORCL` using the `HOME/labs/APP/cleanup_pdb2.sh` script.

```
$ $HOME/labs/APP/cleanup_pdb2.sh
$
```

Practice 4-3: Converting a Regular PDB to an Application PDB *(Optional)*

Overview

In this practice, you will convert the regular `pdb2` of `cdb2` as an application PDB in the `hr_root` application container in `cdb2` so that it can take advantage of all common tables. The conversion uses the clone method. `research` is cloned from `cdb2` as the `research` application PDB of the `hr_root` application container.

Tasks

1. Execute the `$HOME/labs/APP/setup2_hr_app.sh` script. The script installs the `hr_root` application root and the `operations` application PDB. During the `hr_app` application installation, the common `hr_mgr` user creates the shared `hr_mgr.mgr_tab` table.

```
$ $HOME/labs/APP/setup2_hr_app.sh
$
```

2. Clone `pdb2` of `cdb2` as the `research` application PDB in the `hr_root` application container.

   a. Open `pdb2` in read-only mode before cloning it.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE pdb2 CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb2 OPEN READ ONLY;

Pluggable database altered.

SQL>
```

   b. Now, clone `pdb2` as the `research` application PDB in the `hr_root` application container.

```
SQL> ! mkdir /u02/app/oracle/oradata/cdb2/hr_root/research
SQL> ALTER SESSION SET db_create_file_dest =
            '/u02/app/oracle/oradata/cdb2/hr_root/research';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE research FROM pdb2;
```

```
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE research OPEN;

Pluggable database altered.

SQL>
```

Q1/ How can you ensure that `research` is an application PDB associated to `hr_root`?

A1/ Display the `V$CONTAINERS` view.

```
SQL> SELECT name, con_id, application_root "APP_ROOT",
            application_seed "APP_Seed",
            application_pdb "APP_PDB",
            application_root_con_id "APP_ROOT_CONID"
     FROM   v$containers;
  2    3    4    5
NAME           CON_ID APP_ROOT APP_Seed APP_PDB  APP_ROOT_CONID
-------------- ------ -------- -------- -------- --------------
CDB$ROOT            1 NO       NO       NO
PDB$SEED            2 NO       NO       NO
PDB2                3 NO       NO       NO
HR_ROOT             4 YES      NO       NO
OPERATIONS          5 NO       NO       YES                   4
RESEARCH            6 NO       NO       NO

6 rows selected.

SQL>
```

Q2/ Why is the `research` application PDB not associated to `hr_root`?

A2/ You were not connected to `hr_root` when you created the application PDB. It is therefore a regular PDB.

```
SQL> ALTER PLUGGABLE DATABASE research CLOSE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE research INCLUDING DATAFILES;

Pluggable database dropped.

SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
```

```
SQL> ALTER SESSION SET db_create_file_dest =
           '/u02/app/oracle/oradata/cdb2/hr_root/research';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE research FROM pdb2;

Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE research OPEN;

Warning: PDB altered with errors.

SQL> SELECT name, con_id, application_root "APP_ROOT",
            application_seed "APP_Seed",
            application_pdb "APP_PDB",
            application_root_con_id "APP_ROOT_CONID"
     FROM   v$containers;
  2    3    4    5
NAME           CON_ID APP_ROOT APP_Seed APP_PDB  APP_ROOT_CONID
-------------- ------ -------- -------- -------- --------------
HR_ROOT             4 YES      NO       NO
OPERATIONS          5 NO       NO       YES                   4
RESEARCH            6 NO       NO       YES                   4

SQL>
```

Q3/ Which is the reason why the PDB open sends a warning?

```
SQL> SELECT cause, type, message, status, action
     FROM pdb_plug_in_violations
     WHERE  name='RESEARCH';

CAUSE
----------------------------------------------------------------
TYPE
----------
MESSAGE
----------------------------------------------------------------
STATUS
----------
ACTION
----------------------------------------------------------------
Application
WARNING
Application HR_APP in Application Root does not exist in
Application  PDB.
PENDING
Fix the application in the PDB or the application root

Non-Application PDB to Application PDB
ERROR
Non-Application PDB plugged in as an Application PDB, requires
pdb_to_apppdb.sql be run.
PENDING
Run pdb_to_apppdb.sql.


SQL>
```

A3/ The content of the `PDB_PLUG_IN_VIOLATIONS` *view explains that the main error is that the* `pdb_to_apppdb.sql` script *must be executed on the converted regular PDB to become a full application PDB*.

Q4/ Is the shared application table `hr_mgr.mgr_tab` *recognized in* `research`*?*

```
SQL> CONNECT sys@research AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT count(*) FROM hr_mgr.mgr_tab;
SELECT count(*) FROM hr_mgr.mgr_tab
                            *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
```

A4/ The common application entities like users and shared tables are not recognized yet.

Q4/ Are the local tables like `test.bigtab` *recognized in* `research`*?*

```
SQL> SELECT count(*) from test.bigtab;

  COUNT(*)
----------
     10000


SQL>
```

A4/ The local entities like users and tables are still recognized.

3. Execute the conversion script so that object definitions of objects marked as common in the application root are replaced with links in the application PDB.

```
SQL> @$ORACLE_HOME/rdbms/admin/pdb_to_apppdb
Rem
Rem $Header: rdbms/admin/pdb_to_apppdb.sql /main/2 2016/01/22
08:51:46 thbaby Exp $
Rem
```

```
Rem pdb_to_apppdb.sql
Rem
Rem Copyright (c) 2015, 2016, Oracle and/or its affiliates.
Rem All rights reserved.
Rem
Rem    NAME
Rem      pdb_to_apppdb.sql - PDB to Federation PDB
Rem
Rem    DESCRIPTION
Rem      Converts PDB (standalone or Application ROOT) to
Application PDB
…
SQL>
```

Q/ Is the shared application table `hr_mgr.mgr_tab` recognized in `research`?

```
SQL> SHOW con_name

CON_NAME
------------------------------
RESEARCH
SQL> SHOW user
USER is "SYS"
SQL> SELECT count(*) FROM hr_mgr.mgr_tab;

  COUNT(*)
----------
         0

1 row selected.

SQL>
```

A/ The common application entities are now recognized because the synchronization with the application root has been done.

4. Drop the `hr_root` application container and its application PDBs.

   Just like you followed the logical order to unplug an application container (the application PDBs dependent on the application root first, then if it exists, the application seed, and finally the application root), use the same logical order to drop an application container.

```
SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE all CLOSE;

Pluggable database altered.
```

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

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE operations INCLUDING DATAFILES;

Pluggable database dropped.
```

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE hr_root CLOSE;

Pluggable database altered.
```

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

Pluggable database dropped.

SQL> EXIT
$
```

```
$ rm -rf /u02/app/oracle/oradata/cdb2/hr_root
$
```

Practice 4-4: Cloning Remote PDBs in Hot Mode and Automatic Refreshing

Overview

In this practice, because you have been informed of performance issues on the `pdb_source_for_hotclone` PDB in `ORCL`, you will clone the PDB as the `pdb_hotclone` PDB in the `cdb2` test instance, in hot mode for performance tests. The remote `pdb_source_for_hotclone` production PDB in `ORCL` will still be up and fully functional while the actual clone operation is taking place. At the end of the practice, you will create a resfreshable copy, refreshed manually or automatically, which will allow you to take your time to test the performance issue.

Tasks

1. Execute the `$HOME/labs/APP/setup_hotclone.sh` script that creates the production `pdb_source_for_hotclone` from the CDB seed in `ORCL`, then creates a local `SOURCE_USER` user in `pdb_source_for_hotclone`, and finally creates the `source_user.bigtab` table with thousands of rows.

```
$ $HOME/labs/APP/setup_hotclone.sh
…
$
```

2. In `cdb2`, clone `pdb_hotclone` from `pdb_source_for_hotclone` in hot mode.
   a. Start a transaction in the production `pdb_source_for_hotclone` in a new terminal window. We will name it terminal window 2.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus system@pdb_source_for_hotclone
Enter password : ******

SQL> SET sqlprompt "SQL2> "
SQL2> SELECT DISTINCT label FROM source_user.bigtab;

LABEL
------------------------------
DATA FROM source_user.bigtab

SQL2> UPDATE source_user.bigtab SET label='DATA';

10000 rows updated.

SQL2>
```

b. In the terminal window 1 (the original one), clone the `pdb_hotclone` from `pdb_source_for_hotclone` while the source PDB is still up and fully functional.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ mkdir /u02/app/oracle/oradata/cdb2/hotclone
$ sqlplus / AS SYSDBA

SQL> DROP PUBLIC DATABASE LINK link_pdb_source_for_hotclone;
DROP PUBLIC DATABASE LINK link_pdb_source_for_hotclone
                          *
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE PUBLIC DATABASE LINK
            link_pdb_source_for_hotclone
            CONNECT TO system IDENTIFIED BY oracle_4U
            USING 'pdb_source_for_hotclone';
  2    3    4
Database link created.

SQL> ALTER SESSION SET
   db_create_file_dest='/u02/app/oracle/oradata/cdb2/hotclone';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE pdb_hotclone
     FROM pdb_source_for_hotclone@link_pdb_source_for_hotclone
     REFRESH MODE MANUAL;
 2    3
     FROM pdb_source_for_hotclone@link_pdb_source_for_hotclone
                                 *
ERROR at line 2:
ORA-17628: Oracle error 65035 returned by remote Oracle server
ORA-65035: unable to create pluggable database from

SQL> !oerr ora 65035
65035, 00000, "unable to create pluggable database from %s"
// *Cause:  An attempt was made to clone a pluggable database
that did not have local undo enabled.
// *Action: Enable local undo for the PDB and retry the
operation.
//SQL>
```

Q/ Why does the PDB creation fail?

A/ The root cause is that the remote source CDB is using shared UNDO.

c. In the terminal window 2, roll back the transaction, switch the remote CDB to local UNDO mode, and reopen `pdb_source_for_hotclone`.

```
SQL2> ROLLBACK;

Rollback complete.

SQL2> CONNECT / AS SYSDBA
```

```
Connected.
SQL2> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL2> STARTUP UPGRADE
ORACLE instance started.

Total System Global Area  884998144 bytes
Fixed Size                  4586464 bytes
Variable Size             398459936 bytes
Database Buffers          473956352 bytes
Redo Buffers                7995392 bytes
Database mounted.
Database opened.
SQL2>
```

```
SQL2> ALTER DATABASE local undo ON;

Database altered.

SQL2> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL2> STARTUP
ORACLE instance started.

Total System Global Area 884998144 bytes
Fixed Size                  4586464 bytes
Variable Size             398459936 bytes
Database Buffers          473956352 bytes
Redo Buffers                7995392 bytes
```

```
Database mounted.
Database opened.
SQL2> SELECT property_name, property_value
```

```
      FROM   database_properties
      WHERE  property_name = 'LOCAL_UNDO_ENABLED';
  2    3
PROPERTY_NAME       PROPERTY_VALUE
------------------ --------------
LOCAL_UNDO_ENABLED TRUE

SQL2>
```

```
SQL2> ALTER PLUGGABLE DATABASE pdb_source_for_hotclone OPEN;

Pluggable database altered.

SQL2>
```

Q/ Is it sufficient to set the CDB to local UNDO mode?

A/ Yes. The PDB requires an UNDO tablespace. The `UNDO_1` undo tablespace is automatically created in each PDB once local UNDO mode is set.

3. Verify that you can select the same data from the `source_user.bigtab` table in the clone PDB as in the source PDB.
   a. Restart the update operation in terminal window 2.

```
SQL2> CONNECT system@pdb_source_for_hotclone
Enter password : ******
Connected.
SQL> SET sqlprompt "SQL2> "
SQL2> UPDATE source_user.bigtab SET label='DATA';

10000 rows updated.

SQL2>
```

   b. In the terminal window 1, reattempt to clone the `pdb_hotclone` from `pdb_source_for_hotclone` in hot mode.

```
SQL> CREATE PLUGGABLE DATABASE pdb_hotclone
     FROM pdb_source_for_hotclone@link_pdb_source_for_hotclone
     REFRESH MODE MANUAL;
  2    3
Pluggable database created.

SQL>
```

c. Open `pdb_hotclone` in READ ONLY mode only.

```
SQL> ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY;
ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY
*
ERROR at line 1:
ORA-65085: cannot open pluggable database in read-only mode

SQL>
```

Q/ What is the root cause of the error?

```
SQL> !oerr ora 65085
65085, 00000, "cannot open pluggable database in read-only mode"
// *Cause:  The pluggable database has either been created and
not opened or has not been opened in read/write mode after the
undo mode has been changed for the multitenant container
database (CDB).
// *Action: Open the pluggable database in the read/write or the
restricted mode first.
//
```

A/ It is also required that the source PDB uses local UNDO mode.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP UPGRADE
ORACLE instance started.

Total System Global Area  884998144 bytes
Fixed Size                  4586464 bytes
Variable Size             398459936 bytes
Database Buffers          473956352 bytes
Redo Buffers                7995392 bytes
Database mounted.
Database opened.
SQL>
```

```
SQL> ALTER DATABASE local undo ON;

Database altered.

SQL> SHUTDOWN IMMEDIATE
```

```
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area 884998144 bytes
Fixed Size                  4586464 bytes
Variable Size             398459936 bytes
Database Buffers          473956352 bytes
Redo Buffers                7995392 bytes
Database mounted.
Database opened.
SQL> SELECT property_name, property_value
```

```
     FROM   database_properties
     WHERE  property_name = 'LOCAL_UNDO_ENABLED';
  2    3
PROPERTY_NAME      PROPERTY_VALUE
------------------ --------------
LOCAL_UNDO_ENABLED TRUE
```

```
SQL> ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY;

Pluggable database altered.

SQL>
```

d. Select the same data from `source_user.bigtab` in the cloned PDB.

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

Session altered.

SQL> SELECT DISTINCT label FROM source_user.bigtab;

LABEL
------------------------------
DATA FROM source_user.bigtab

SQL> SELECT count(*) FROM source_user.bigtab;

  COUNT(*)
----------
     10000
```

```
SQL>
```

4. Back in the terminal window 2, commit the updated production source data in `ORCL`.

```
SQL2> COMMIT;

Commit complete.

SQL2>
```

5. Back in the terminal window 1, refresh the data in the cloned PDB in `cdb2`.

```
SQL> ALTER PLUGGABLE DATABASE pdb_hotclone REFRESH;
ALTER PLUGGABLE DATABASE pdb_hotclone REFRESH
*
ERROR at line 1:
ORA-65025: Pluggable database PDB_HOTCLONE is not closed on all
instances.
SQL>
```

The refreshable copy PDB must be closed in order for refresh to be performed. If it is not closed when automatic refresh is attempted, the refresh will be deferred until the next scheduled refresh.

```
SQL> ALTER PLUGGABLE DATABASE pdb_hotclone CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb_hotclone REFRESH;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY;

Pluggable database altered.

SQL> SELECT DISTINCT label FROM source_user.bigtab;

LABEL
------------------------------
DATA

1 row selected.

SQL>
```

6. Drop the current refreshable copy PDB and recreate it to configure it as an automatic refreshable clone.

```
SQL> ALTER PLUGGABLE DATABASE pdb_hotclone CLOSE;

Pluggable database altered.

SQL> ALTER SESSION SET container = CDB$ROOT;

Session altered.

SQL> SELECT pdb_name, status FROM cdb_pdbs;

PDB_NAME     STATUS
------------ ----------
PDB_HOTCLONE REFRESHING
PDB$SEED     NORMAL
PDB2         NORMAL

SQL> DROP PLUGGABLE DATABASE pdb_hotclone
                             INCLUDING DATAFILES;
  2
Pluggable database dropped.
```

```
SQL> ALTER SESSION SET
   db_create_file_dest='/u02/app/oracle/oradata/cdb2/hotclone';
```

```
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE pdb_hotclone
     FROM pdb_source_for_hotclone@link_pdb_source_for_hotclone
     REFRESH MODE every 2 minutes;
  2    3
Pluggable database created.

SQL>
```

The refreshable copy PDB must be closed in order for refresh to be completed. If it is not closed when automatic refresh is attempted, the refresh will be deferred until the next scheduled refresh.

7. Back in the terminal window 2, update and commit the source data in `ORCL`.

```
SQL2> UPDATE source_user.bigtab SET label='DATA2';

10000 rows updated.

SQL2> COMMIT;
```

```

Commit complete.

SQL2> EXIT
$
```

8. Back in the terminal window 1, check after 2 minutes that the data in `pdb_hotclone` is refreshed.

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

Session altered.

SQL> ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY;

Pluggable database altered.

SQL> SELECT DISTINCT label FROM source_user.bigtab;

LABEL
------------------------------
DATA

1 row selected.

SQL>
```

Q/ Why can you not see the updated data?

A/ The refresh is scheduled every 2 minutes. If the PDB is not closed when automatic refresh is attempted, then the refresh is deferred until the next scheduled refresh. You must close and wait until the next automatic refresh operation to reopen the PDB in read only mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb_hotclone CLOSE;

Pluggable database altered.

SQL>
```

Wait 2 minutes.

```
SQL> ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY;

Pluggable database altered.

SQL> SELECT DISTINCT label FROM source_user.bigtab;

LABEL
```

```
------------------------------
DATA2

1 row selected.

SQL> EXIT
$
```

9. Execute the `$HOME/labs/APP/cleanup_hotclones.sh` script to drop the `pdb_source_for_hotclone` in `ORCL` and `pdb_hotclone` in `cdb2`.

```
$ $HOME/labs/APP/cleanup_hotclones.sh
…
$
```

Practice 4-5: Relocating PDBs

In this practice, you will move the test `PDB_ORCL` from `ORCL` into the production `cdb2` in one step, using the new 12.2 feature: Near-zero Downtime PDB Relocation. This is what is called PDB relocation.

Tasks

1. Even if `PDB_ORCL` exists in `ORCL`, execute the `setup_pdb_orcl.sh` shell script to re-create the PDB with a table containing a lot of rows.

```
$ $HOME/labs/APP/setup_pdb_orcl.sh
…
$
```

2. In terminal window (1), verify that the source remote CDB is configured to use local UNDO.

```
$ . oraenv
ORACLE_SID = [cdb2] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> SELECT property_name, property_value
```

```
     FROM   database_properties
     WHERE  property_name = 'LOCAL_UNDO_ENABLED';
```

```
  2    3
PROPERTY_NAME      PROPERTY_VALUE
------------------ --------------
LOCAL_UNDO_ENABLED TRUE

SQL> CONNECT test@pdb_orcl
Enter password: ******
Connected.
SQL> SELECT LABEL, COUNT(*) FROM test.bigtab GROUP BY label;

LABEL                             COUNT(*)
------------------------------ ----------
DATA FROM test.bigtab                10000

SQL>
```

3. Prepare a terminal window (2) to start a session and a transaction in `pdb_orcl` to show that while PDB relocation will take place, the transaction will be transferred to the new relocated PDB.
   DO NOT start the `$HOME/labs/APP/sessions.sh` shell script until `CREATE PLUGGABLE DATABASE pdb_relocated` is ready in the terminal window (3).

4. In terminal window (3), relocate `pdb_orcl` from `ORCL` into `cdb2` as `pdb_relocated`.
   a. In `ORCL`, in terminal window (1), create the database link to access `cdb2`.

```
$ . oraenv
ORACLE_SID = [cdb2] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> DROP PUBLIC DATABASE LINK link_cdb2;
DROP PUBLIC DATABASE LINK link_cdb2
                          *
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE PUBLIC DATABASE LINK link_cdb2
            CONNECT TO system IDENTIFIED BY oracle_4U
            USING 'cdb2';
  2    3
Database link created.

SQL>
```

   b. In terminal window (3), in `cdb2`, create the database link to access `pdb_orcl` in `ORCL`.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> DROP PUBLIC DATABASE LINK link_ORCL;
DROP PUBLIC DATABASE LINK link_ORCL
                          *
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE PUBLIC DATABASE LINK link_ORCL
            CONNECT TO system IDENTIFIED BY oracle_4U
            USING 'ORCL';
  2    3
Database link created.

SQL>
```

c. Relocate `pdb_orcl`. Display the status of the new PDB.

```
SQL> !mkdir /u02/app/oracle/oradata/cdb2/pdb_relocated

SQL> ALTER SESSION SET db_create_file_dest =
            '/u02/app/oracle/oradata/cdb2/pdb_relocated';
  2
System altered.

SQL> CREATE PLUGGABLE DATABASE pdb_relocated
                               FROM pdb_orcl@link_ORCL RELOCATE;
  2
                   *
ERROR at line 2:
ORA-17628: Oracle error 1031 returned by remote Oracle server
ORA-01031: insufficient privileges

SQL>
```

Q/ If you consider the operations performed during the opening of a relocated PDB (read below) and the error message issued above, which administrative privilege is to be granted to the user connected to the source PDB via the DB link?

A/ Grant the `SYSOPER` privilege to the user defined in the DB link.

1. *It sets and synchronizes a begin SCN between the source and target CDBs.*
2. *It copies datafiles, undo, and redo over the database link from source to target.*
3. *The redo will be refreshed on the target while the source is still open and the 'alter pluggable database … open' has not been issued on the target.*
4. *When the 'alter pluggable database … open' is issued, active connections are drained on the source at txn boundaries, and subsequent reconnects will be directed to the new relocated PDB and held in a wait state until the PDB is open.*
5. *During the open statement on the target, transactions are rolled forward and rolled backward for transaction consistency based on an end SCN that is set once the open statement is issued. Then the PDB is recovered to the end SCN.*
6. *Before the open completes, it will close the source PDB and drop the datafiles.*

The service is open to client connections on both servers for a short time.

d. Back in terminal window (1), grant the right privilege to `SYSTEM`.

```
SQL> CONNECT sys@ORCL AS SYSDBA
Enter password: ******
Connected.
SQL> GRANT sysoper TO system CONTAINER=all;

Grant succeeded.

SQL>
```

e. Back in terminal window (3), now relocate `pdb_relocated` from `ORCL` into `cdb2`. *You can relocate with the `AVAILABILITY MAX` clause, which ensures smooth migration of workload and persistent connection forwarding from `ORCL` to `cdb2`.*

   The "maximum availability" mode reduces application impact by handling the migration of connections. The source PDB is preserved in mount state to guarantee the connection forwarding of the listener to the remote listener where the PDB is now relocated. This forwarding persists even after the relocation operation has been completed and effectively allows for no changes to connect strings. It is expected that connect strings are updated at a time that is convenient for the application. Once this is done and all clients connect to the new host without forwarding, the source PDB can be dropped.

```
SQL> CREATE PLUGGABLE DATABASE pdb_relocated
                              FROM pdb_orcl@link_ORCL RELOCATE;

Pluggable database created.

SQL> SELECT pdb_name, status FROM cdb_pdbs;

PDB_NAME      STATUS
------------- ----------
PDB_RELOCATED RELOCATING
PDB$SEED      NORMAL
PDB2          NORMAL

SQL>
```

5. In terminal window (2), execute **`$HOME/labs/APP/sessions.sh`**. It will last around 5000 seconds.

```
$ $HOME/labs/APP/sessions.sh
SQL> Connected.
SQL>   2    3    4    5    6    7    8
```

6. During **`sessions.sh`** execution:

   a. In another terminal window (1), you can display the status of the source PDB.

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

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

SQL>
```

b. In terminal window (3), you can open the relocated PDB in read-only mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb_relocated OPEN READ ONLY;

Pluggable database altered.

SQL> ALTER SESSION SET container = pdb_relocated;

Session altered.

SQL> SELECT label, count(*) FROM test.bigtab GROUP BY label;

LABEL                            COUNT(*)
------------------------------ ----------
DATA FROM test.bigtab               10000
NEW DATA during relocation            484

SQL>
```

Q/ What does the number of 484 rows correspond to?

A/ This is the current number of rows already updated in the source PDB and cloned in the relocated PDB.

7. If you consider that **`sessions.sh`** execution is too long, you can open the relocated PDB in force mode. When the newly created PDB is opened in read-write mode for the first time, the final steps of the relocation take effect.
   a. The source PDB is closed and dropped from the source CDB.
   b. Any session that was established while the PDB was first opened in read-only mode is preserved if the FORCE option is used to transition the PDB from read-only to read-write.

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

Session altered.

SQL> ALTER PLUGGABLE DATABASE pdb_relocated
                              OPEN READ WRITE FORCE;

Pluggable database altered.

SQL>
```

Observe that this interrupts `sessions.sh` execution taking place in the terminal window (2).

```
*
ERROR at line 1:
ORA-03113: end-of-file on communication channel
Process ID: 13551
```

```
Session ID: 8 Serial number: 62494

SQL> Disconnected from Oracle Database 12c Enterprise Edition
Release 12.2.0.1.0 - 64bit Production
$
```

8. Still in terminal window (3), verify that the application data is relocated in `pdb_relocated` in `cdb2`.

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

Session altered.

SQL> SELECT label, count(*) FROM test.bigtab GROUP BY label;

LABEL                              COUNT(*)
------------------------------ ----------
DATA FROM test.bigtab                 10000
NEW DATA during relocation              519

SQL>
```

9. In terminal window (1), verify that `pdb_source_for_hotclone` does not exist in `ORCL` anymore.

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

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

SQL> EXIT
$
```

Q1/ In which situation would the source PDB not be dropped?

A1/ The source PDB would not be dropped if the relocated PDB was created in "maximum availability" mode.

`CREATE PLUGGABLE DATABASE toys_root`

`FROM toys_root@link_ORCL RELOCATE AVAILABILITY MAX;`

In this case, the DBA has to drop the PDB once the connection to the relocated PDB has registered with the new listener.

Q2/ Which container should be logically relocated first when all containers within the application container (the application root and its associated application PDBs) have to be relocated? This is what you would get if `toys_root` still existed.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> !mkdir /u02/app/oracle/oradata/cdb2/pdb_relocated2
SQL> ALTER SESSION SET db_create_file_dest =
            '/u02/app/oracle/oradata/cdb2/pdb_relocated2';
  2
System altered.

SQL> CREATE PLUGGABLE DATABASE toys_root
     FROM toys_root@link_ORCL RELOCATE;
CREATE PLUGGABLE DATABASE toys_root FROM toys_root@link_ORCL
RELOCATE
                                                    *
ERROR at line 1:
ORA-17628: Oracle error 65289 returned by remote Oracle server
ORA-65289: Application root cannot be relocated to another
container database.

SQL>
```

A2/ An application root cannot be relocated to another CDB.

10. In terminal window (3), drop `pdb_relocated` in `cdb2`.

```
SQL> ALTER SESSION SET container = cdb$root;

Session altered.

SQL> ALTER PLUGGABLE DATABASE pdb_relocated CLOSE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE pdb_relocated INCLUDING DATAFILES;

Pluggable database dropped.

SQL> EXIT
$
```

Practice 4-6: Querying Data Across CDBs Using Proxy PDBs

Overview

In this practice, you will query data across `toys_root` application PDBs created in `ORCL` and `cdb2`.
Then you will use this new feature to create an application replica of an application root and proxy the application root to query data across application PDBs in different CDBs.

1. In `ORCL`, create the `toys_root` application root container and install the application.
2. Create the `robots` and `dolls` application PDBs in `toys_root` and synchronize them with the application installed in the application root.
3. In `cdb2`, create an application root replica of `toys_root`:
   – Create a remote clone of the `toys_root` application root, named `toys_rr`.
   – In `ORCL`, in the `toys_root` application root, create the `px_toys_rr` proxy PDB referencing the application root replica `toys_rr` in `cdb2`. The proxy PDB provides a context to execute SQL statements and perform operations in the proxied `toys_rr`.
4. Create the `doodles` application PDB in `toys_rr`.
5. Write the application code to aggregate data across the `robots`, `dolls`, and `doodles` application PDBs.

Tasks

1. Execute the `$HOME/labs/APP/setup_toys_app.sh` shell script to create the `toys_root` application container as requested in points 1) and 2) above.

```
$ $HOME/labs/APP/setup_toys_app.sh
…
$
```

2. In `cdb2`, create the application root replica of `toys_root`.
   a. Create a remote clone of the `toys_root` application root, named `toys_rr`.

```
$ . oraenv
ORACLE_SID = [cdb2] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ mkdir /u02/app/oracle/oradata/cdb2/toys_rr
$ sqlplus / AS SYSDBA

SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/cdb2/toys_rr';

Session altered.

SQL> CREATE PLUGGABLE DATABASE toys_rr AS APPLICATION CONTAINER
     FROM toys_root@link_ORCL;
  2

```

```
Pluggable database created.
```

```
SQL> ALTER PLUGGABLE DATABASE toys_rr OPEN;

Pluggable database altered.

SQL> EXIT
$
```

b. In `ORCL`, in the `toys_root` application root, create the `px_toys_rr` proxy PDB referencing the application root replica `toys_rr` in `cdb2`, as requested in point 3) earlier.

1) Create the directory for the `px_toys_rr` proxy PDB for the files copied from the proxied PDB, `toys_rr` application root.

```
$ . oraenv
ORACLE_SID = [cdb2] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ mkdir /u02/app/oracle/oradata/ORCL/px_toys_rr
$
```

2) Create the `px_toys_rr` proxy PDB in the `toys_root` application root, referencing the `toys_rr` application root.

```
$ sqlplus sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> DROP PUBLIC DATABASE LINK link_cdb2;
DROP PUBLIC DATABASE LINK link_cdb2
                          *
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE PUBLIC DATABASE LINK link_cdb2
            CONNECT TO system IDENTIFIED BY oracle_4U
            USING 'cdb2';
  2    3
Database link created.

SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/ORCL/px_toys_rr';

Session altered.
```

```
SQL> CREATE PLUGGABLE DATABASE px_toys_rr AS PROXY
            FROM toys_rr@link_cdb2;
  2
```

```
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE px_toys_rr OPEN;

Pluggable database altered.

SQL>
```

Q1/ What would have happened if you had created the proxy PDB in CDB root?

A1/ The proxied PDB would not belong to the `toys_root` application container and therefore would not share the common tables. Any query across the application PDBs of the `toys_root` application container, including a root replica of another CDB, would not have taken into account the proxied root replica and thus application PDBs associated to the root replica because the proxy PDB is outside the `toys_root` application source container.

Q2/ Which new column in the CDB_PDBS view certifies that a PDB is a proxy PDB?

```
SQL> COL pdb_name format A20
```

```
SQL> SELECT pdb_name, con_id, is_proxy_pdb, foreign_cdb_dbid,
            foreign_pdb_id
     FROM   cdb_pdbs;
  2   3
PDB_NAME             CON_ID IS_ FOREIGN_CDB_DBID FOREIGN_PDB_ID
-------------------- ------ --- ---------------- --------------
TOYS_ROOT                 3 NO        1434391901              2
TOYS_ROOT$SEED            4 NO        1434391901              2
ROBOTS                    5 NO        1434391901              4
DOLLS                     6 NO        1434391901              4
PX_TOYS_RR                7 YES        653674479              4
```

```

SQL>
```

A2/ The new column `IS_PROXY_PDB` with a value 'YES'.

Q3/ What do the new columns `FOREIGN_CDB_DBID` and `FOREIGN_PDB_ID` display?

A3/ These columns display the DBID of the proxied PDB in the remote CDB.

```
SQL> CONNECT system@cdb2
Enter password: *******
Connected.
SQL> SELECT dbid FROM v$database;

      DBID
----------
 653674479

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

```

PDB_NAME      CON_ID
------------ ------
PDB$SEED           2
PDB2               3
TOYS_RR            4

SQL>
```

The FOREIGN_CDB_DBID in the first query in ORCL corresponds to the DBID in the second query in cdb2. The FOREIGN_PDB_ID in the first query in ORCL corresponds to the CON_ID in the third query in cdb2.

3) Check that you can connect to the proxy PDB and select information from `toys_rr` as if you were connected to `toys_rr`.

```
SQL> CONNECT system@toys_rr
Enter password: ******
Connected.
SQL> SELECT code, label FROM toys_owner.codes;

      CODE LABEL
---------- ----------
         1 Puppet
         2 Car

SQL> CONNECT system@px_toys_rr
Enter password: ******
Connected.
SQL> SELECT code, label FROM toys_owner.codes;

      CODE LABEL
---------- ----------
         1 Puppet
         2 Car

SQL> EXIT
$
```

The context of SQL statements execution from the proxy PDB is by default the proxied PDB and not the proxy PDB.

3. Execute the `$HOME/labs/APP/setup_doodles.sh` shell script to create the `doodles` application PDB in `toys_rr`, as requested in point 4) earlier.

```
$ $HOME/labs/APP/setup_doodles.sh
…
$
```

4. Execute the `$HOME/labs/APP/insert.sh` shell script to insert rows into the application metadata-linked `toys_owner.sales_data` table in the `robots`, `dolls`, and `doodles` application PDBs.

```
$ $HOME/labs/APP/insert.sh
…
$
```

5. Write the application code to aggregate data across the `robots`, `dolls`, and `doodles` application PDBs.

```
$ sqlplus toys_owner@toys_root
Enter password: ******
Connected.
SQL> break on year skip 1
SQL> set pagesize 999
SQL> compute sum label "Yearly Revenue" of revenue on year
SQL> SELECT sum(a.revenue) revenue, a.year, a.region,
            con$name, cdb$name
     FROM   containers(toys_owner.sales_data) a
     GROUP BY a.year, a.region, con$name, cdb$name
     ORDER BY con$name, a.year, a.region;
    2    3    4    5
   REVENUE       YEAR REGION     CON$NAME        CDB$NAME
---------- ---------- ---------- --------------- ----------
     67079       2012 east       DOLLS           ORCL
     64237            north      DOLLS           ORCL
     65842            south      DOLLS           ORCL
     45385            west       DOLLS           ORCL
---------- **********
    242543 Yearly Rev

     54726       2013 east       DOLLS           ORCL
     51890            north      DOLLS           ORCL
     53857            south      DOLLS           ORCL
     54841            west       DOLLS           ORCL
---------- **********
    215314 Yearly Rev

     67347       2012 east       DOODLES         cdb2
     58157            north      DOODLES         cdb2
     66177            south      DOODLES         cdb2
     70955            west       DOODLES         cdb2
---------- **********
    262636 Yearly Rev
```

```
     52971       2013 east       DOODLES         cdb2
     55856            north      DOODLES         cdb2
     58163            south      DOODLES         cdb2
     63389            west       DOODLES         cdb2
---------- **********
    230379 Yearly Rev

     62966       2012 east       ROBOTS          ORCL
     63784            north      ROBOTS          ORCL
     60842            south      ROBOTS          ORCL
     65064            west       ROBOTS          ORCL
---------- **********
    252656 Yearly Rev

     62873       2013 east       ROBOTS          ORCL
     53893            north      ROBOTS          ORCL
     54644            south      ROBOTS          ORCL
     60023            west       ROBOTS          ORCL
---------- **********
    231433 Yearly Rev


24 rows selected.

SQL> EXIT
$
```

Observe that data from the application shared table, `toys_owner.sales_data`, of the `toys_app` application stored in application PDBs in both `cdb2` and `ORCL` are retrieved.

6. You want to balance the data of the `toys_owner.sales_data` shared table stored in the applications PDBs of the `toys_app` application between `ORCL` and `cdb2`.
   a. Execute the `$HOME/labs/APP/relocate_dolls.sh` shell script. The script relocates `dolls` from `ORCL` to `cdb2`.

```
$ $HOME/labs/APP/relocate_dolls.sh
…
$
```

   b. Verify that `dolls` has been relocated into `cdb2` and has been dropped from `ORCL`.

```
$ sqlplus sys@toys_root AS SYSDBA
Enter password: ******
Connected to:
SQL> SHOW pdbs
```

```
    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 TOYS_ROOT                      READ WRITE NO
         4 TOYS_ROOT$SEED                 READ ONLY  NO
         5 ROBOTS                         READ WRITE NO
         7 PX_TOYS_RR                     READ WRITE NO
SQL> CONNECT sys@toys_rr AS SYSDBA
Enter password: ******
Connected.
SQL> SHOW pdbs

   CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         4 TOYS_RR                        READ WRITE NO
         5 DOODLES                        READ WRITE NO
         6 DOLLS                          READ WRITE NO
SQL>
```

c. Reexecute the query of step 5).

```
SQL> CONNECT toys_owner@toys_root
Enter password: ******
Connected.
SQL> break on year skip 1
SQL> set pagesize 999
SQL> compute sum label "Yearly Revenue" of revenue on year
SQL> SELECT sum(a.revenue) revenue, a.year, a.region,
            con$name, cdb$name
     FROM  containers(toys_owner.sales_data) a
     GROUP BY a.year, a.region, con$name, cdb$name
     ORDER BY con$name, a.year, a.region;
  2    3    4    5
   REVENUE       YEAR REGION     CON$NAME        CDB$NAME
---------- ---------- ---------- --------------- ----------
     67079       2012 east       DOLLS           cdb2
     64237            north      DOLLS           cdb2
     65842            south      DOLLS           cdb2
     45385            west       DOLLS           cdb2
---------- **********
    242543 Yearly Rev

     54726       2013 east       DOLLS           cdb2
     51890            north      DOLLS           cdb2
     53857            south      DOLLS           cdb2
     54841              west       DOLLS            cdb2
---------- **********
    215314 Yearly Rev

     67347        2012 east       DOODLES          cdb2
     58157             north      DOODLES          cdb2
     66177             south      DOODLES          cdb2
     70955             west       DOODLES          cdb2
---------- **********
    262636 Yearly Rev

     52971        2013 east       DOODLES          cdb2
     55856             north      DOODLES          cdb2
     58163             south      DOODLES          cdb2
     63389             west       DOODLES          cdb2
---------- **********
    230379 Yearly Rev

     62966        2012 east       ROBOTS           ORCL
     63784             north      ROBOTS           ORCL
     60842             south      ROBOTS           ORCL
     65064             west       ROBOTS           ORCL
---------- **********
    252656 Yearly Rev

     62873        2013 east       ROBOTS           ORCL
     53893             north      ROBOTS           ORCL
     54644             south      ROBOTS           ORCL
     60023             west       ROBOTS           ORCL
---------- **********
    231433 Yearly Rev


24 rows selected.
SQL> EXIT
```

Practice 4-7: Dropping Unnecessary PDBs

Overview

In this practice, you will drop unnecessary PDBs and observe what happens when PDBs referenced by others, like proxy PDBs do, are dropped.

Tasks

1. Drop `toys_rr` referenced by the `px_toys_rr` proxy PDB.

```
$ . oraenv
ORACLE_SID = [cdb2] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB2                           MOUNTED
         4 TOYS_RR                        READ WRITE NO
         5 DOODLES                        READ WRITE NO
         6 DOLLS                          READ WRITE NO
SQL> ALTER PLUGGABLE DATABASE toys_rr CLOSE;

Pluggable database altered.

SQL>
```

   a. Find the application PDBs associated to the `toys_rr` application root to drop them first.

```
SQL> COL name FORMAT A14
SQL> SELECT name, con_id, application_root "APP_ROOT",
            application_seed "APP_Seed",
            application_pdb "APP_PDB",
            application_root_con_id "APP_ROOT_CONID"
     FROM   v$containers;
```

```
  2    3    4    5
NAME           CON_ID APP_ROOT APP_Seed APP_PDB  APP_ROOT_CONID
-------------- ------ -------- -------- -------- --------------
CDB$ROOT            1 NO       NO       NO
PDB$SEED            2 NO       NO       NO
PDB2                3 NO       NO       NO
TOYS_RR             4 YES      NO       NO
```

```
DOODLES                   5 NO        NO        YES                       4
DOLLS                     6 NO        NO        YES                       4

6 rows selected.

SQL>
```

b. Drop the application PDBs associated to the `toys_rr` application root.

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

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE dolls INCLUDING DATAFILES;

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE toys_rr INCLUDING DATAFILES;

Pluggable database dropped.

SQL>
```

2. Connect to the `px_toys_rr` proxy PDB, which used to reference the `toys_rr` application root.

```
SQL> CONNECT system@px_toys_rr
Enter password: ******
ERROR:
ORA-12514: TNS:listener does not currently know of service
requested in connect descriptor

SQL> EXIT
$
```

3. Drop the proxy PDB `px_toys_rr`.

```
$ . oraenv
ORACLE_SID = [cdb2] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE px_toys_rr CLOSE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE px_toys_rr INCLUDING DATAFILES;

```

```
Pluggable database dropped.

SQL> EXIT
$
```

4. Execute the `$HOME/labs/APP/cleanup_apps.sh` shell script to finish the PDBs cleanup.

```
$ $HOME/labs/APP/cleanup_apps.sh
$
```

5. Release `cdb2` resources.

```
$ . oraenv
ORACLE_SID = [cdb2] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

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

Practices for Lesson 5: Recovery and Flashback of PDBs

Chapter 5

Practices for Lesson 5: Overview

Practices Overview

In the following practices, you will perform recovery and flashback operations on PDBs.

- Recovering from a PDB `SYSTEM` datafile loss
- Flashing back an application container from the loss of a common application user
- Flashing back an application PDB from the loss of a local user
- Flashing back an application upgrade at different steps of the upgrade using restore points

Practice 5-1: Recovering from Essential PDB Datafiles Damage

Overview

In this practice, you will perform a CDB cold and hot backup of `ORCL` that you can use in case you lose all further backups or you cannot recover from a difficult situation.

Then you will perform a PDB recovery after the loss of the `SYSTEM` datafiles.

Tasks

1. Before starting the practice, execute the `$HOME/labs/admin/glogin_5.sh`, `$HOME/labs/APP/setup2_toys_app.sh`, and `$HOME/labs/APP/backup.sh` shell scripts. The first script sets formatting for all columns selected in queries, the second one creates the `toys_root` application container with two application PDBs, and the last one backs up `ORCL` in cold and hot mode.

```
$ $HOME/labs/admin/glogin_5.sh
$ $HOME/labs/APP/setup2_toys_app.sh
…
$ $HOME/labs/APP/backup.sh
…
$
```

2. Execute the `$HOME/labs/APP/crash.sh` shell script.

```
$ $HOME/labs/APP/crash.sh
…
$
```

Even if you see some error in removing files, continue to the next step.

a. As the application owner, you want to insert rows in the data-linked `toys_owner.codes` table. There is a strange error message.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ sqlplus toys_owner@toys_root
Enter password: ******
sqlplus toys_owner@toys_root

Enter password:
ORA-00604: error occurred at recursive SQL level 1
ORA-01116: error in opening database file 43
ORA-01110: data file 43:
'/u02/app/oracle/oradata/ORCL/toys_root/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
ORA-00604: error occurred at recursive SQL level 1
ORA-01116: error in opening database file 43
```

```
ORA-01110: data file 43:
'/u02/app/oracle/oradata/ORCL/toys_root/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3

SQL>
```

Q/ Can you work in the application PDBs of the application container while the application root has lost the `SYSTEM` datafile?

```
SQL> CONNECT system@robots

Enter password: ******
ERROR:
ORA-00604: error occurred at recursive SQL level 2
ORA-01116: error in opening database file 43
ORA-01110: data file 43:
'/u02/app/oracle/oradata/ORCL/toys_root/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3

SQL>
```

A/ The `SYSTEM` datafile of the application root is required for the application PDBs to work.
It may happen that the connection still works but when you need to access data-linked objects, the error comes out.

```
SQL> CONNECT system@robots
Enter password:
Connected.
SQL> SELECT * FROM toys_owner.codes;
SELECT * FROM toys_owner.codes
*
ERROR at line 1:
ORA-01116: error in opening database file 137
ORA-01110: data file 137:
'/u02/app/oracle/oradata/ORCL/toys_root/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3



SQL>
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHOW pdbs
```

```

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 TOYS_ROOT                      READ WRITE NO
         4 TOYS_ROOT$SEED                 READ ONLY  NO
         5 ROBOTS                         READ WRITE NO
         6 DOLLS                          READ WRITE NO
SQL>
```

b. In terminal window (2), proceed with `SYSTEM` datafile recovery in the `toys_root` PDB.

```
$ . oraenv
ORACLE_SID = [cdb2] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ rman target /

connected to target database: ORCL (DBID=1434391901)
RMAN> LIST FAILURE;
…
Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
2235       CRITICAL OPEN      20-MAR-16     System datafile 43:
'/u02/app/oracle/oradata/ORCL/toys_root/system01.dbf' is missing

RMAN>
```

```
RMAN> ADVISE FAILURE;
```

```
Database Role: PRIMARY

List of Database Failures
=========================


Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
2235       CRITICAL OPEN      20-MAR-16     System datafile 43:
'/u02/app/oracle/oradata/ORCL/toys_root/system01.dbf' is missing

analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
```

```
channel ORA_DISK_1: SID=745 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/toys_root/system01.dbf
was unintentionally renamed or moved, restore it
2. Automatic repairs may be available if you shutdown the
database and restart it in mount mode


Automated Repair Options
========================
Option Repair Description
------ ------------------
1      Restore and recover datafile 43
  Strategy: The repair includes complete media recovery with no
data loss
  Repair script:
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_2322832242.hm


RMAN>
```

```
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_2322832242.hm


contents of repair script:
   # restore and recover datafile
   sql 'TOYS_ROOT' 'alter database datafile 43 offline';
   restore ( datafile 43 );
   recover datafile 43;
   sql 'TOYS_ROOT' 'alter database datafile 43 online';


RMAN>
```

```
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_2322832242.hm

contents of repair script:
   # restore and recover datafile
   sql 'TOYS_ROOT' 'alter database datafile 43 offline';
   restore ( datafile 43 );
   recover datafile 43;
   sql 'TOYS_ROOT' 'alter database datafile 43 online';

Do you really want to execute the above repair (enter YES or
NO)? YES
executing repair script

sql statement: alter database datafile 43 offline
RMAN-00571: ===================================================
RMAN-00569: ========== ERROR MESSAGE STACK FOLLOWS ========
RMAN-00571: ===================================================
RMAN-03002: failure of repair command at 09/21/2016 12:20:33
RMAN-03015: error occurred in stored script Repair Script
RMAN-03009: failure of sql command on default channel at
09/21/2016 12:20:33
RMAN-11003: failure during parse/execution of SQL statement:
alter database datafile 137 offline
ORA-01541: system tablespace cannot be brought offline; shut
down if necessary

RMAN>
```

Before restoring and recovering the datafile, first close the PDB. Because the PDB cannot be closed normally, use the `CLOSE ABORT` clause.

In terminal window (1) within the `SYSDBA` session still opened, close the application root with the `ABORT` clause.

```
RMAN> ALTER PLUGGABLE DATABASE toys_root CLOSE ABORT;

Statement processed.

RMAN>
```

c. Now use the RMAN commands described in the DRA (Data Recovery Advisor) script.

```
RMAN> RESTORE (DATAFILE 43);
```

```
Starting restore at 20-MAR-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=251 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 00043 to
/u02/app/oracle/oradata/ORCL/toys_root/ORCL/system01.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/2E518B633BED646DE0532633
960A04B0/backupset/2016_03_18/o1_mf_nnndf_TAG20160318T112718_cgq
slnts_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/2E518B633BED646DE
0532633960A04B0/backupset/2016_03_18/o1_mf_nnndf_TAG20160318T112
718_cgqslnts_.bkp tag=TAG20160318T112718
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
Finished restore at 20-MAR-16

RMAN>
```

```
RMAN> RECOVER DATAFILE 43;

Starting recover at 20-MAR-16
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 63 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2016_03_18/o1
_mf_1_63_cgrf0vn4_.arc
…
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2016_03_
20/o1_mf_1_82_cgwh4ftt_.arc thread=1 sequence=82
media recovery complete, elapsed time: 00:01:52
Finished recover at 20-MAR-16

RMAN>
```

d. Back in terminal window (1), open `toys_root`.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 TOYS_ROOT                      MOUNTED
         4 TOYS_ROOT$SEED                 MOUNTED
         5 ROBOTS                         MOUNTED
         6 DOLLS                          MOUNTED
SQL> STARTUP
ORA-01113: file 45 needs media recovery
ORA-01110: data file 45:
'/u02/app/oracle/oradata/ORCL/toys_root/pdbseed_i1_undo.dbf'

SQL>
```

Q/ It can happen that some datafiles of the application PDBs require recovery.Which could the root cause of this situation?

A/ All application PDBs may be desynchronized with the application root and may require recovery before being opened. The `LIST FAILURE ALL` command would provide the full list of impacted datafiles and would suggest to recover them all in a script provided by the `ADVISE FAILURE ALL` command.

e. In terminal window (2), as requested, proceed with other tablespaces recovery by using DRA.

```
RMAN> LIST FAILURE ALL;
using target database control file instead of recovery catalog
Database Role: PRIMARY

List of Database Failures
=========================


Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
2347       CRITICAL OPEN      20-MAR-16     System datafile 52:
'/u02/app/oracle/oradata/ORCL/toys_root/dolls/system01.dbf'
needs media recovery
2332       CRITICAL OPEN      20-MAR-16     System datafile 49:
'/u02/app/oracle/oradata/ORCL/toys_root/robots/system01.dbf'
needs media recovery
989        HIGH     OPEN      20-MAR-16     One or more non-
system datafiles need media recovery
```

```
RMAN>
```

```
RMAN> ADVISE FAILURE ALL;
Database Role: PRIMARY

List of Database Failures
=========================


Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
2347       CRITICAL OPEN      20-MAR-16     System datafile 52:
'/u02/app/oracle/oradata/ORCL/toys_root/dolls/system01.dbf'
needs media recovery
2332       CRITICAL OPEN      20-MAR-16     System datafile 49:
'/u02/app/oracle/oradata/ORCL/toys_root/robots/system01.dbf'
needs media recovery
989        HIGH     OPEN      20-MAR-16     One or more non-
system datafiles need media recovery

analyzing automatic repair options; this may take some time
using channel ORA_DISK_1
analyzing automatic repair options complete

Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
=======================
1. If you restored the wrong version of data file
/u02/app/oracle/oradata/ORCL/toys_root/dolls/system01.dbf, then
replace it with the correct one
2. Automatic repairs may be available if you shutdown the
database and restart it in mount mode
3. If you restored the wrong version of data file
/u02/app/oracle/oradata/ORCL/toys_root/robots/system01.dbf, then
replace it with the correct one
4. If you restored the wrong version of data file
/u02/app/oracle/oradata/ORCL/toys_root/sysaux01.dbf, then
replace it with the correct one
5. If you restored the wrong version of data file
/u02/app/oracle/oradata/ORCL/toys_root/pdbseed_i1_undo.dbf, then
replace it with the correct one
```

```
6. If you restored the wrong version of data file
/u02/app/oracle/oradata/ORCL/toys_root/robots/sysaux01.dbf, then
replace it with the correct one
7. If you restored the wrong version of data file
/u02/app/oracle/oradata/ORCL/toys_root/robots/pdbseed_i1_undo.db
f, then replace it with the correct one
8. If you restored the wrong version of data file
/u02/app/oracle/oradata/ORCL/toys_root/dolls/sysaux01.dbf, then
replace it with the correct one
9. If you restored the wrong version of data file
/u02/app/oracle/oradata/ORCL/toys_root/dolls/pdbseed_i1_undo.dbf
, then replace it with the correct one

Automated Repair Options
========================
Option Repair Description
------ ------------------
1      Recover datafile 52; Recover datafile 49; Recover
datafile 46; ...
  Strategy: The repair includes complete media recovery with no
data loss
  Repair script:
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_3147366726.hm

RMAN>
```

```
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_3147366726.hm

contents of repair script:
   # recover datafile
   sql 'DOLLS' 'alter database datafile 52, 53, 54 offline';
   sql 'ROBOTS' 'alter database datafile 49, 50, 51 offline';
   sql 'TOYS_ROOT' 'alter database datafile 44, 45 offline';
   recover datafile 44, 45, 49, 50, 51, 52, 53, 54;
   sql 'DOLLS' 'alter database datafile 52, 53, 54 online';
   sql 'ROBOTS' 'alter database datafile 49, 50, 51 online';
   sql 'TOYS_ROOT' 'alter database datafile 44, 45 online';

RMAN>
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_3147366726.hm

contents of repair script:
   # recover datafile
   sql 'DOLLS' 'alter database datafile 52, 53, 54 offline';
   sql 'ROBOTS' 'alter database datafile 49, 50, 51 offline';
   sql 'TOYS_ROOT' 'alter database datafile 44, 45 offline';
   recover datafile 44, 45, 49, 50, 51, 52, 53,
   54;
   sql 'DOLLS' 'alter database datafile 52, 53, 54 online';
   sql 'ROBOTS' 'alter database datafile 49, 50, 51 online';
   sql 'TOYS_ROOT' 'alter database datafile 44, 45 online';

Do you really want to execute the above repair (enter YES or
NO)? YES
executing repair script

sql statement: alter database datafile 52, 53, 54 offline

sql statement: alter database datafile 49, 50, 51 offline

sql statement: alter database datafile 44, 45 offline

Starting recover at 20-MAR-16
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:02

Finished recover at 20-MAR-16

sql statement: alter database datafile 52, 53, 54 online

sql statement: alter database datafile 49, 50, 51 online

sql statement: alter database datafile 44, 45 online
repair failure complete
```

```
RMAN> EXIT
$
```

f. In terminal window (1), open `toys_root`.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected to:
SQL> STARTUP
Pluggable Database opened.
SQL>
```

g. Open the application PDBs of the `toys_root` application container.

```
SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 TOYS_ROOT                      READ WRITE NO
         4 TOYS_ROOT$SEED                 MOUNTED
         5 ROBOTS                         MOUNTED
         6 DOLLS                          MOUNTED
SQL> ALTER PLUGGABLE DATABASE all OPEN;
```

```
Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 TOYS_ROOT                      READ WRITE NO
         4 TOYS_ROOT$SEED                 READ WRITE NO
         5 ROBOTS                         READ WRITE NO
         6 DOLLS                          READ WRITE NO
SQL> CONNECT toys_owner@toys_root
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.sales_data WHERE  year=2000;

      YEAR REGION     QUAR    REVENUE CON_ID
---------- ---------- ---- ---------- ------
      2000 US         Q4            1      5
      2000 FRANCE     Q3            2      5
      2000 UK         Q2            3      5
      2000 GERMANY    Q1            4      6
```

```
       2000 US          Q4             1       6
       2000 FRANCE      Q3             2       6
       2000 UK          Q2             3       6
       2000 GERMANY     Q1             4       6
       2000 US          Q4             1       3
       2000 FRANCE      Q3             2       3
       2000 UK          Q2             3       3
       2000 GERMANY     Q1             4       3

12 rows selected.

SQL> EXIT
$
```

3. After a recovery, back up the whole CDB.

```
$ rman target /

RMAN> BACKUP DATABASE PLUS ARCHIVELOG delete all input;
…
RMAN> DELETE OBSOLETE;
…
RMAN> EXIT
$
```

Practice 5-2: Flashing Back an Application Container from the Loss of Application Common Users (*Optional*)

Overview

In this practice, after a common user has been dropped in the `toys_root` application container, you will flash back the `toys_root` application container to the time before the unintentional action.

Tasks

1. Find a common user to drop in the `toys_root` application container.

```
$ sqlplus sys@toys_root AS SYSDBA
Enter password: ******
Connected to:

SQL> SELECT username, common, con_id FROM cdb_users
     WHERE  common = 'YES' AND inherited = 'NO';
  2

USERNAME               COMMON CON_ID
---------------------- ------ ------
TOYS_OWNER             YES         3

SQL>
```

2. Set `ORCL` in FLASHBACK mode.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 TOYS_ROOT                      READ WRITE NO
         4 TOYS_ROOT$SEED                 READ WRITE NO
         5 ROBOTS                         READ WRITE NO
         6 DOLLS                          READ WRITE NO
SQL> SELECT flashback_on from V$DATABASE;

FLASHBACK_ON
------------------
NO

SQL> SHUTDOWN IMMEDIATE
```

```
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT
ORACLE instance started.

Total System Global Area 4697620480 bytes
Fixed Size                  2923760 bytes
Variable Size             989856528 bytes
Database Buffers          690987520 bytes
Redo Buffers               13852672 bytes
Database mounted.
SQL> ALTER SYSTEM SET
           DB_FLASHBACK_RETENTION_TARGET=2880 SCOPE=BOTH;
  2
System altered.

SQL> ALTER DATABASE FLASHBACK ON;

Database altered.

SQL> ALTER DATABASE OPEN;

Database altered.

SQL> SELECT flashback_on from V$DATABASE;

FLASHBACK_ON
------------------
YES

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 TOYS_ROOT                      MOUNTED
         4 TOYS_ROOT$SEED                 MOUNTED
         5 ROBOTS                         MOUNTED
         6 DOLLS                          MOUNTED
SQL>
```

3. Preserve the OPEN state for all PDBs except those that will be impacted by the `toys_owner` common user DROP operation.

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE ALL SAVE STATE;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 TOYS_ROOT                      READ WRITE NO
         4 TOYS_ROOT$SEED                 READ WRITE NO
         5 ROBOTS                         READ WRITE NO
         6 DOLLS                          READ WRITE NO
SQL>
```

4. Drop the `toys_owner` common user in the `toys_root` application container.
   a. Before dropping the application common user, retrieve the current timestamp.

```
SQL> SELECT timestamp_to_scn(current_timestamp)
     FROM v$database;
  2

TIMESTAMP_TO_SCN(CURRENT_TIMESTAMP)
-----------------------------------
                            2603794

SQL>
```

Q/ Does the user own common objects?

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT object_name, sharing, application FROM dba_objects
     WHERE  owner = 'TOYS_OWNER';
  2
OBJECT_NAME  SHARING       A
------------ ------------- -
SALES_DATA   METADATA LINK Y
CODES        OBJECT LINK   Y
```

```
SQL>
```

A/ Yes. The user owns the common tables shared by the different application PDBs in the application container.

b. Drop the user.

```
SQL> DROP USER toys_owner CASCADE;
DROP USER app CASCADE
*
ERROR at line 1:
ORA-65270: operation is not allowed in an application patch

SQL>
```

Q/ Why is the operation disallowed?

A/ A common user in an application can only be dropped if the operation is part of an application upgrade.

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                        BEGIN UPGRADE '1.0' TO '1.1';
  2
Pluggable database altered.

SQL> DROP USER toys_owner CASCADE;

User dropped.

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app
                        END UPGRADE TO '1.1';
  2
Pluggable database altered.

SQL> CONNECT sys@robots AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.
```

```
SQL> CONNECT sys@toys_root$seed AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.
SQL>
```

5. You realize you made a mistake by dropping the `toys_owner` user. Proceed with the flashback PDB operation.

   Q/ Do you have to shut the CDB down?

   A/ No. Oracle Database 12.2 allows you to flashback at PDB level.

   a. Close the PDB.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SHUTDOWN IMMEDIATE
Pluggable Database closed.
SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 TOYS_ROOT                      MOUNTED
         4 TOYS_ROOT$SEED                 MOUNTED
         5 ROBOTS                         MOUNTED
         6 DOLLS                          MOUNTED
SQL> EXIT
$
```

   Q/ How do you know that you closed the PDB and not shut down the CDB?

   A/ A CDB shutdown would have displayed more messages:

   Database closed.

   Database dismounted.

   b. Flashback `toys_app`.

```
$ rman target /
…
RMAN> FLASHBACK PLUGGABLE DATABASE toys_root TO SCN 2603794;

Starting flashback at 20-MAR-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=379 device type=DISK
```

```
starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished flashback at 20-MAR-16

RMAN>
```

c. Open the PDB in READ ONLY mode to review changes before opening the PDB with `RESETLOGS`.

```
RMAN> ALTER PLUGGABLE DATABASE toys_root OPEN READ ONLY;

Statement processed

RMAN> EXIT
$
```

```
$ sqlplus sys@toys_root AS SYSDBA
```

```
Enter password: ******
Connected.
SQL> SELECT username, common, con_id FROM cdb_users
     WHERE  username = 'TOYS_OWNER';
  2
USERNAME               COMMON CON_ID
---------------------- ------ ------
TOYS_OWNER             YES         3

SQL>
```

Q/ Does "PDB flashback" flash back only the application root?

A/ Yes. Opening an application root in read-only mode does not open the application PDBs.

```
SQL> SHOW pdbs

```

```
    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 TOYS_ROOT                      READ ONLY  NO
         4 TOYS_ROOT$SEED                 MOUNTED
         5 ROBOTS                         MOUNTED
         6 DOLLS                          MOUNTED
SQL> ALTER PLUGGABLE DATABASE all OPEN READ ONLY;

Warning: PDB altered with errors.

```

```
SQL> SELECT name, message, action FROM PDB_PLUG_IN_VIOLATIONS
     WHERE status='PENDING';

NAME
--------------------
MESSAGE
----------------------------------------------------------------
ACTION
----------------------------------------------------------------
TOYS_ROOT$SEED
Application version mismatch for application TOYS_APP:
Application PDB has version 1.1 NORMAL, Application Root does
not have the corresponding version.
Fix the application in the PDB or the application root

ROBOTS
Application version mismatch for application TOYS_APP:
Application PDB has version 1.1 NORMAL, Application Root does
not have the corresponding version.
Fix the application in the PDB or the application root

DOLLS
Application version mismatch for application TOYS_APP:
Application PDB has version 1.1 NORMAL, Application Root does
not have the corresponding version.
Fix the application in the PDB or the application root

SQL>
```

```
SQL> SELECT username, common, con_id FROM cdb_users
     WHERE  username = 'TOYS_OWNER';
```

```
  2
USERNAME               COMMON CON_ID
---------------------- ------ ------
TOYS_OWNER             YES         3

SQL>
```

```
SQL> CONNECT toys_owner@robots
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied
```

```
Warning: You are no longer connected to ORACLE.
SQL>
```

Q/ Why is the `toys_owner` user not known in the application PDB?

A/ The application PDBs have not been flashed back.

6. Flashback the application PDBs.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SHUTDOWN IMMEDIATE
Pluggable Database closed.
SQL> EXIT
$
$ rman target /
…
RMAN> FLASHBACK PLUGGABLE DATABASE "toys_root$seed" TO SCN
2603794;

Starting flashback at 20-MAR-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=505 device type=DISK


starting media recovery
media recovery complete, elapsed time: 00:00:07

Finished flashback at 20-MAR-16

RMAN>
```

```
RMAN> FLASHBACK PLUGGABLE DATABASE robots TO SCN 2603794;

Starting flashback at 20-MAR-16
using channel ORA_DISK_1


starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished flashback at 20-MAR-16

RMAN>
RMAN> FLASHBACK PLUGGABLE DATABASE dolls TO SCN 2603794;

Starting flashback at 20-MAR-16
using channel ORA_DISK_1


starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished flashback at 20-MAR-16

RMAN> EXIT
$
```

7. After verification, you decide to verify the data in application PDBs before closing, flashbacking, and opening the application container with RESETLOGS.
   a. Verify restored application data in application PDBs.

```
$ sqlplus sys@toys_root AS SYSDBA
Enter password: ******
Connected to:

SQL> ALTER PLUGGABLE DATABASE OPEN READ ONLY;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE all OPEN READ ONLY;

Pluggable database altered.

SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
```

```
SQL> SELECT * FROM toys_owner.sales_data WHERE  year=2000;

      YEAR REGION     QUAR    REVENUE
---------- ---------- ---- ----------
      2000 US         Q4            1
      2000 FRANCE     Q3            2
      2000 UK         Q2            3
      2000 GERMANY    Q1            4

SQL> SELECT code, label FROM codes;
```

```
      CODE LABEL
---------- ----------
         1 Puppet
         2 Car
```

```
SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
```

```
SQL> SELECT * FROM toys_owner.sales_data WHERE  year=2000;

      YEAR REGION     QUAR    REVENUE
---------- ---------- ---- ----------
      2000 US         Q4            1
      2000 FRANCE     Q3            2
      2000 UK         Q2            3
      2000 GERMANY    Q1            4

SQL> SELECT code, label FROM codes;

      CODE LABEL
---------- ----------
         1 Puppet
         2 Car

SQL>
```

b. Close the application container.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER PLUGGABLE DATABASE all CLOSE;
```

```

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 TOYS_ROOT                      READ ONLY  NO
         4 TOYS_ROOT$SEED                 MOUNTED
         5 ROBOTS                         MOUNTED
         6 DOLLS                              MOUNTED
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> EXIT
$
```

c. Flashback the application container.

```
$ rman target /
…
RMAN> FLASHBACK PLUGGABLE DATABASE toys_root TO SCN 2603794;

Starting flashback at 20-MAR-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=628 device type=DISK


starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished flashback at 20-MAR-16

RMAN>
RMAN> FLASHBACK PLUGGABLE DATABASE "toys_root$seed" TO SCN
2603794;

Starting flashback at 20-MAR-16
using channel ORA_DISK_1


starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished flashback at 20-MAR-16

RMAN> FLASHBACK PLUGGABLE DATABASE robots TO SCN 2603794;

Starting flashback at 20-MAR-16
using channel ORA_DISK_1
```

```
starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished flashback at 20-MAR-16

RMAN> FLASHBACK PLUGGABLE DATABASE dolls TO SCN 2603794;

Starting flashback at 20-MAR-16
using channel ORA_DISK_1


starting media recovery
media recovery complete, elapsed time: 00:00:03

Finished flashback at 20-MAR-16

RMAN> EXIT
$
```

d. Open the application container with `RESETLOGS`.

```
$ sqlplus sys@toys_root AS SYSDBA

Enter password: ******
Connected to:

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 TOYS_ROOT                      MOUNTED
         4 TOYS_ROOT$SEED                 MOUNTED
         5 ROBOTS                         MOUNTED
         6 DOLLS                          MOUNTED
SQL> ALTER PLUGGABLE DATABASE toys_root OPEN RESETLOGS;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE "toys_root$seed" OPEN RESETLOGS;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE robots OPEN RESETLOGS;
```

```
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE dolls OPEN RESETLOGS;

Pluggable database altered.

SQL>
```

e. Verify the restored application data in application PDBs.

```
SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
```

```
SQL> SELECT * FROM toys_owner.sales_data WHERE  year=2000;

      YEAR REGION     QUAR    REVENUE CON_ID
---------- ---------- ---- ---------- ------
      2000 US         Q4            1      5
      2000 FRANCE     Q3            2      5
      2000 UK         Q2            3      5
      2000 GERMANY    Q1            4      5
```

```

SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
```

```
SQL> SELECT count(*) FROM sales_data;

  COUNT(*)
----------
        32

SQL> SELECT code, label FROM codes;

      CODE LABEL
---------- ----------
         1 Puppet
         2 Car
```

```

SQL> EXIT
$
```

8. After a flashback, back up the CDB.

```
$ rman target /

RMAN> BACKUP DATABASE PLUS ARCHIVELOG delete all input;
…
RMAN> DELETE OBSOLETE;
…
RMAN> EXIT
$
```

Practice 5-3: Flashing Back Using Restore Points (*Optional*)

Overview

In this practice, you are testing the `toys_app` application upgrade in the `toys_root` application container. You may discover that application upgrade does not match what was expected. You would like to be able to revert the situation back to what it was before the application upgrade. For this purpose, you will use a restore point to flashback the `toys_root` application container to the time before the application upgrade was committed.

Tasks

1. If you did not complete Practice 5-2, set `ORCL` in FLASHBACK mode. Follow instructions in Practice 5-2, step 2.
2. Before starting the practice, execute the `$HOME/labs/APP/setup3_toys_app.sh` shell script. The script creates the `toys_root` application container with two application PDBs, installs the `toys_app` application, and backs up the PDBs.

```
$ $HOME/labs/APP/setup3_toys_app.sh
…
$
```

3. You plan to apply an application upgrade by executing the `@$HOME/labs/APP/script_upgrade_toys_app.sql` script. The script upgrades the `toys_app` application by creating the new application shared table, `toys_owner.categories`, in the `toys_root` application container. You want to be able to revert back to the situation before the application upgrade in case this does not match your requirements.
   a. Create a restore point before you upgrade the application.

      Q/ What is the advantage of performing PDB flashback using restore points?

      A/ Because the PDB has no outstanding transactions at the PDB restore point, a PDB flashback to a restore point requires neither restoring backups nor creating a clone instance.
      In application containers, all restore points are "CLEAN" with local undo, because the availability of local undo means that the effects of active transactions at the time of the restore point can be undone.

```
$ sqlplus / AS SYSDBA

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 TOYS_ROOT                      READ WRITE NO
         4 ROBOTS                         READ WRITE NO
         5 DOLLS                          READ WRITE NO
SQL> CREATE RESTORE POINT start_upgrade
           FOR PLUGGABLE DATABASE toys_root;
  2
```

```
Restore point created.

SQL>
```

Q/ How can you check that the restore point is created?

A/ Use the `V$RESTORE_POINT` view.

```
SQL> SELECT name, pdb_restore_point, clean_pdb_restore_point,
            con_id
     FROM   v$restore_point;
 2   3
NAME                 PDB CLE CON_ID
-------------------- --- --- ------
START_UPGRADE        YES NO       4

SQL>
```

b. Apply an application upgrade. The script synchronizes the application PDBs.

```
SQL> @$HOME/labs/APP/script_upgrade_toys_app.sql
…
SQL>
```

c. Connect to the application root and PDBs and check that the table is created.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.categories;

        C1 CATEGORY
---------- --------------------
         1 GAMES
         2 PUPPETS
         3 VEHICLES

SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.categories;

        C1 CATEGORY
---------- --------------------
         1 GAMES
         2 PUPPETS
         3 VEHICLES

```

```
SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.categories;

        C1 CATEGORY
---------- --------------------
         1 GAMES
         2 PUPPETS
         3 VEHICLES

SQL>
```

4. Now, you decide to reset the application container back to what it was before the `toys_app` application upgrade.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE toys_root CLOSE;

Pluggable database altered.

SQL> EXIT
$
```

```
$ export NLS_DATE_FORMAT='DD-MM-YYYY HH:MI:SS'
$ rman target /

RMAN> FLASHBACK PLUGGABLE DATABASE toys_root TO RESTORE POINT
start_upgrade;

Starting flashback at 21-03-2016 12:37:54
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=746 device type=DISK


starting media recovery

archived log for thread 1 with sequence 93 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2016_03_21/o1
_mf_1_93_cgyjjbgh_.arc
media recovery complete, elapsed time: 00:00:07
Finished flashback at 21-03-2016 12:38:11
```

```

RMAN>
```

5. Open the application root and the application PDBs.

```
RMAN> ALTER PLUGGABLE DATABASE toys_root OPEN RESETLOGS;

Statement processed

RMAN> ALTER PLUGGABLE DATABASE robots OPEN;

Statement processed

RMAN> ALTER PLUGGABLE DATABASE dolls OPEN;

Statement processed

RMAN> EXIT
$
```

a. Connect to the application root and PDBs and check that the table has been dropped as it was before the application upgrade.

```
$ sqlplus sys@toys_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.categories;
SELECT * FROM toys_owner.categories
                         *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> CONNECT toys_owner@robots
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.categories;
SELECT * FROM toys_owner.categories
                         *
ERROR at line 1:
ORA-65047: object TOYS_OWNER.CATEGORIES is invalid or compiled
with errors in root.

SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.categories;
```

```
SELECT * FROM toys_owner.categories
                        *
ERROR at line 1:
ORA-65047: object TOYS_OWNER.CATEGORIES is invalid or compiled
with errors in root.

SQL>
```

Q1/ Why is the error message in application PDBs different from the error in the application root?

A1/ Checking with "`OERR ORA 65047`" command:

65047, 00000, "Object %s.%s is invalid or compiled with errors in root."

// *Cause: An attempt was made to issue a metadata link DDL for an object

//   that was invalid or compiled with errors in a CDB$ROOT or an

//   application root.

// *Action: Check the validity of the object in CDB$ROOT or application root.

relates that Oracle checks the table's possible existence in the CDB root because it does not find it in the application root.

Q2/ The flashback operation flashed back the application root and therefore dropped the `categories` table. Why are the application PDBs still at the state of the end of the application upgrade?

A2/ The application PDBs were not flashed back. To keep the application consistent all across the application PDBs, namely for the common tables of the application, flashback the application PDBs too.

b. Reset the application PDBs back to what they were before the `toys_app` application upgrade.

1) Close the application container.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE toys_root CLOSE;

Pluggable database altered.

SQL> EXIT
$
```

2) Flashback the application PDBs.

```
$ rman target /

RMAN> FLASHBACK PLUGGABLE DATABASE robots TO RESTORE POINT
start_upgrade;

Starting flashback at 21-03-2016 01:05:27
using channel ORA_DISK_1
```

```
starting media recovery

archived log for thread 1 with sequence 93 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2016_03_21/o1
_mf_1_93_cgyjjbgh_.arc
media recovery complete, elapsed time: 00:00:01
Finished flashback at 21-03-2016 01:05:32

RMAN> FLASHBACK PLUGGABLE DATABASE dolls TO RESTORE POINT
start_upgrade;

Starting flashback at 21-03-2016 01:05:36
using channel ORA_DISK_1


starting media recovery

archived log for thread 1 with sequence 93 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2016_03_21/o1
_mf_1_93_cgyjjbgh_.arc
media recovery complete, elapsed time: 00:00:03
Finished flashback at 21-03-2016 01:05:46

RMAN>
```

6. Open the application root and the application PDBs.

```
RMAN> ALTER PLUGGABLE DATABASE toys_root OPEN;

Statement processed

RMAN> ALTER PLUGGABLE DATABASE robots OPEN RESETLOGS;

Statement processed

RMAN> ALTER PLUGGABLE DATABASE dolls OPEN RESETLOGS;

Statement processed

RMAN> EXIT
$
```

7. Verify that the `toys_app` application is as it was before the application upgrade.

```
$ sqlplus toys_owner@robots
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.categories;
SELECT * FROM toys_owner.categories
                         *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> CONNECT toys_owner@dolls
Enter password: ******
Connected.
SQL> SELECT * FROM toys_owner.categories;
SELECT * FROM toys_owner.categories
                         *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
```

Q/ You completed the application upgrade using restore points. What should be not forgotten to avoid performance degradation?

A/ Drop the restore points which increase the flashback log volume.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> DROP RESTORE POINT start_upgrade
          FOR PLUGGABLE DATABASE toys_root;
  2
Restore point dropped.

SQL> EXIT
$
```

8. After a flashback, back up the CDB.

```
$ rman target /

RMAN> BACKUP DATABASE PLUS ARCHIVELOG delete all input;
…
RMAN> DELETE OBSOLETE;
…
RMAN> EXIT
$
```

Practices for Lesson 6: Managing Performance in CDBs and PDBs

Chapter 6

Practices for Lesson 6: Overview

Practices Overview

In these practices, you monitor performance at CDB and PDB levels, run ADDM and get recommendations at CDB and PDB levels, monitor and tune SQL execution based on application shared objects in application PDBs, and finally use performance profiles to limit CPU resources between application PDBs.

Practice 6-1: Monitoring Performance at CDB and PDB Levels

Overview

In this practice, you monitor the resources for the CDB and PDBs by using EM Database Express.

Tasks

1. Before starting the practice, execute the `$HOME/labs/admin/glogin_6.sh` shell script. It appends `COL` commands to format all columns selected in queries.

```
$ $HOME/labs/admin/glogin_6.sh
$
```

2. Then use the `$HOME/labs/APP/setup_tuning.sh` shell script to create shared application tables in the `hr_root` application root for the `hr_app` application. The script may take a few minutes to load data in data-linked and metadata-linked tables.

```
$ $HOME/labs/APP/setup_tuning.sh
…
$
```

3. Then you start an application workload in `sales` and `research`.

```
$ cd $HOME/labs/APP
$ $HOME/labs/APP/start_workload.sh 1 sales
$ $HOME/labs/APP/start_workload.sh 1 research
$
```

   Until you remove the `$HOME/labs/APP/runload` file, the workload continues.

4. Check whether Enterprise Manager Database Express is configured for `ORCL`.
   a. Verify that the value of the `DISPATCHERS` instance parameter is set to `(PROTOCOL=TCP)(SERVICE=ORCLXDB)` in the `ORCL` instance.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ sqlplus / AS SYSDBA
Connected.
SQL> SHOW PARAMETER dispatchers
NAME                 TYPE       VALUE
-------------------- ---------- -------------
dispatchers          string      (PROTOCOL=TCP)(SERVICE=ORCLXDB)
dispatchers_mode                 string
max_dispatchers      integer

SQL>
```

   b. Select the port number used for Enterprise Manager Database Express. If there is none configured, use port 5500.

```
SQL> SELECT dbms_xdb_config.gethttpsport FROM DUAL;
```

```

GETHTTPSPORT
------------
        5500

SQL> EXEC dbms_xdb_config.sethttpsport(5500)

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

c. Verify that the listener is running and listens to the localhost (*yourserver*) using TCP protocol, the port 5500 for `ORCL`, the http presentation with RAW session data.

```
$ lsnrctl status
…
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<your
hostname>)(PORT=5500))(Security=(my_wallet_directory=/u01/app/or
acle/admin/ORCL/xdb_wallet))(Presentation=HTTP)(Session=RAW))
Services Summary...
…
The command completed successfully
$
```

d. Launch a browser and use the following URL https://localhost:5500/em.

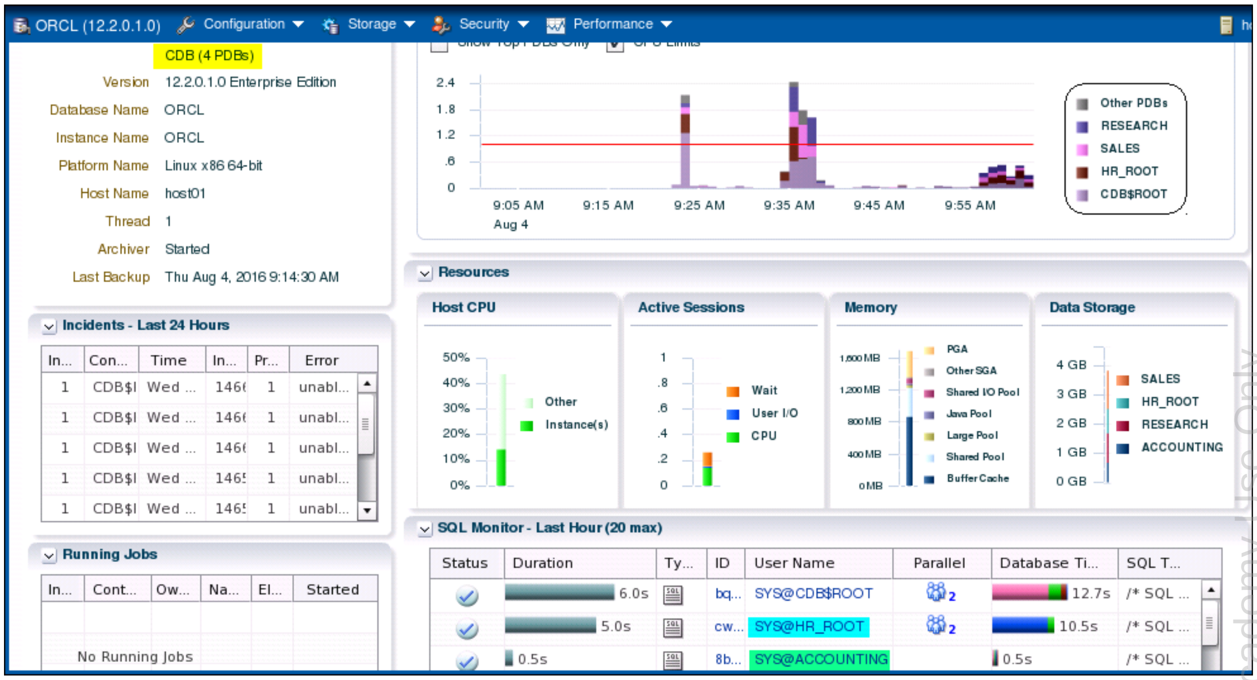
e. Most probably, you receive a Secure Connection Failed message and you need to add a security exception. At the end of the alert box, click **I Understand the Risks**.

f. At the bottom of the page, click **Add Exception**.

g. Confirm that "Permanently store this exception" is selected in your training environment and click **Confirm Security Exception**.

h. Enter **`system`** in the User Name field. Enter the password in the Password field. Then click **Login**.

Observe that the `ORACLE_HOME` for the database instance is 12.2.0.1 and the database name is `ORCL`.



Version	12.2.0.1.0 Enterprise Edition
Database Name	ORCL
Instance Name	ORCL
Platform Name	Linux x86 64-bit
Host Name	host01
Thread	1
Archiver	Started
Last Backup	Thu Aug 4, 2016 9:14:30 AM

Incidents - Last 24 Hours

In...	Con...	Time	In...	Pr...	Error
1	CDB$I	Wed ...	146(	1	unabl...
1	CDB$I	Wed ...	146(	1	unabl...
1	CDB$I	Wed ...	146(	1	unabl...
1	CDB$I	Wed ...	1465	1	unabl...
1	CDB$I	Wed ...	1465	1	unabl...

Running Jobs

In...	Cont...	Ow...	Na...	El...	Started

No Running Jobs

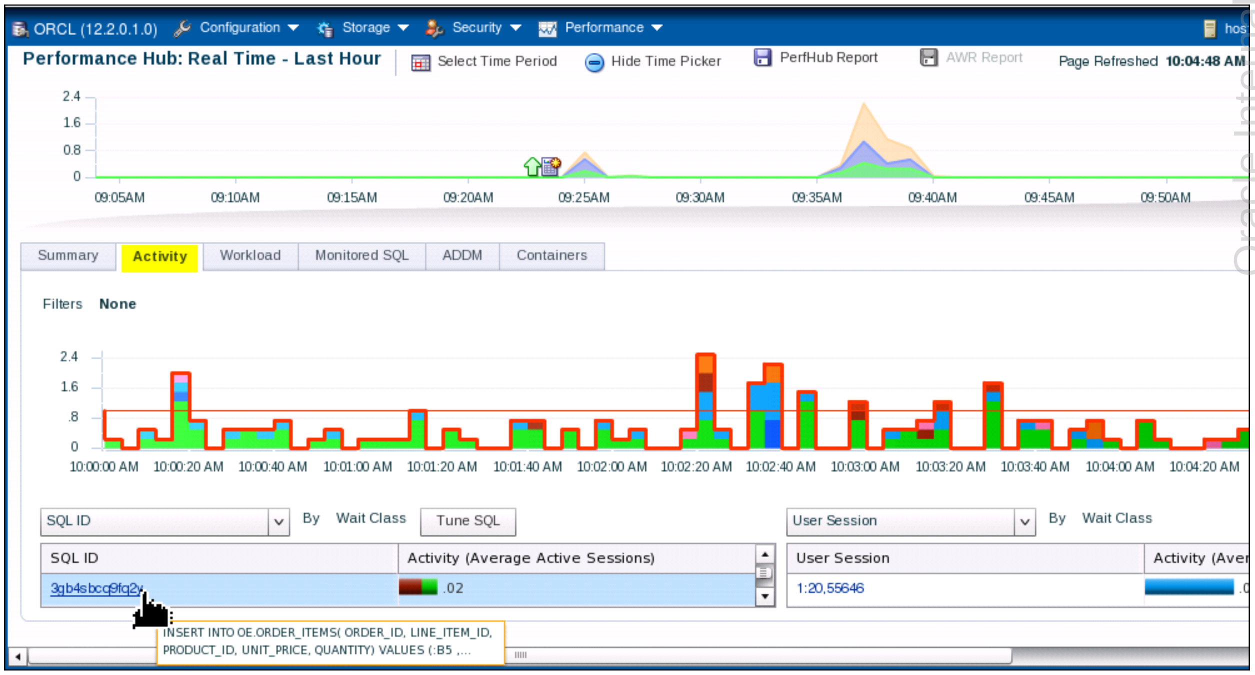
SQL Monitor - Last Hour (20 max)

Status	Duration	Ty...	ID	User Name	Parallel	Database Ti...	SQL T...
✓	6.0s	SQL	bq...	SYS@CDB$ROOT	2	12.7s	/* SQL ...
✓	5.0s	SQL	cw...	SYS@HR_ROOT	2	10.5s	/* SQL ...
✓	0.5s	SQL	8b...	SYS@ACCOUNTING		0.5s	/* SQL ...

Q/ In the Performance pane on the top right, in the Containers tab (third tab), what happens when you move your mouse on the name of one of the PDBs?

A/ You have a global idea of resources consumed by each PDB and at which time.

i. To get all statements executed in all containers, click the "Performance Hub" option in the "Performance" menu. Then click the "Activity" tab.



Performance Hub: Real Time - Last Hour

Page Refreshed **10:04:48 AM**

Filters **None**

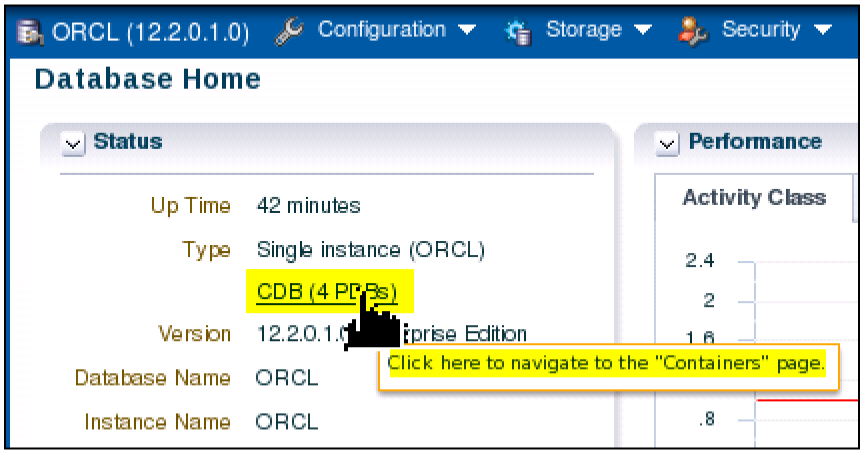
SQL ID	Activity (Average Active Sessions)
3gb4sbcq9fq2y	.02

User Session	Activity (Aver
1:20,55646	.0

INSERT INTO OE.ORDER_ITEMS(ORDER_ID, LINE_ITEM_ID, PRODUCT_ID, UNIT_PRICE, QUANTITY) VALUES (:B5 ,...

Q1/ Do you get details on Host CPU consumed by each PDB?

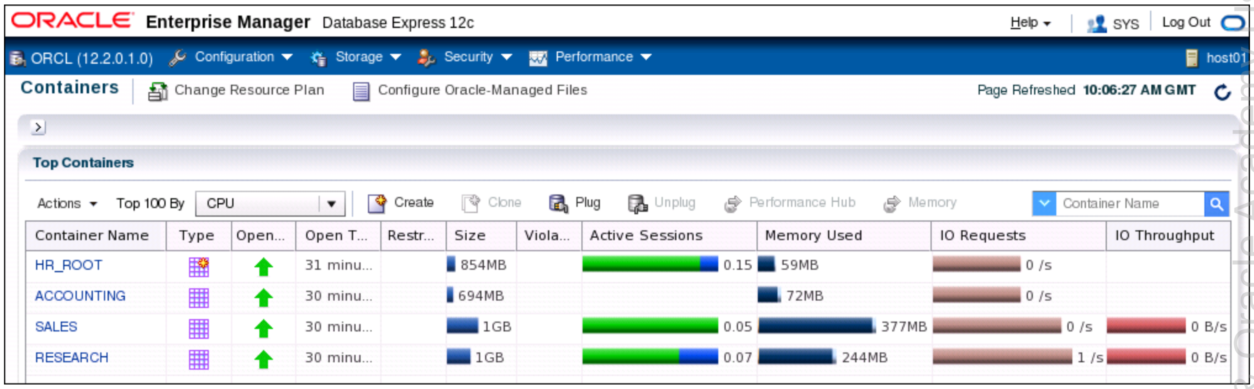
A1/ You will not get details on Host CPU consumed by each PDB from the current page, nor from the Database Home page. Click the `ORCL` link at the top left corner of the window.



Up Time	42 minutes
Type	Single instance (ORCL)
	CDB (4 PDBs)
Version	12.2.0.1.0 ... prise Edition
Database Name	ORCL
Instance Name	ORCL

Click here to navigate to the "Containers" page.

Q2/ How would you get information on resources consumed for a specific PDB?



Page Refreshed 10:06:27 AM GMT

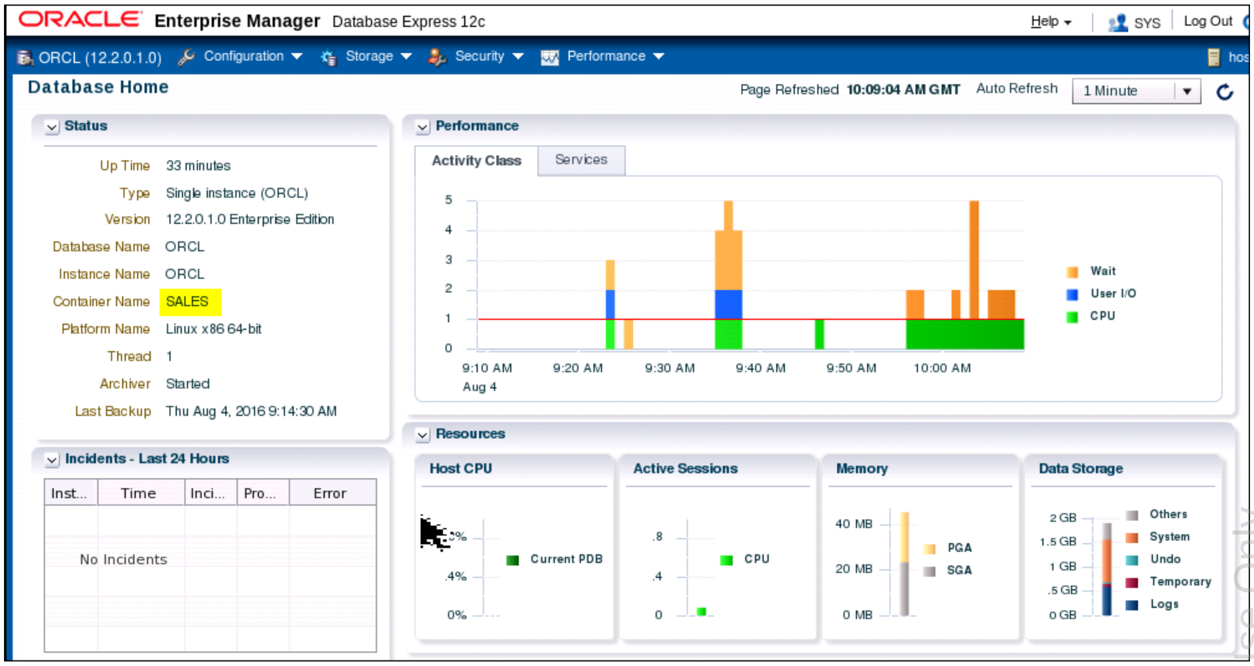
Top 100 By CPU

Container Name	Type	Open...	Open T...	Restr...	Size	Viola...	Active Sessions	Memory Used	IO Requests	IO Throughput
HR_ROOT	▦	↑	31 minu...		854MB		0.15	59MB	0 /s	
ACCOUNTING	▦	↑	30 minu...		694MB			72MB	0 /s	
SALES	▦	↑	30 minu...		1GB		0.05	377MB	0 /s	0 B/s
RESEARCH	▦	↑	30 minu...		1GB		0.07	244MB	1 /s	0 B/s

A2/ To get the summary on resources consumed by each PDB, click the link `CDB(4 PDBs)`.

Then to get detailed information for each PDB, click the PDB name in the screenshot above.

Below is the Performance Hub of the `sales` PDB.


ORACLE Enterprise Manager Database Express 12c

Help ▾ | SYS | Log Out

ORCL (12.2.0.1.0) | Configuration ▾ | Storage ▾ | Security ▾ | Performance ▾

Database Home

Page Refreshed **10:09:04 AM GMT** Auto Refresh 1 Minute

Status

Up Time	33 minutes
Type	Single instance (ORCL)
Version	12.2.0.1.0 Enterprise Edition
Database Name	ORCL
Instance Name	ORCL
Container Name	SALES
Platform Name	Linux x86 64-bit
Thread	1
Archiver	Started
Last Backup	Thu Aug 4, 2016 9:14:30 AM

Incidents - Last 24 Hours

Inst...	Time	Inci...	Pro...	Error
No Incidents				

5. Stop the workload by removing the `$HOME/labs/APP/runload` file.

```
$ rm runload
```

```
$
```

Practice 6-2: Getting Performance Recommendations at CDB and PDB Levels

Overview

In this practice, you will ask for ADDM recommendations for `ORCL` as you were used to doing in Oracle Database 12.1.

In Oracle Database 12.2, a centralized AWR continues to serve as the repository for the performance data for the whole database—CDB root container and all its PDBs. Oracle Enterprise Manager now provides the ability to transfer the performance data from Automatic Workload Repository across all enterprise databases into a central performance warehouse called AWR Warehouse. This is covered in an Enterprise Manager Cloud Control course.

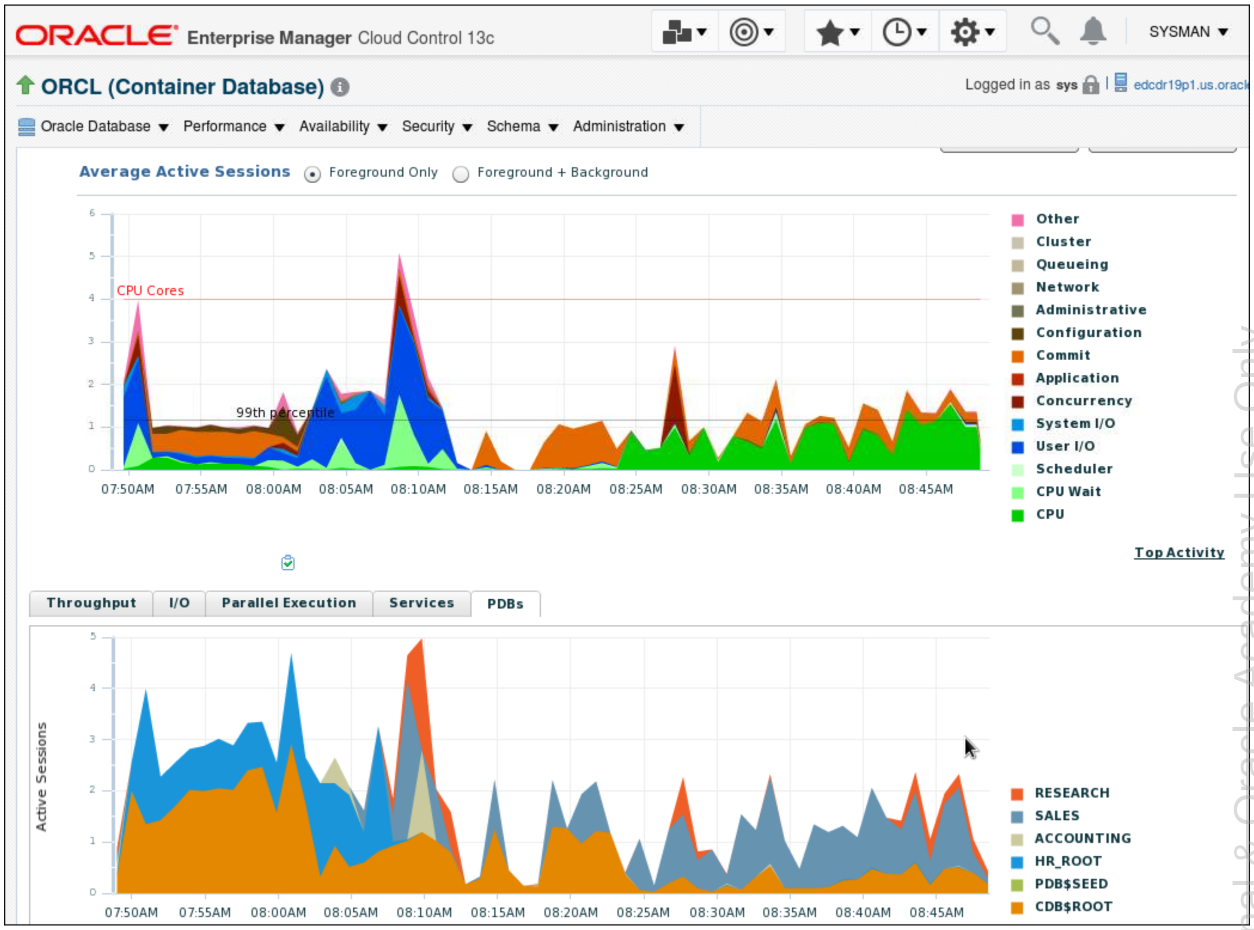
Oracle Database 12.2 allows AWR data to be collected, viewed, and managed from both the CDB root level as well as the PDB level.

Tasks

1. You have the choice to still work with EM Express or use EM Cloud Control. If you want to see how to proceed with EM Cloud Control, click the Firefox icon on the top panel (toolbar region) above the desktop to open a browser to access the Enterprise Manager Cloud Control console. Enter the URL for Cloud Control: `https://<em_server_hostname>.<domain>:7802/em`. In the current setup, use https://localhost:7802/em. Enter **`sysman`** in the User Name field. Then click Login.
2. Access the performance information in `ORCL`:
   a. Click Targets, and then Databases.
   b. To get the list of databases, click Search List. Click the `ORCL` link.
3. ***First ADDM test***: Collect ADDM recommendations for `ORCL`. The operation creates a CDB level snapshot. This type of collection is called 'CDB level AWR'. A CDB level snapshot is one whose main objective is getting the snapshot of the statistics that matter at the global level. Prepare two terminal windows.
   a. In another terminal window (we call it *Window1*), start a workload in application PDBs.

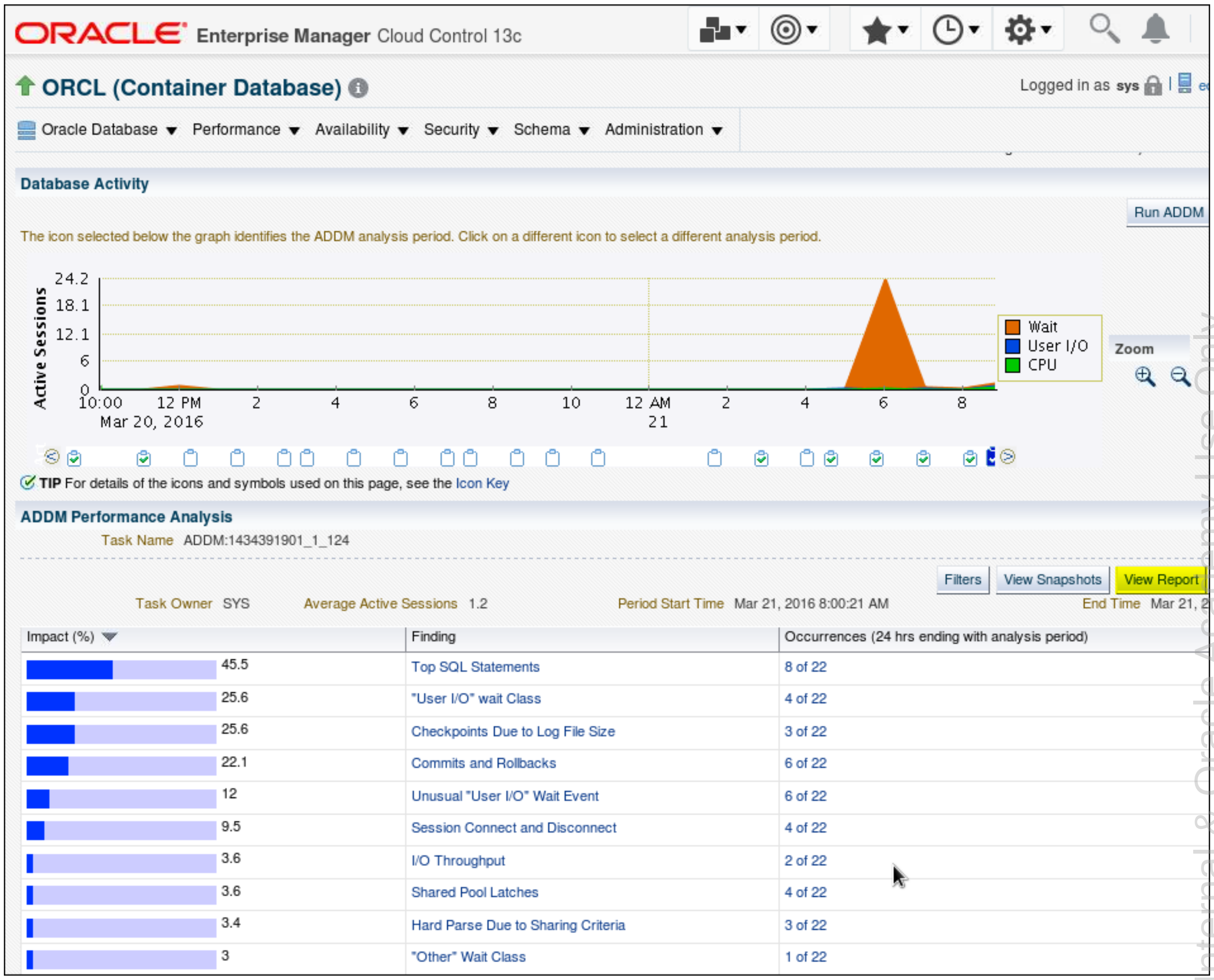
```
$ $HOME/labs/APP/loop.sh
…
$
```

b. In EM Cloud Control, once connected to `ORCL`, click “Performance Home” from the Performance menu. The `ORCL_SYS` preferred credentials appear. Click Login. To display the active sessions per container, click the “PDBs” tab in the “Active Sessions” bottom section. When you see the load growing, click “Run ADDM Now”.



c. To the “Are you sure you want to create a new AWR snapshot and run ADDM on this and the previous snapshot?” message, click Yes. The ADDM analysis task is processing.

1) When the ADDM analysis task is completed, from the “ADDM Performance Analysis” section, click “View Report” to display the recommendations.



The icon selected below the graph identifies the ADDM analysis period. Click on a different icon to select a different analysis period.

TIP For details of the icons and symbols used on this page, see the Icon Key

ADDM Performance Analysis

Task Name ADDM:1434391901_1_124

Task Owner SYS   Average Active Sessions 1.2   Period Start Time Mar 21, 2016 8:00:21 AM   End Time Mar 21, 2

Impact (%)	Finding	Occurrences (24 hrs ending with analysis period)
45.5	Top SQL Statements	8 of 22
25.6	"User I/O" wait Class	4 of 22
25.6	Checkpoints Due to Log File Size	3 of 22
22.1	Commits and Rollbacks	6 of 22
12	Unusual "User I/O" Wait Event	6 of 22
9.5	Session Connect and Disconnect	4 of 22
3.6	I/O Throughput	2 of 22
3.6	Shared Pool Latches	4 of 22
3.4	Hard Parse Due to Sharing Criteria	3 of 22
3	"Other" Wait Class	1 of 22

Q/ Are the recommendations related to CDB level only?

```
Analysis Target
---------------
Database 'ORCL' with DB ID 1434391901.
Database version 12.2.0.1.0.
ADDM performed an analysis of instance ORCL, numbered 1 and
hosted at your_server.
…
          Findings and Recommendations
          ----------------------------
```

`Finding 1: Top SQL Statements`

…

`Recommendation 4: SQL Tuning`

```
   Estimated benefit is .03 active sessions, 2.64% of total
activity.
   ------------------------------------------------------------
   Action
      Run SQL Tuning Advisor on the SELECT statement with SQL_ID
      "2bxhpf2vfhqnu".
      Related Object
         SQL statement with SQL_ID 2bxhpf2vfhqnu.
         SELECT /*+ monitor USE_NL(d l pi i)
         FULL(d) FULL(l) FULL(pi) FULL(i) */
         l.order_id, SUM(unit_price * quantity) amount
         FROM   oe.orders d , oe.order_items l,
         oe.product_information pi, oe.inventories i
         WHERE  d.order_id = l.order_id
         AND    pi.product_id = l.product_id
         AND    pi.product_id = i.product_id
         AND    d.order_date  100
         GROUP BY l.order_id
   Rationale
      The SQL statement executed in container SALES with
database ID
      403617970.
…
  Recommendation 5: SQL Tuning
   Estimated benefit is .02 active sessions, 2.05% of total
activity.
   -----------------------------------------------------------
   Action
      Run SQL Tuning Advisor on the SELECT statement with SQL_ID
      "2bxhpf2vfhqnu".
      Related Object
         SQL statement with SQL_ID 2bxhpf2vfhqnu.
         SELECT /*+ monitor USE_NL(d l pi i)
         FULL(d) FULL(l) FULL(pi) FULL(i) */
         l.order_id, SUM(unit_price * quantity) amount
         FROM   oe.orders d , oe.order_items l,
         oe.product_information pi, oe.inventories i
         WHERE  d.order_id = l.order_id
         AND    pi.product_id = l.product_id
         AND    pi.product_id = i.product_id
         AND    d.order_date  100
         GROUP BY l.order_id
   Rationale
```

```
      The SQL statement executed in container RESEARCH with
database ID
      1533873697.
…
Finding 3: Checkpoints Due to Log File Size
Impact is .31 active sessions, 25.62% of total activity.
--------------------------------------------------------
Buffer cache writes due to small log files were consuming
significant database
time.

   Recommendation 1: Database Configuration
   Estimated benefit is .31 active sessions, 25.62% of total
activity.
   -----------------------------------------------------------
   Action
      Increase the size of the log files to 72 M to hold at
least 20 minutes of redo information.
 …
Finding 5: Unusual "User I/O" Wait Event
  Recommendation 3: Application Analysis
   Estimated benefit is .15 active sessions, 11.97% of total
activity.
   -----------------------------------------------------------
   Action
      Investigate the cause for high "Pluggable Database file
copy" waits in Service "hr_root".
   Rationale
      The session connected to container HR_ROOT with database
ID 1379244433.
…
```

A/ No. Recommendations are reported for CDB and PDB levels because AWR snapshots contain statistics related to CDB and PDBs. View the recommendations related to statements executed in application PDBs.

4. Although ‘PDB level AWR’ and ‘CDB level AWR’ data is stored in the CDB root `SYSAUX` tablespace, data can be viewed for specific PDBs or CDB root only or both. *`CDB_HIST_`xxx* views still point to the union of all global and local snapshots.

```
$ sqlplus / AS SYSDBA
Enter password: ******
Connected to:
SQL> SELECT table_name FROM dict
     WHERE  table_name like '%CDB%HIST%SNAP%' ORDER BY 1;
  2
TABLE_NAME
```

```
----------------------
CDB_HIST_ASH_SNAPSHOT
CDB_HIST_PDB_IN_SNAP
CDB_HIST_SNAPSHOT
CDB_HIST_SNAP_ERROR

SQL>
```

5. Find *AWR_xxx* new views related to AWR PDB level snapshots.

```
SQL> SELECT view_name FROM dba_views
     WHERE  view_name LIKE '%AWR%_PDB%' ORDER BY 1;
  2
VIEW_NAME
--------------------------------------------------------------
AWR_PDB_ACTIVE_SESS_HISTORY
AWR_PDB_APPLY_SUMMARY
AWR_PDB_ASH_SNAPSHOT
AWR_PDB_ASM_BAD_DISK
AWR_PDB_ASM_DISKGROUP
…
AWR_PDB_PDB_IN_SNAP
…
AWR_PDB_WAITSTAT
AWR_PDB_WR_CONTROL
AWR_ROOT_PDB_INSTANCE
AWR_ROOT_PDB_IN_SNAP
AWR_ROOT_RSRC_PDB_METRIC

144 rows selected.

SQL>
```

6. Find *AWR_xxx* new views related to AWR CDB root level snapshots.

```
SQL> SELECT view_name FROM dba_views
     WHERE  view_name LIKE '%AWR%_ROOT%' ORDER BY 1;
  2
VIEW_NAME
--------------------------------------------------------------
AWR_ROOT_ACTIVE_SESS_HISTORY
AWR_ROOT_APPLY_SUMMARY
AWR_ROOT_ASH_SNAPSHOT
AWR_ROOT_ASM_BAD_DISK
…
AWR_ROOT_UNDOSTAT
```

```
AWR_ROOT_WAITCLASSMET_HISTORY
AWR_ROOT_WAITSTAT
AWR_ROOT_WR_CONTROL

142 rows selected.

SQL>
```

Q/ Which categories of AWR views do you discover?

A/ There are two categories of AWR views: AWR_PDB_xxx for PDB level snapshots and AWR_ROOT_xxx for all CDB snapshots.
CDB_HIST_xxx views display data for both CDB root level and PDB level snapshots.

d. Connect to `sales` and create a new snapshot. There are four levels of collections: `BESTFIT`, `LITE`, `TYPICAL`, and `ALL`.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT dbid FROM v$pdbs;

      DBID
----------
2401763088

SQL> SELECT snap_id, dbid, con_dbid, con_id
     FROM   awr_root_pdb_in_snap ORDER BY 1;
  2
   SNAP_ID       DBID   CON_DBID CON_ID
---------- ---------- ---------- ------
       144 1434391901 2401763088      5
       145 1434391901 2401763088      5

SQL> exec dbms_workload_repository.create_snapshot(FLUSH_LEVEL
=>  'BESTFIT',  DBID => 2401763088)

PL/SQL procedure successfully completed.

SQL> SELECT snap_id, dbid, con_dbid, con_id
     FROM   awr_root_pdb_in_snap ORDER BY 1;
  2
   SNAP_ID       DBID   CON_DBID CON_ID
---------- ---------- ---------- ------
       144 1434391901 2401763088      5
       145 1434391901 2401763088      5
```

```
SQL> SELECT snap_id, dbid, con_dbid, con_id
     FROM   awr_pdb_pdb_in_snap;
  2
```

```
   SNAP_ID       DBID   CON_DBID CON_ID
---------- ---------- ---------- ------
         1 2401763088 2401763088      5

SQL> SELECT snap_id, dbid, con_dbid, con_id
     FROM   cdb_hist_pdb_in_snap;
  2
   SNAP_ID       DBID   CON_DBID CON_ID
```

```
---------- ---------- ---------- ------
         1 2401763088 2401763088      5

SQL>
```

```
SQL> CONNECT / AS SYSDBA
Connected.
```

```
SQL> SELECT snap_id, dbid, con_dbid, con_id
     FROM   cdb_hist_pdb_in_snap;
  2
   SNAP_ID       DBID   CON_DBID CON_ID
---------- ---------- ---------- ------
         1 2401763088 2401763088      5
…
       144 1434391901 2401763088      5
       144 1434391901 3942641642      6
       145 1434391901 1520029004      3
       145 1434391901 2381044334      4
       145 1434391901 2401763088      5
       145 1434391901 3942641642      6

655 rows selected.
```

```
SQL>
```

Q/ What can you conclude at this step of the observation?

A/ When you are connected to a PDB, the `CDB_HIST_PDB_IN_SNAP` view displays the new snapshots created at the PDB level and whose data pertain to the PDB. When you are connected in the CDB root, the `CDB_HIST_PDB_IN_SNAP` view displays all snapshots created at the CDB root level and whose data pertain to containers that were opened when the snapshots were created.

When you are connected to a PDB, the `awr_pdb_pdb_in_snap` view displays the PDB level snapshots created in the PDB and the `awr_root_pdb_in_snap` view displays the snapshots for the PDB created when the snapshots were created at the CDB root level and the PDB was opened.

e. Create another snapshot.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> exec dbms_workload_repository.create_snapshot(FLUSH_LEVEL
=>  'BESTFIT',  DBID => 2401763088)

PL/SQL procedure successfully completed.

SQL> SELECT snap_id, dbid, con_dbid, con_id
     FROM   cdb_hist_pdb_in_snap;
  2
   SNAP_ID       DBID   CON_DBID CON_ID
---------- ---------- ---------- ------
         1 2401763088 2401763088      5
         2 2401763088 2401763088      5

SQL>
```

Q/ Did the PDB-level AWR snapshot collection performed in the `sales` PDB whose DBID is 2401763088 collect AWR data for this single PDB?

```
SQL> CONNECT sys@research AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT snap_id, dbid, con_dbid, con_id
     FROM   cdb_hist_pdb_in_snap;
  2
no rows selected

SQL>
```

A/ Yes. It did not collect data for other PDBs like `research`. Nevertheless the snapshot created is stored in the `SYSAUX` tablespace of the CDB root and contains AWR data that pertains to `sales` only.

f. Ask ADDM to analyze the period between the last two PDB level snapshots.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> var task_name VARCHAR2(60)
SQL> DECLARE
       taskid NUMBER;
```

```
     BEGIN
       dbms_advisor.create_task('ADDM',taskid,:task_name);
       dbms_advisor.set_task_parameter(:task_name,
                         'START_SNAPSHOT', 1);
       dbms_advisor.set_task_parameter(:task_name,
                         'END_SNAPSHOT', 2);
       dbms_advisor.set_task_parameter(:task_name,
                         'DB_ID', 2401763088);
       dbms_advisor.execute_task(:task_name);
     END;
/
  2    3    4    5    6    7    8    9   10   11  12  13
DECLARE
*
ERROR at line 1:
ORA-65040: operation not allowed from within a pluggable
database
ORA-06512: at "SYS.PRVT_ADVISOR", line 5825
ORA-06512: at "SYS.PRVT_ADVISOR", line 1659
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 79
ORA-06512: at "SYS.PRVT_ADVISOR", line 6797
ORA-06512: at "SYS.PRVT_ADVISOR", line 6846
ORA-06512: at "SYS.PRVT_ADVISOR", line 1451
ORA-06512: at "SYS.PRVT_ADVISOR", line 5788
ORA-06512: at "SYS.DBMS_ADVISOR", line 103
ORA-06512: at line 4

SQL>
```

The message is explicit. Connect to the CDB root.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> var task_name VARCHAR2(60)
SQL> DECLARE
       taskid NUMBER;
     BEGIN
       dbms_advisor.create_task('ADDM',taskid,:task_name);
       dbms_advisor.set_task_parameter(:task_name,
                         'START_SNAPSHOT', 1);
       dbms_advisor.set_task_parameter(:task_name,
                         'END_SNAPSHOT', 2);
       dbms_advisor.set_task_parameter(:task_name,
                         'DB_ID', 2401763088);
```

```
        dbms_advisor.execute_task(:task_name);
    END;
/
 10   11   12   13  DECLARE
*
ERROR at line 1:
ORA-13703: The snapshot pair [1, 2] for database_id 2401763088
and instance_id [] are not found in the current repository.
ORA-06512: at "SYS.DBMS_ADVISOR", line 200
ORA-06512: at "SYS.PRVT_ADVISOR", line 3306
ORA-06512: at "SYS.PRVT_ADVISOR", line 756
ORA-06512: at "SYS.PRVT_HDM", line 10
ORA-06512: at "SYS.WRI$_ADV_HDM_T", line 39
ORA-06512: at "SYS.PRVT_ADVISOR", line 739
ORA-06512: at "SYS.PRVT_ADVISOR", line 3211
ORA-06512: at "SYS.DBMS_ADVISOR", line 247
ORA-06512: at "SYS.DBMS_ADVISOR", line 195
ORA-06512: at line 11

SQL>
```

Q/ Which DBID did you use?

A/ The DBID is the PDB DBID. Snapshots contain PDB-level and CDB-level AWR data but data is collected altogether for both levels within a single snapshot.

```
SQL> DECLARE
        taskid NUMBER;
    BEGIN
        dbms_advisor.create_task('ADDM',taskid,:task_name);
        dbms_advisor.set_task_parameter(:task_name,
                          'START_SNAPSHOT', 1);
        dbms_advisor.set_task_parameter(:task_name,
                          'END_SNAPSHOT', 2);
        dbms_advisor.set_task_parameter(:task_name,
                          'DB_ID', 1434391901);
        dbms_advisor.execute_task(:task_name);
    END;
/
  2    3    4    5    6    7    8    9   10   11  12  13

PL/SQL procedure successfully completed.

SQL>
```

Q/ Are the recommendations related to PDB level included in the AWR report even if the referring DBID is the CDB root?

A/ Yes. Recommendations at PDB level are reported because PDB AWR snapshots contain statistics related to PDBs also. This has been displayed in step 3.c in practice 6-2. You can retrieve the ADDM report in EM Cloud Control by clicking "Advisors Home" from the Performance menu, and then by clicking ADDM link from the Advisors section. Click the first ADDM task in the list.

Practice 6-3: Monitoring and Tuning SQL Executions at PDB Level

Overview

In this practice, you will monitor and tune SQL statements based on shared application tables and executed in application PDBs. The tables on which the SQL statements rely are shared by application PDBs in the `hr_root` application container. In the previous practice, you monitored resource consumption for the CDB and each PDB. To perform tuning actions in PDBs, configure ports for each PDB.

Tasks

1. Use Enterprise Manager Database Express to tune statements executed in `sales` and `research`.
   a. In a terminal window (*Window sales*), configure the port number for Enterprise Manager Database Express for the `sales` application PDB.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT dbms_xdb_config.gethttpsport FROM DUAL;

GETHTTPSPORT
------------
           0

SQL> EXEC dbms_xdb_config.sethttpsport(5510)

PL/SQL procedure successfully completed.

SQL>
```

   b. Verify that the listener is running and listens to the localhost (*yourserver*) using TCP protocol, the port 5510 for `sales`, the https presentation with RAW session data.

```
SQL> ! lsnrctl status
…
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<your
hostname>)(PORT=5500))(Security=(my_wallet_directory=/u01/app/or
acle/admin/ORCL/xdb_wallet))(Presentation=HTTP)(Session=RAW))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<your
hostname>)(PORT=5510))(Security=(my_wallet_directory=/u01/app/or
acle/admin/ORCL/xdb_wallet))(Presentation=HTTP)(Session=RAW)…
Services Summary...
…
The command completed successfully
$
```

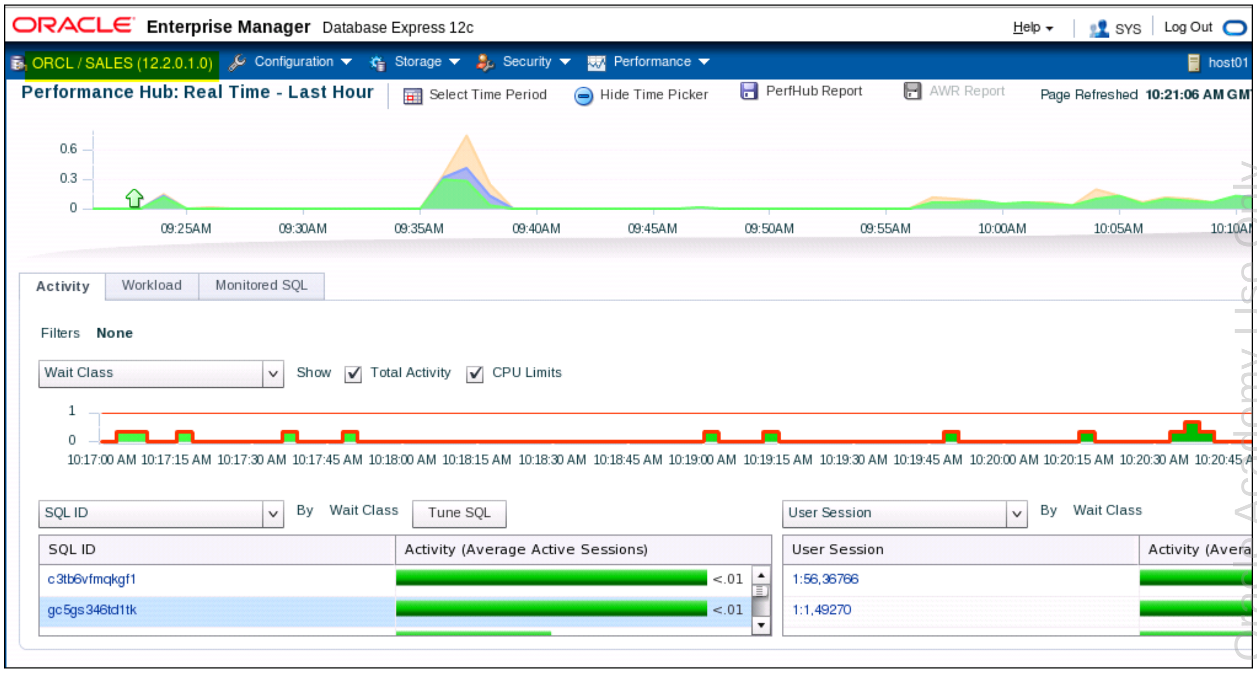
   c. Launch a browser and use the following URL https://localhost:5510/em.
   d. Most probably, you receive a Secure Connection Failed message and you need to add a security exception. At the end of the alert box, click **I Understand the Risks**.

e. At the bottom of the page, click **Add Exception**.

f. Confirm that “Permanently store this exception” is selected in your training environment and click **Confirm Security Exception**.

g. Enter **`sys`** in the User Name field. Enter the password in the Password field. Check the `as sysdba` box. Then click **Login**.

Observe that the `ORACLE_HOME` for the database instance is 12.2.0.1 and the database name is `sales`. To get all statements executed in the container, from the top menu, click “Performance”, and then click the “Performance Hub” option.



SQL ID	Activity (Average Active Sessions)
c3tb6vfmqkgf1	<.01
gc5gs346td1tk	<.01

User Session	Activity (Avera
1:56,36766	
1:1,49270	

Q/ Do you get details on SQL executions in `sales`?

A/ You get the list of the SQL and PL/SQL executed in sessions, but not the execution details.

h. In another terminal window (*Window research*), repeat the same operation for `research` using port 5520.

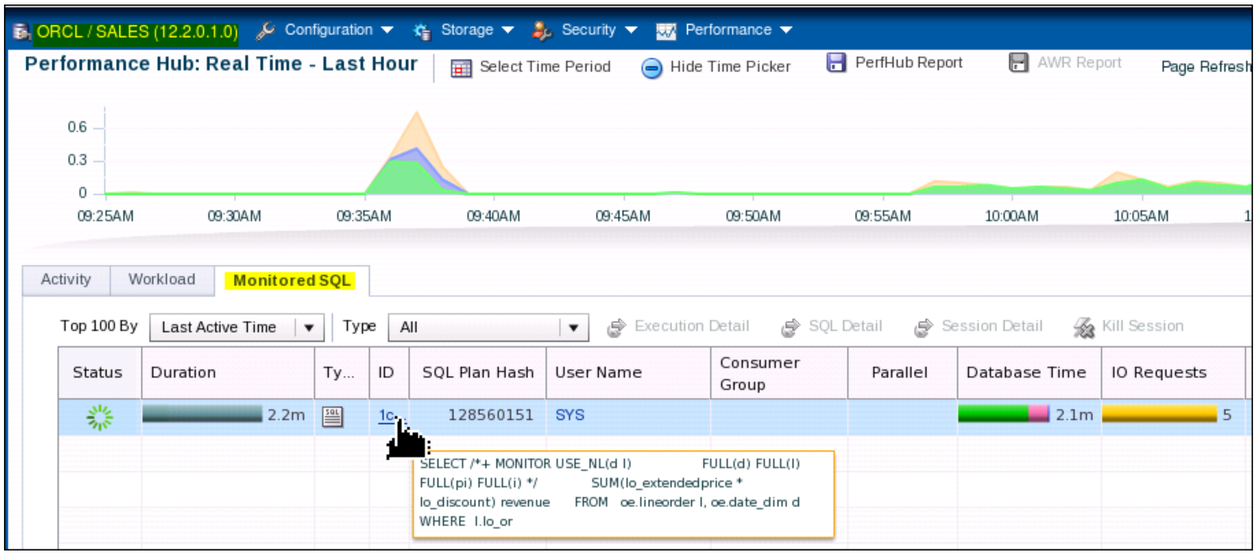
```
$ sqlplus sys@research AS SYSDBA
Enter password: ******
Connected to:

SQL> SELECT dbms_xdb_config.gethttpsport FROM DUAL;

GETHTTPSPORT
------------
           0

SQL> EXEC dbms_xdb_config.sethttpsport(5520)

PL/SQL procedure successfully completed.
```

```
SQL>
```

2. In *Window sales*, execute the following query in the `sales` application PDB. The query is based on common tables shared by `sales` and `research` application PDBs through the `hr_app` application installed in the `hr_root` application container.

```
SQL> SET timing on
SQL> ALTER SESSION SET NLS_DATE_FORMAT='DD-MM-YYYY';

Session altered.

Elapsed: 00:00:00.12
SQL> SELECT /*+ MONITOR USE_NL(d l)
                FULL(d) FULL(l) FULL(pi) FULL(i) */
            SUM(lo_extendedprice * lo_discount) revenue
     FROM   oe.lineorder l, oe.date_dim d
     WHERE  l.lo_orderdate = d.d_datekey
     AND    l.lo_discount BETWEEN 2 AND 3
     AND    l.lo_quantity < 24
     AND    d.d_date='December 24, 1996';
2    3    4    5    6    7

   REVENUE
----------
 939172011

Elapsed: 00:03:15.65
SQL>
```

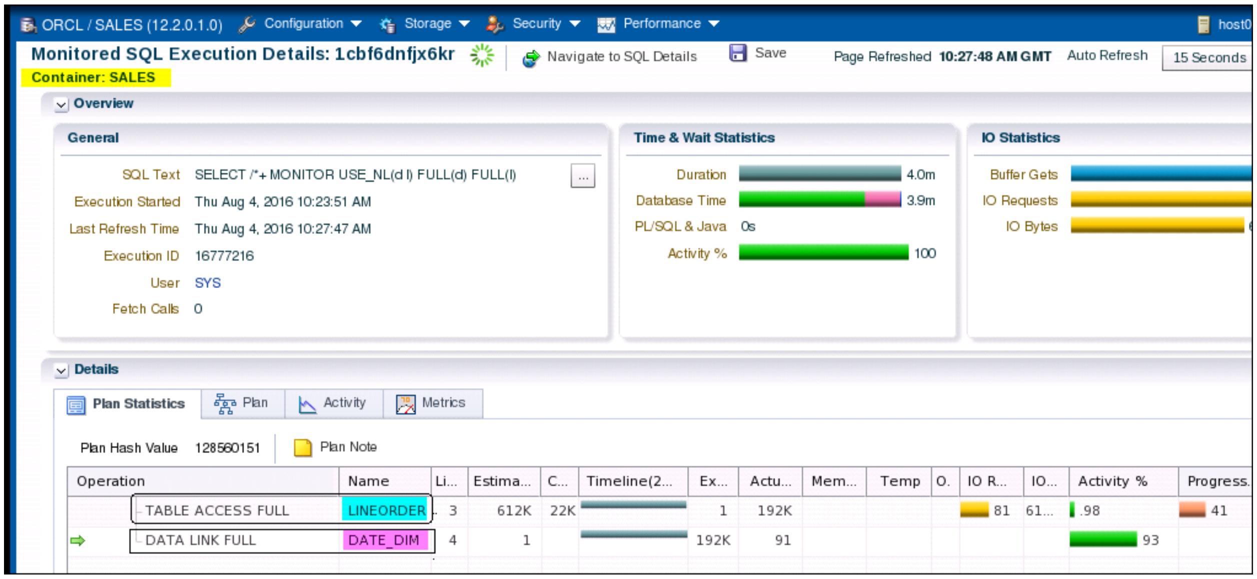
3. Connected to `sales` in EM Database Express, monitor the SQL statement. From the top menu, click “Performance”, then click the “Performance Hub” option, and then click the “Monitored SQL” tab.



Status	Duration	Ty...	ID	SQL Plan Hash	User Name	Consumer Group	Parallel	Database Time	IO Requests
	2.2m		1c..	128560151	SYS			2.1m	5

```
SELECT /*+ MONITOR USE_NL(d l)           FULL(d) FULL(l)
FULL(pi) FULL(i) */          SUM(lo_extendedprice *
lo_discount) revenue      FROM   oe.lineorder l, oe.date_dim d
WHERE  l.lo_or
```

4. Click the ID link of the SQL statement to display the execution plan and statistics.



Page Refreshed 10:27:48 AM GMT

Container: SALES

General

SQL Text	SELECT /*+ MONITOR USE_NL(d l) FULL(d) FULL(l)
Execution Started	Thu Aug 4, 2016 10:23:51 AM
Last Refresh Time	Thu Aug 4, 2016 10:27:47 AM
Execution ID	16777216
User	SYS
Fetch Calls	0

Duration	4.0m
Database Time	3.9m
PL/SQL & Java	0s
Activity %	100

IO Requests
IO Bytes

Plan Hash Value 128560151

Operation	Name	Li...	Estima...	C...	Timeline(2...	Ex...	Actu...	Mem...	Temp	O.	IO R...	IO...	Activity %	Progress.
TABLE ACCESS FULL	LINEORDER	3	612K	22K		1	192K				81	61...	.98	41
DATA LINK FULL	DATE_DIM	4	1			192K	91						93	

Q/ What is the difference between DATA LINK FULL and TABLE ACCESS FULL?

A/ The tables accessed, even if they are all application shared tables, some of them have been declared as data-linked tables during the `hr_app` application installation whose data content can only be declared commonly for all application PDBs. This is the case for the `DATE_DIM` table. The metadata-linked `LINEORDER` table is also an application shared table. Only its definition is common to all application PDBs in the `hr_app` application, and its data is non-shared by application PDBs. This is the reason why in the execution plan, there is a difference between the DATA LINK FULL access to the data-linked `DATE_DIM` table (access to the definition and rows in the application root) and the TABLE ACCESS FULL access to the metadata-linked `LINEORDER` table (access to the definition in the application root and to the rows in the application PDB).

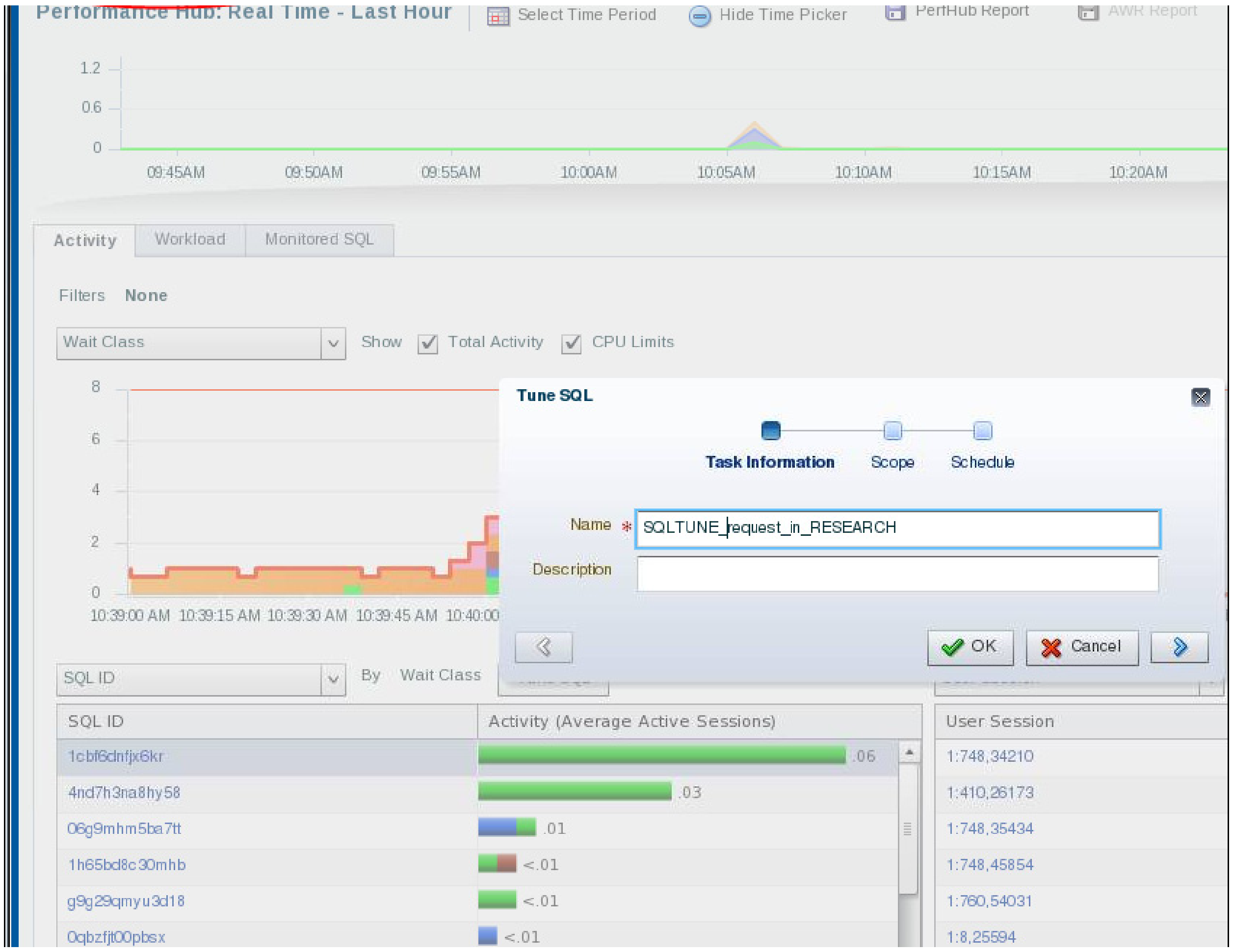
5. In *Window research*, execute the same query in the `research` application PDB. The query is based on the same application tables that are shared by `sales` and `research` application PDBs. However, the `LINEORDER` table has been created as metadata-linked table. Only its structure is common to all application PDBs. Its data is specific to each PDB.

```
SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;

System altered.

SQL> ALTER SYSTEM FLUSH SHARED_POOL;

System altered.

SQL> SET timing on
SQL> ALTER SESSION SET NLS_DATE_FORMAT='DD-MM-YYYY';

Session altered.

Elapsed: 00:00:00.00
SQL> SELECT /*+ MONITOR USE_NL(d l)
                FULL(d) FULL(l) FULL(pi) FULL(i) */
            SUM(lo_extendedprice * lo_discount) revenue
     FROM   oe.lineorder l, oe.date_dim d
     WHERE  l.lo_orderdate = d.d_datekey
     AND    l.lo_discount BETWEEN 2 AND 3
     AND    l.lo_quantity < 24
     AND    d.d_date='December 24, 1996';
2    3    4    5    6    7

   REVENUE
----------
 230845368

Elapsed: 00:01:37.20
SQL>
```

Q/ How can you get tuning recommendations on the recent SQL execution to obtain better performance?

A/ Ask for the SQL Tuning Advisor to run a SQL tuning task.

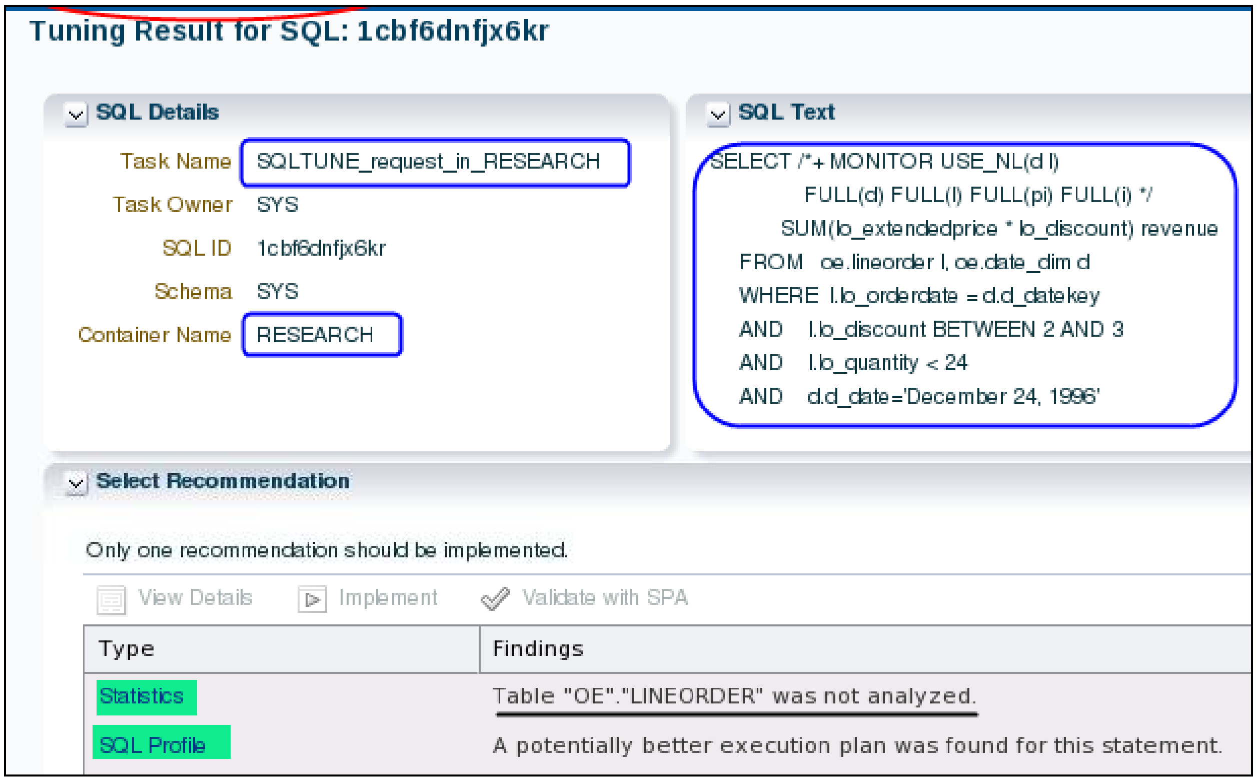
6. Connected to `research` in EM Database Express, from the top menu, click “Performance”, and then click “Performance Hub”. In the “Activity” tab, move the mouse to the line of the SQL execution, right-click and click “Tune SQL”. Enter the `SQLTUNE_request_in_RESEARCH` name for the SQL tuning task. Then click OK.



Name * SQLTUNE_request_in_RESEARCH
Description

SQL ID	Activity (Average Active Sessions)	User Session
1cbf6dnfjx6kr	.06	1:748,34210
4nd7h3na8hy58	.03	1:410,26173
06g9mhm5ba7tt	.01	1:748,35434
1h65bd8c30mhb	<.01	1:748,45854
g9g29qmyu3d18	<.01	1:760,54031
0qbzfjt00pbsx	<.01	1:8,25594

7. Wait until the SQL tuning task is completed. The Status changes to a green check mark when the task is completed. Click the `SQLTUNE_request_in_RESEARCH` link to display and read the SQL tuning recommendations.



Tuning Result for SQL: 1cbf6dnfjx6kr

SQL Details

Task Name	SQLTUNE_request_in_RESEARCH
Task Owner	SYS
SQL ID	1cbf6dnfjx6kr
Schema	SYS
Container Name	RESEARCH

SQL Text

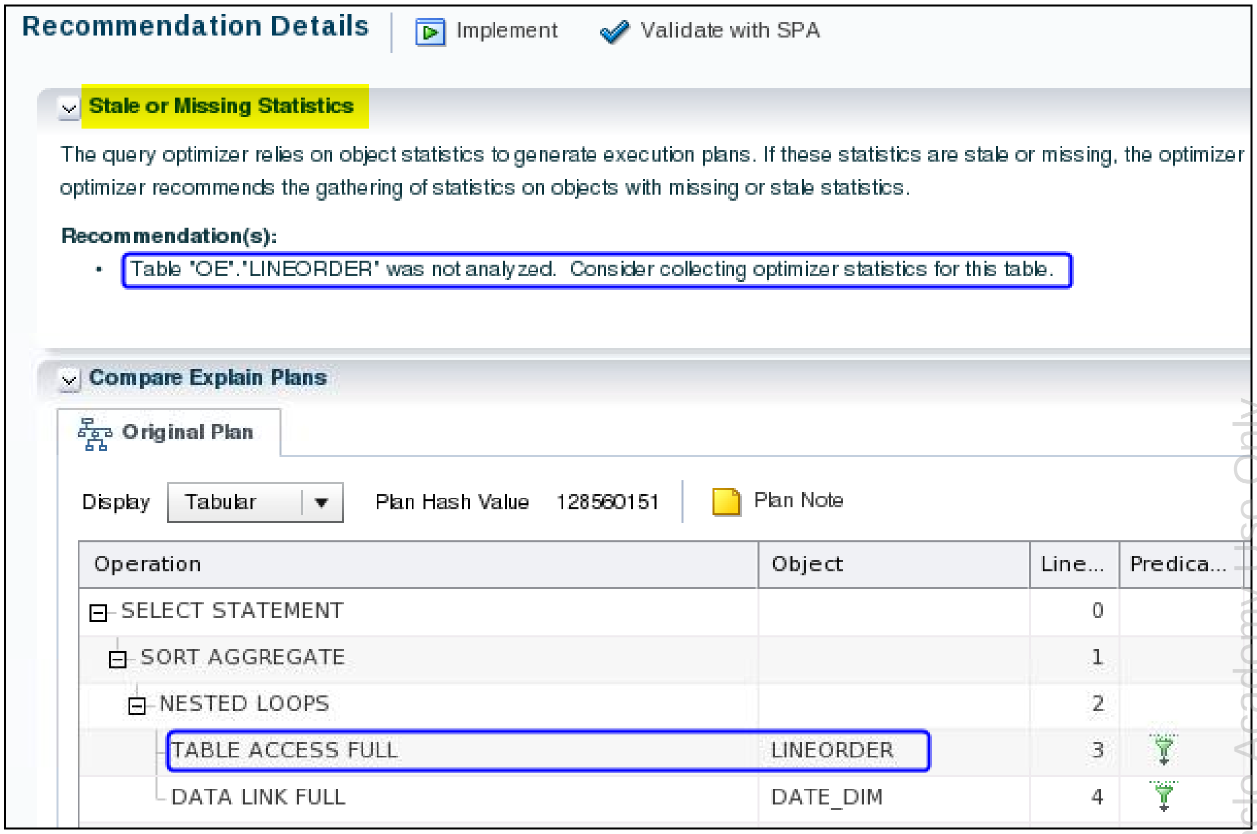
```
SELECT /*+ MONITOR USE_NL(d l)
        FULL(d) FULL(l) FULL(pi) FULL(i) */
      SUM(lo_extendedprice * lo_discount) revenue
  FROM   oe.lineorder l, oe.date_dim d
  WHERE  l.lo_orderdate = d.d_datekey
  AND    l.lo_discount BETWEEN 2 AND 3
  AND    l.lo_quantity < 24
  AND    d.d_date='December 24, 1996'
```

Select Recommendation

Only one recommendation should be implemented.

Type	Findings
Statistics	Table "OE"."LINEORDER" was not analyzed.
SQL Profile	A potentially better execution plan was found for this statement.

a. Click the Statistics link to get details about this first recommendation. Then click Implement to gather the missing statistics. Click OK to start the statistics collection task and then OK when you receive the confirmation message.



Recommendation Details | Implement | Validate with SPA

Stale or Missing Statistics

The query optimizer relies on object statistics to generate execution plans. If these statistics are stale or missing, the optimizer optimizer recommends the gathering of statistics on objects with missing or stale statistics.

Recommendation(s):

- Table "OE"."LINEORDER" was not analyzed. Consider collecting optimizer statistics for this table.

Compare Explain Plans

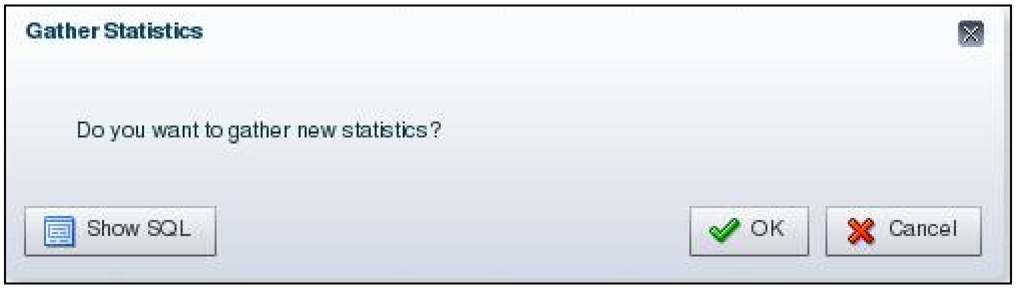
Original Plan

Display: Tabular | Plan Hash Value 128560151 | Plan Note

Operation	Object	Line...	Predica...
SELECT STATEMENT		0	
SORT AGGREGATE		1	
NESTED LOOPS		2	
TABLE ACCESS FULL	LINEORDER	3	
DATA LINK FULL	DATE_DIM	4	

Q/ Why are recommendations about stale statistics on the `DATE_DIM` table not reported?

A/ The `DATE_DIM` table being a data-linked table needs statistics collected in the application root, whereas metadata-linked tables need statistics collected in the application PDB. This difference comes from the data location. Data in data-linked tables is stored in the application root and data in metadata-linked tables is stored in each respective application PDB.



Gather Statistics

Do you want to gather new statistics?

Show SQL | OK | Cancel

This will perform the following PL/SQL execution in the `research` application PDB:

```
EXEC dbms_stats.gather_table_stats(ownname => 'OE', tabname =>
'LINEORDER', estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
method_opt => 'FOR ALL COLUMNS SIZE AUTO')
```

b. Verify that the statistics are collected for the tables in `research`.

```
SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT table_name, con_id, num_rows, blocks, avg_row_len
     FROM   cdb_tables WHERE table_name = 'LINEORDER';
  2
TABLE_NAME            CON_ID   NUM_ROWS     BLOCKS AVG_ROW_LEN
--------------------- ------ ---------- ---------- -----------
LINEORDER                  5
LINEORDER                  4
LINEORDER                  3
LINEORDER                  6    2804860      40569          98

SQL>
```

Q1/ Are the statistics on the application shared tables collected in `sales`?

A1/ No, they aren't. It is the normal expectation because these tables are metadata-linked tables. The statistics were collected in `research` and not in `sales`.

Q2/ What should you do to have statistics across the application container?

A2/ Statistics for the same shared tables should be gathered in the `sales` application PDB.

Q3/ What about the collection of statistics for the data-linked `DATE_DIM` table?

A3/ Statistics for the shared data-linked tables can be gathered in the `hr_root` application root.

c. Go back to "Tuning Result for SQL: 1cbf6dnfjx6kr". Then click "SQL Profile". You can now view from the "Original Plan" tab the original plan used that conducted to a low performance execution. Click the "Plan Using SQL Profile" tab to see what could be the execution plan if you applied the suggested SQL profile to the SQL statement.

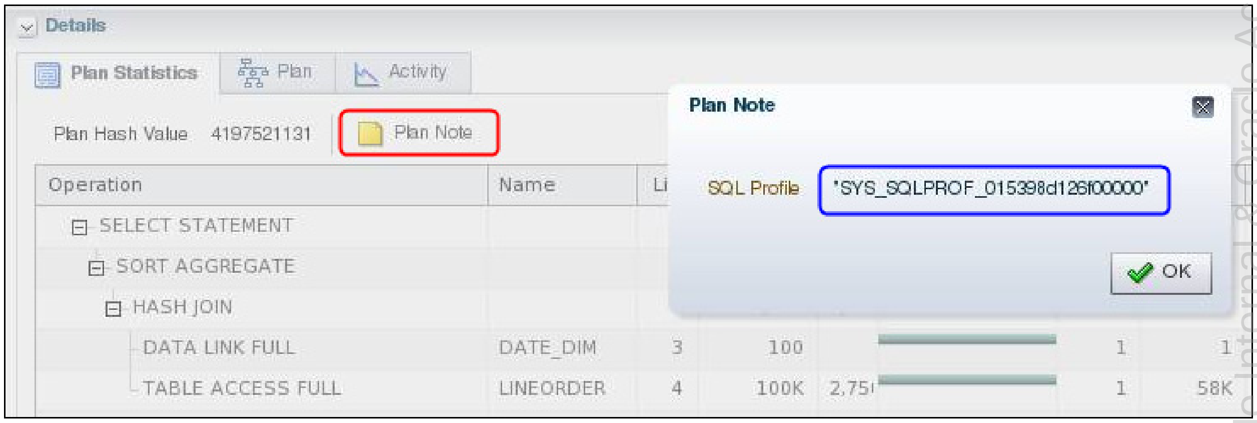
d. Click Implement, then click OK to start the SQL Profile application task, and then click OK when you receive the confirmation message.

8. Reexecute the same query after having emptied the buffer cache and shared pool. The query should execute faster.

```
SQL> CONNECT oe@research
Enter password: ******
Connected.
SQL> ALTER SYSTEM FLUSH buffer_cache;

System altered.

SQL> ALTER SYSTEM FLUSH shared_pool;

System altered.
```

```
SQL> SELECT /*+ MONITOR USE_NL(d l)
                FULL(d) FULL(l) FULL(pi) FULL(i) */
            SUM(lo_extendedprice * lo_discount) revenue
     FROM   oe.lineorder l, oe.date_dim d
     WHERE  l.lo_orderdate = d.d_datekey
     AND    l.lo_discount BETWEEN 2 AND 3
     AND    l.lo_quantity < 24
     AND    d.d_date='December 24, 1996';
2    3    4    5    6    7

   REVENUE
----------
 230845368

Elapsed: 00:00:01.33
SQL>
```

a. Verify that the monitored execution used the SQL profile applied. In EM Express, from the top menu, click "Performance", then click "Performance Hub", and finally click the "Monitored SQL" tab. Click the "ID" link of the monitored SQL. Then click "Plan Note".



Plan Hash Value 4197521131

Operation	Name	Li				
SELECT STATEMENT						
SORT AGGREGATE						
HASH JOIN						
DATA LINK FULL	DATE_DIM	3	100		1	1
TABLE ACCESS FULL	LINEORDER	4	100K	2,75	1	58K

Plan Note

SQL Profile 'SYS_SQLPROF_015398d126f00000'

Q1/ Is the execution time different from the first execution?

A1/ Yes. It is much faster.

Q2/ How would the same query perform in the other application PDB of the same application container?

A2/ The statistics of the shared tables of the common `oe` schema in the `hr_root` application container were gathered for the metadata-linked tables within one application PDB only, `research`. Because statistics were not collected on tables in `sales`, the performance will still be as bad as it was.

9. After emptying the buffer cache and the shared pool, execute the same query in `sales`.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SYSTEM FLUSH buffer_cache;

System altered.

SQL> ALTER SYSTEM FLUSH shared_pool;

System altered.

SQL> SET timing on
SQL> SELECT /*+ MONITOR USE_NL(d l)
                FULL(d) FULL(l) FULL(pi) FULL(i) */
            SUM(lo_extendedprice * lo_discount) revenue
     FROM   oe.lineorder l, oe.date_dim d
     WHERE  l.lo_orderdate = d.d_datekey
     AND    l.lo_discount BETWEEN 2 AND 3
     AND    l.lo_quantity < 24
     AND    d.d_date='December 24, 1996';
2    3    4    5    6    7

   REVENUE
----------
 939172011
```

```
Elapsed: 00:03:15.65

SQL>
```

Q/ What do you notice?

A/ The performance is still not good.

10. Collect the statistics on the same shared tables as it was done in `research`.

```
SQL> EXEC dbms_stats.gather_table_stats(ownname => 'OE', tabname
=> 'LINEORDER', estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
method_opt => 'FOR ALL COLUMNS SIZE AUTO')

PL/SQL procedure successfully completed.

SQL>
```

11. Verify that the statistics are collected for the tables in `sales`.

```
SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT table_name, con_id, num_rows, blocks, avg_row_len
     FROM   cdb_tables WHERE table_name = 'LINEORDER';
  2
TABLE_NAME               CON_ID   NUM_ROWS     BLOCKS AVG_ROW_LEN
---------------------- ------ ---------- ---------- -----------
LINEORDER                     5    5609720      80529          98
LINEORDER                     6    2804860      40569          98
LINEORDER                     3
LINEORDER                     4

SQL>
```

Q/ Why are the statistics in the two application PDBs different for the same shared table?
A/ The table is a metadata-linked table and therefore only the definition of the table is common to the application PDBs. Each PDB stores the rows for the shared table in its own tablespace. This is the reason why the number of rows is different in each PDB.

12. After emptying the buffer cache and the shared pool, reexecute the same query in `sales`.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SYSTEM FLUSH buffer_cache;

System altered.

SQL> ALTER SYSTEM FLUSH shared_pool;

System altered.
```

```
SQL> SELECT /*+ MONITOR USE_NL(d l)
                FULL(d) FULL(l) FULL(pi) FULL(i) */
            SUM(lo_extendedprice * lo_discount) revenue
     FROM   oe.lineorder l, oe.date_dim d
     WHERE  l.lo_orderdate = d.d_datekey
     AND    l.lo_discount BETWEEN 2 AND 3
     AND    l.lo_quantity < 24
     AND    d.d_date='December 24, 1996';
2    3    4    5    6    7
```

```
   REVENUE
----------
 939172011

Elapsed: 00:01:26.25
SQL>
```

Q/ Does the SQL execution use the same SQL Profile as used for the query execution in `research`?

```
SQL> CONNECT sys@research AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT name, sql_text FROM dba_sql_profiles;

NAME
------------------------------
SQL_TEXT
----------------------------------------------------------------
SYS_SQLPROF_015398d126f00000
SELECT /*+ MONITOR USE_NL(d l)
                FULL(d) FULL(l) FULL(pi) FULL(i)

SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT name FROM dba_sql_profiles;

no rows selected

SQL>
```

A/ No, it does not use any profile. The SQL Profile created for the execution of the query in `research` is stored in `research` and linked to the SQL ID in `research` whereas no SQL profile was created in `sales`. If any had been created, it would be linked to the SQL ID in `sales` with another profile name.

13. Complete statistics collection on shared data-linked tables.
    a. Check that there are no statistics yet on the `date_dim` shared table.

```
SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT owner, object_name, sharing,
              application Application
     FROM   dba_objects
     WHERE  object_name = 'DATE_DIM';
  2    3    4
OWNER  OBJECT_NAME          SHARING       APPLICATION
------ -------------------- ------------- ------------
OE     DATE_DIM             DATA LINK     Y

SQL> SELECT table_name, con_id, num_rows, blocks, avg_row_len
     FROM   cdb_tables WHERE table_name = 'DATE_DIM';

TABLE_NAME              CON_ID   NUM_ROWS     BLOCKS AVG_ROW_LEN
----------------------- ------ ---------- ---------- -----------
DATE_DIM                     6
DATE_DIM                     3
DATE_DIM                     4
DATE_DIM                     5

SQL>
```

b. Collect statistics for the data-linked table.

```
SQL> EXEC dbms_stats.gather_table_stats(ownname => 'OE', tabname
=> 'DATE_DIM', estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
method_opt => 'FOR ALL COLUMNS SIZE AUTO')

PL/SQL procedure successfully completed.

SQL> SELECT table_name, con_id, num_rows, blocks, avg_row_len
     FROM   cdb_tables
     WHERE table_name = 'DATE_DIM';

TABLE_NAME              CON_ID   NUM_ROWS     BLOCKS AVG_ROW_LEN
----------------------- ------ ---------- ---------- -----------
DATE_DIM                     6
DATE_DIM                     5
DATE_DIM                     4
DATE_DIM                     3       2556         43         100

SQL>
```

Q/ Which statistics values would you expect if you collected statistics on a data-linked table while connected to an application PDB?

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
```

```
SQL> EXEC dbms_stats.gather_table_stats(ownname => 'OE', tabname
=> 'DATE_DIM', estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE,
method_opt => 'FOR ALL COLUMNS SIZE AUTO')


PL/SQL procedure successfully completed.

SQL> SELECT table_name, con_id, num_rows, blocks, avg_row_len
     FROM   cdb_tables WHERE table_name = 'DATE_DIM';
  2
TABLE_NAME             CON_ID   NUM_ROWS     BLOCKS AVG_ROW_LEN
---------------------- ------ ---------- ---------- -----------
DATE_DIM                    5          0          0           0

SQL> EXIT
$
```

A/ Values would be set to 0 because the rows of the segment are stored in the application root tablespace.

14. Because EM Cloud Control will not be used for some time, release `cdbem` resources.

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdbem
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

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

Practice 6-4 Using Performance Profiles to Limit CPU Between Application PDBs

Overview

In this practice, you create two Resource Manager performance profiles and associated directives to limit CPU resources used by application PDBs in an application container.
In the first test, you want the `sales` application PDB to be allocated more CPU resources than the `research` application PDB.

In the second test, you want the `pdb_orcl` PDB to be allocated more CPU resources than both application PDBs within the `hr_root` application container, which get equal resources within the application container.

Tasks

1. Create the `pdb_orcl` regular PDB.

```
$ $HOME/labs/APP/setup_pdb_orcl.sh
…
$
```

2. Open a terminal window (it will be referred to as *Window sales*) to connect to `sales` in `ORCL` and create a PL/SQL procedure that burns CPU in `sales` as the `SYSTEM` user. You can use the `create_burn_cpu.sql` script to create the procedure after connecting to `sales`.

```
$ sqlplus system@sales
Enter password: ******
Connected.
SQL> @$HOME/labs/APP/create_burn_cpu.sql

Procedure created.

SQL>
```

3. Open a second terminal window (it will be referred to as *Window research*) to connect to `research` in `ORCL` and create a PL/SQL procedure that burns CPU in `research` as the `SYSTEM` user. You can use the `create_burn_cpu.sql` script to create the procedure after connecting to `research`.

```
$ sqlplus system@research
Enter password: ******
Connected.
SQL> @$HOME/labs/APP/create_burn_cpu.sql

Procedure created.

SQL>
```

4. From *Window sales*, execute the `setup_prof_directives.sql` script to create a new CDB plan called `HR_plan` and two performance profiles.
   `prof_high` in the `HR_plan` plan gives four shares and 80% CPU limit to the PDB being assigned the performance profile directive and `prof_low` in the `HR_plan` plan gives two shares and 50% CPU limit to the PDB being assigned the performance profile.

```
SQL> @$HOME/labs/APP/setup_prof_directives.sql
…
SQL>
```

5. Still from *Window sales*, make sure both performance profiles in the `Toys_plan` plan and associated directives were created correctly. Execute the `display_prof_directives.sql` script.

```
SQL> @$HOME/labs/APP/display_prof_directives.sql

PLAN
----------
HR_PLAN



PLAN       PROFILE        SHARES  CPU limit
---------- ---------- ---------- ----------
HR_PLAN    PROF_HIGH           4         80
HR_PLAN    PROF_LOW            2         50
HR_PLAN                                  90
HR_PLAN                        1        100

SQL>
```

6. Still from *Window sales*, activate the `HR_plan` CDB plan.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET resource_manager_plan = 'HR_PLAN';

System altered.

SQL> SELECT name FROM v$rsrc_plan where con_id = 1;

NAME
------------------------------
HR_PLAN

SQL>
```

Q/ What else do you have to set to achieve your first goal?

A/ Set the appropriate performance profile to each application PDB.

7. ***First test:***

a. Still from *Window sales*, connect as the `SYS` user in `sales`.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL>
```

c. Set the appropriate performance profile.

```
SQL> ALTER SYSTEM SET db_performance_profile='PROF_HIGH'
                     SCOPE = spfile;
   2
System altered.

SQL>
```

d. Restart the PDB.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> SHOW PARAMETER db_performance_profile

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------
db_performance_profile              string      PROF_HIGH
SQL>
```

e. Connect as `SYSTEM` and set `SERVEROUPUT` variable to `ON`.

```
SQL> CONNECT system@sales
Enter password: ******
Connected.
SQL> SET serveroutput on
SQL>
```

f. From *Window research*, connect as the `SYS` user in `research`.

```
SQL> CONNECT sys@research AS SYSDBA
Enter password: ******
Connected.
SQL>
```

g. Set the appropriate performance profile.

```
SQL> ALTER SYSTEM SET db_performance_profile='PROF_LOW'
                      SCOPE = spfile;
  2
System altered.

SQL>
```

h. Restart the PDB.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> SHOW PARAMETER db_performance_profile

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------
db_performance_profile               string      PROF_LOW
SQL>
```

i. Connect as `SYSTEM` and set `SERVEROUPUT` variable to `ON`.

```
SQL> CONNECT system@research
Enter password: ******
Connected.
SQL> SET serveroutput on
SQL>
```

j. **DO NOT WAIT AND GO TO THE NEXT STEP IMMEDIATELY:** From *Window sales*, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU:    76.4 Wall:  134.8 k: 2000000000

PL/SQL procedure successfully completed.

SQL>
```

k. From *Window research*, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
```

```
CPU:    76.4 Wall:  207.2 k: 2000000000

PL/SQL procedure successfully completed.

```

```
SQL> EXIT
$
```

Q/ What do you observe?
A/ The procedure in `sales` finishes its execution before `research` and has consumed less CPU and wall-clock time than `research`.
This is expected because `sales` is receiving four shares of CPU whereas `research` is receiving only two shares.

8. ***Second test:***
   a. From *Window sales*, connect as user `SYS` in `hr_root` application root, and set the performance profile to `PROF_LOW` after resetting both application PDBs' performance profile to NULL.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: ******
Connected.
SQL>
SQL> ALTER SYSTEM SET db_performance_profile='' SCOPE = spfile;

System altered.

SQL> CONNECT sys@research AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SYSTEM SET db_performance_profile='' SCOPE = spfile;

System altered.

SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: ******
Connected.
SQL> ALTER SYSTEM SET db_performance_profile='PROF_LOW'
                         SCOPE = spfile;
  2
System altered.

SQL>
```

   b. Restart the application root and application PDBs.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.
```

```
SQL> ALTER PLUGGABLE DATABASE all OPEN;

Pluggable database altered.

SQL> SHOW PARAMETER db_performance_profile

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------
db_performance_profile               string       PROF_LOW
SQL>
```

c. Connect as SYSTEM and set SERVEROUPUT variable to ON.

```
SQL> CONNECT system@sales
Enter password: ******
Connected.
SQL> SET serveroutput on
SQL>
```

d. From *Window research*, connect as SYSTEM and set SERVEROUPUT variable to ON. Be ready to execute the CPU burner procedure.

```
SQL> CONNECT system@research
Enter password: ******
Connected.
SQL> SET serveroutput on
SQL>
```

e. Open a third terminal window (it will be referred to as W*indow pdb_orcl*) to connect to pdb_orcl in ORCL and create a PL/SQL procedure that burns CPU as the SYSTEM user.

```
$ sqlplus system@pdb_orcl
Enter password: ******
Connected.
SQL> @$HOME/labs/APP/create_burn_cpu.sql

Procedure created.

SQL>
```

f. Connect to pdb_orcl as user SYS and set the performance profile to PROF_HIGH.

```
SQL> CONNECT sys@pdb_orcl AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET db_performance_profile='PROF_HIGH'
                  SCOPE = spfile;
  2
System altered.
```

```

SQL>
```

g. Restart the PDB.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> SHOW PARAMETER db_performance_profile

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------
db_performance_profile               string      PROF_HIGH
SQL>
```

h. Connect as `SYSTEM` and set `SERVEROUPUT` variable to `ON`. **DO NOT WAIT AND GO TO THE NEXT STEP IMMEDIATELY:** From *Window sales* and *Window research*, execute the CPU burner procedure.

```
SQL> CONNECT system@pdb_orcl
Enter password: ******
Connected.
SQL> SET serveroutput on
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU:    76.4 Wall:  131.7 k: 2000000000

PL/SQL procedure successfully completed.

SQL>
```

i. From *Window sales*, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU:    76.4 Wall:  259.5 k: 2000000000

PL/SQL procedure successfully completed.

SQL>
```

j. From *Window research*, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU:    76.4 Wall:  259.8 k: 2000000000

PL/SQL procedure successfully completed.
```

```
SQL>
```

Q/ What do you observe?

A/ The procedure in `pdb_orcl` finishes its execution much earlier than both application PDBs and has consumed much less wall-clock time. This is expected because `pdb_orcl` is granted four shares of CPU.

The procedure in both application PDBs (`sales` and `research`) in `hr_root` application container finish their execution at the same time, and have consumed the same wall-clock time.
This is expected because the Resource Manager plan grants the two shares of CPU allocated to the application equally to `sales` and `research`.

9. From any *window*, connect as user `SYS` in the CDB root, and change the CDB Resource Manager plan back to its default.

```
SQL> CONNECT / AS SYSDBA
Connected.

SQL> ALTER SYSTEM SET resource_manager_plan = '';

System altered.

SQL> SELECT name FROM v$rsrc_plan where con_id = 1;

NAME
--------------------------------
ORA$INTERNAL_CDB_PLAN

SQL> EXIT
$
```

Note: You can close the other two terminal windows.

Practices for Lesson 7: Upgrade and Other Operations in CDBs and PDBs

Chapter 7

Practices for Lesson 7: Overview

Practices Overview

During upgrade operations, you will:

- Upgrade the Oracle Database 12.1.0.2 regular PDB `pdb1` to the Oracle Database 12.2.0.1 `cdb2` as a new application PDB of the `hr_root` application container
- Optionally upgrade the Oracle Database 12.1.0.2 `cdb1` to the Oracle Database 12.2.0.1 environment

During all the practices, it is recommended that you keep two windows open to avoid any inappropriate operations due to a wrong environment setup:

  – One with `ORACLE_HOME` set to 12.1.0.2 and `ORACLE_SID` set to `cdb1` (named *session 12.1*)
  – Another one with `ORACLE_HOME` set to 12.2.0.1 and `ORACLE_SID` set to `cdb2` (named *session 12.2*)

Practice 7-1: Upgrading a 12.1.0.2 Regular PDB to a 12.2.0.1 Application PDB

Overview

You will upgrade the Oracle Database 12.1.0.2 `pdb1` from `cdb1` to Oracle Database 12.2.0.1 `pdb1` into `cdb2` as an application PDB of the `hr_root` application container.

Tasks

1. Open two terminal windows, one with Oracle Database 12.1.0.2 environment variables set and another with Oracle Database 12.2.0.1 environment variables set.
   a. Before starting the practice, execute the `$HOME/labs/admin/glogin_7.sh` shell script. The script sets formatting for all columns selected in queries in both Oracle Database 12.1.0.2 and Oracle Database 12.2.0.1 environments.

```
$ $HOME/labs/admin/glogin_7.sh
$
```

   b. Open a terminal window and set title to 12.1.0.2. *(Terminal -> Set Title…).* This will be *session 12.1.* If `cdb1` is not started, start it up.

```
$ . oraenv
ORACLE_SID = [cdb1] ? cdb1
The Oracle base remains unchanged with value /u01/app/oracle
$ env|grep ORACLE
ORACLE_SID=cdb1
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
$ sqlplus / AS SYSDBA

Connected to an idle instance.

SQL> STARTUP
ORACLE instance started.

Total System Global Area  666894336 bytes
Fixed Size                  2927960 bytes
Variable Size             373293736 bytes
Database Buffers          285212672 bytes
Redo Buffers                5459968 bytes
Database mounted.
Database opened.
SQL> EXIT
$
```

c. Execute the `$HOME/labs/APP/setup_pdb1.sh` shell script to create in `pdb1` PDB the *`test.bigtab`* table. While the script is executing, go to step d).

```
$ $HOME/labs/APP/setup_pdb1.sh
…
$
```

d. Open another terminal window and set title to 12.2.0.1. *(Terminal -> Set Title…).* This will be *session 12.2.* The `$HOME/labs/APP/setup2_hr_app.sh` shell script creates the `hr_root` application container and its `operations` application PDB. If the instance is not started, restart it.
Ensure that `ORCL` is now shut down. If you keep it open, you may encounter service confusion.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

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

```
$ . oraenv
ORACLE_SID = [ORCL] ? cdb2
The Oracle base remains unchanged with value /u01/app/oracle
$ env|grep ORACLE
ORACLE_SID=cdb2
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/12.2.0/dbhome_1
$ $HOME/labs/APP/setup2_hr_app.sh
…
$
```

2. In *session 12.1*, prepare `pdb1` to be unplugged from `cdb1` and upgraded in `cdb2`.

a. Execute the Pre-Upgrade Information Tool on the 12.1.0.2 `pdb1` by executing the `preupgrade.jar` file.

```
$ cd /u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin
$ $ORACLE_HOME/jdk/bin/java -jar preupgrade.jar -c 'PDB1'
Preupgrade generated files:
/u01/app/oracle/cfgtoollogs/cdb1/preupgrade/preupgrade.log
/u01/app/oracle/cfgtoollogs/cdb1/preupgrade/preupgrade_fixups.sq
l
```

```
/u01/app/oracle/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups.s
ql
$
```

Q1/ Did the Pre-Upgrade Information Tool work successfully?

A1/ The default output is a directory. This directory is by default ***`$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade`.***

Reexecute the same command to get the report on your terminal.

```
$ cd /u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin
$ $ORACLE_HOME/jdk/bin/java -jar preupgrade.jar -c 'PDB1'
TERMINAL TEXT
Report generated by Oracle Database Pre-Upgrade Information Tool
Version 12.2.0.1.0

Upgrade-To version: 12.2.0.1.0

=======================================
Status of the database prior to upgrade
=======================================

      Database Name:  CDB1
     Container Name:  PDB1
       Container ID:  3
            Version:  12.1.0.2.0
         Compatible:  12.1.0.2.0
          Blocksize:  8192
           Platform:  Linux x86 64-bit
      Timezone File:  18
  Database log mode:  ARCHIVELOG
           Readonly:  FALSE
            Edition:  EE

  Oracle Component              Upgrade Action    Current Status
  ----------------              --------------    -----------
  Oracle Server                  [to be upgraded]  VALID
  JServer JAVA Virtual Machine   [to be upgraded]  VALID
  Oracle XDK for Java            [to be upgraded]  VALID
  Real Application Clusters      [to be upgraded]  OPTION OFF
  Oracle Workspace Manager       [to be upgraded]  VALID
  OLAP Analytic Workspace        [to be upgraded]  VALID
  Oracle Label Security          [to be upgraded]  VALID
  Oracle Database Vault          [to be upgraded]  VALID
  Oracle Text                    [to be upgraded]  VALID
  Oracle XML Database            [to be upgraded]  VALID
  Oracle Java Packages           [to be upgraded]  VALID
  Oracle Multimedia              [to be upgraded]  VALID
  Oracle Spatial                 [to be upgraded]  VALID
  Oracle Application Express     [to be upgraded]  VALID
  Oracle OLAP API                [to be upgraded]  VALID

==============
BEFORE UPGRADE
==============

  Run <preupgradeLogDirPath>/preupgrade_fixups_PDB1.sql to
complete all of the BEFORE UPGRADE action items below marked
with '(AUTOFIXUP)'.

  REQUIRED ACTIONS
  ================
   + Adjust TABLESPACE SIZES as needed.
                            Auto    12.2.0.1.0
     Tablespace         Size  Extend  Min Size    Action
     ----------       ------  ------- ----------  -----
     SYSAUX            570 MB ENABLED   1484 MB   None
     SYSTEM            260 MB ENABLED    761 MB   None
     TEMP               20 MB ENABLED    150 MB   None

     Note that 12.2.0.1.0 minimum sizes are estimates.
     If you plan to upgrade multiple pluggable databases
concurrently, then you must ensure that the UNDO tablespace size
is equal to at least the number of pluggable databases that you
upgrade concurrently, multiplied by that minimum.  Failing to
allocate sufficient space can cause the upgrade to fail.

  RECOMMENDED ACTIONS
  ===================
   + (AUTOFIXUP) Gather SYS schema and stale data dictionary
statistics prior to database upgrade in off-peak time using:

      EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;
      EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS('SYS');

     Dictionary statistics do not exist or are stale (not up-to-
date).
```

```
     Dictionary statistics help the Oracle optimizer find
efficient SQL execution plans and are essential for proper
upgrade timing. Oracle recommends gathering dictionary
statistics in the last 24 hours before database upgrade.

  INFORMATION ONLY
  ================
   + Consider upgrading APEX manually, before the database
upgrade.

     The database contains APEX version 4.2.5.00.08 and will
need to be upgraded to at least version 5.0.4.00.12.

     To reduce database upgrade time, you can upgrade APEX
manually before the database upgrade.  Refer to My Oracle
Support Note 1088970.1 for information on APEX installation
upgrades.


=============
AFTER UPGRADE
=============


  Run <preupgradeLogDirPath>/postupgrade_fixups_PDB1.sql to
complete all of the AFTER UPGRADE action items below marked with
'(AUTOFIXUP)'.

  REQUIRED ACTIONS
  ================
  None

  RECOMMENDED ACTIONS
  ===================
   + Upgrade the database time zone version using the DBMS_DST
package.

     The database is using timezone datafile version 18 and the
target 12.2.0.1.0 database ships with timezone datafile version
26.

     Oracle recommends using the most recent timezone data.  For
further information, refer to My Oracle Support Note 1585343.1.

   + (AUTOFIXUP) Gather dictionary statistics after the upgrade
using the command:
        EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

     Oracle recommends gathering dictionary statistics after
upgrade.

     Dictionary statistics provide essential information to the
Oracle optimizer to help it find efficient SQL execution plans.
After a database upgrade, statistics need to be re-gathered as
there can now be tables that have significantly changed during
the upgrade or new tables that do not have statistics gathered
yet.

Preupgrade generated files:

/u01/app/oracle/cfgtoollogs/cdb1/preupgrade/preupgrade_fixups.sq
l
/u01/app/oracle/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups.s
ql
$
```

Q2/ Which containers in the CDB does the Pre-Upgrade Information Tool provide actions for?

A2/ The Pre-Upgrade Information Tool provides actions for `PDB1` only because the PDB was explicitly defined in the inclusion list.

Q3/ Does the Pre-Upgrade Information Tool provide only recommendations?

A3/ The Pre-Upgrade Information Tool provides required and recommended actions, and useful information.

Q4/ Does the Pre-Upgrade Information Tool provide the required and recommended actions to be executed before the upgrade only?

A4/ The Pre-Upgrade Information Tool provides the required and recommended actions to be executed BEFORE and some of them to be executed AFTER the upgrade. This is the reason why you will find `preupgrade_fixups` and `postupgrade_fixups` SQL scripts.

Q5/ What is the structure of the report generated by Pre-Upgrade Information Tool?

A5/ The structure of the report is as follows:

- Status of the database prior to upgrade
- BEFORE UPGRADE
  - REQUIRED ACTIONS
  - RECOMMENDED ACTIONS
  - INFORMATION ONLY
- AFTER UPGRADE
  - REQUIRED ACTIONS
  - RECOMMENDED ACTIONS

b. Read the required actions and recommendations of the Pre-Upgrade Information Tool reported for `pdb1`.

```
$ cd /u01/app/oracle/cfgtoollogs/cdb1/preupgrade/
$ ls *PDB1.sql
postupgrade_fixups_PDB1.sql  preupgrade_fixups_PDB1.sql
$
```

Q1/ Is there a script created to fix the recommendations?

A1/ Yes. There is the `preupgrade_fixups_PDB1.sql` SQL script.

Note: Read the script before executing it.

```
$ sqlplus / AS SYSDBA
Connected.
SQL> ALTER SESSION SET container = pdb1;

Session altered.

SQL>
@$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade/preupgrade_fixups_PDB1
.sql
Executing Oracle PRE-Upgrade Fixup Script

Auto-Generated by:       Oracle Preupgrade Script
                         Version: 12.2.0.1.0 Build: 1
Generated on:            2016-03-30 21:28:30

Source Database:         CDB1
Source Database Version: 12.1.0.2.0
For Upgrade to Version:  12.2.0.1.0
Executing in container:  PDB1

                          Fixup
Check Name                Status  Further DBA Action
----------                ------  ------------------
dictionary_stats          Passed  None
apex_upgrade_msg          Failed  Manual fixup recommended.

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

Q2/ Did the script fix all required and recommended actions?

A2/ No, it did not. APEX is a manual operation to be completed if the 12.2 CDB does not hold APEX at the CDB root level. There are several solutions.

- In the target 12.2 CDB, you could install APEX common, prior to plugging in the PDB. APEX in the newly plugged-in PDB would have to be upgraded to the version installed common.
- Alternatively, and more cumbersome, use the command-line utility APEXExport to export all workspaces, applications, scripts, users, artifacts, then plug in the PDB, remove APEX from the PDB, install APEX locally into the PDB and finally import all of the artifacts which were exported.

c. Check if the target CDB holds APEX as a common component. If this is the case, you do not have to install APEX common in `cdb2`.
In *session 12.2*:

```
$ sqlplus / as sysdba
Connected.
SQL> SELECT comp_name, version FROM dba_registry
     WHERE  comp_name = 'Oracle Application Express';

  2
no rows selected


SQL> EXIT
$
```

If APEX common is not installed, you install APEX common in `cdb2` as follows and meanwhile back up `cdb1` and unplug `pdb1` (see next two steps). Ensure that `pdb2` is started in `cdb2`.

```
$ cd $ORACLE_HOME/apex/
$ sqlplus / as sysdba
Connected.
SQL> ALTER PLUGGABLE DATABASE pdb2 OPEN;

Pluggable database altered.

SQL> @apexins.sql SYSAUX SYSAUX TEMP /i/

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Performing installation in multitenant container database in the
background.
The installation progress is spooled into apexins_cdb*.log
files.

Please wait...
```

```
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/apex/apexins_cdb_catcon
_5681.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/apex/apexins_cdb*.log]
files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/apex/apexins_cdb_*.lst]
files for spool files, if any
…
    SQL>   2
    Pluggable database altered.

    SQL> SQL>
    SQL> Disconnected from Oracle Database 12c Enterprise
Edition Release 12.2.0.1.0 - 64bit Production
  ] end of output produced in exec_DB_script
Installation completed. Log files for each container can be
found in:

apexins_cdb*.log

You can quickly scan for ORA errors or compilation errors by
using a utility
like grep:

grep ORA- *.log
grep PLS- *.log
```

```
SQL>
```

```
SQL> SELECT comp_name, status, version FROM dba_registry
     WHERE  comp_name like '%Appli%Exp%';

COMP_NAME                              STATUS       VERSION
-------------------------------------- ------------ -----------
Oracle Application Express             VALID        5.0.4.00.12

SQL>
```

d. In *session 12.1*, back up pdb1.

```
$ rman target /

Recovery Manager: Release 12.1.0.2.0 - Production on Wed Mar 30
21:39:41 2016

connected to target database: CDB1 (DBID=876442563)

RMAN> BACKUP PLUGGABLE DATABASE pdb1 PLUS ARCHIVELOG;

Starting backup at 09-SEP-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
…
Starting Control File and SPFILE Autobackup at 09-SEP-16
piece
handle=/u03/app/oracle/fast_recovery_area/CDB1/autobackup/2016_0
9_09/o1_mf_s_922098145_cx53c2mw_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 09-SEP-16

RMAN> EXIT;
$
```

3. Unplug pdb1.

```
$ rm /tmp/xmlfilePDB1.xml
rm: cannot remove `/tmp/xmlfilePDB1.xml': No such file or
directory
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE pdb1 CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb1
                    UNPLUG INTO '/tmp/xmlfilePDB1.xml';
  2
Pluggable database altered.

SQL> SELECT pdb_name, status FROM cdb_pdbs
     WHERE  pdb_name = 'PDB1';
```

```
   2

PDB_NAME     STATUS
------------ ---------
PDB1         UNPLUGGED

SQL> EXIT
$
```

4. Plug `pdb1` into `cdb2`.
   a. Before performing the plugging operation, you can optionally check whether the unplugged `pdb1` is compatible with `cdb2`. Execute the `DBMS_PDB.CHECK_PLUG_COMPATIBILITY` function in ***session 12.2***:

```
DECLARE
  compat BOOLEAN := FALSE;
  BEGIN
  compat := DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
  pdb_descr_file => '/tmp/xmlfilePDB1.xml', pdb_name => 'pdb1');
  if compat then
  DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? YES');
  else DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? NO');
  end if;
end;
/
```

```
SQL> SET serveroutput on
SQL> DECLARE
  compat BOOLEAN := FALSE;
  BEGIN
  compat := DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
  pdb_descr_file => '/tmp/xmlfilePDB1.xml', pdb_name => 'pdb1');
  if compat then
  DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? YES');
  else DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? NO');
  end if;
end;
/
2    3    4    5    6    7    8    9   10   11
Is pluggable compatible? NO

PL/SQL procedure successfully completed.

SQL>
```

b. If the value returned is YES, you can immediately proceed with step d.
If the value returned is NO, examine the `PDB_PLUG_IN_VIOLATIONS` view to see why it is not compatible.

```
SQL> SELECT message, action FROM pdb_plug_in_violations
     WHERE  name = 'PDB1';
  2
MESSAGE
----------------------------------------------------------------
ACTION
----------------------------------------------------------------
PDB's version does not match CDB's version: PDB's version
12.1.0.2.0 . CDB's version 12.2.0.1.0.
Either upgrade the PDB or reload the components in the PDB.

APEX mismatch: PDB installed version 4.2.5.00.08 CDB installed
version 5.0.4.00.12
Install, upgrade, or patch APEX in the PDB or the CDB


CDB parameter sga_target mismatch: Previous 2352M Current 1504M
Please check the parameter in the current CDB


CDB parameter compatible mismatch: Previous '12.1.0.2.0' Current
'12.2.0'
Please check the parameter in the current CDB


CDB parameter pga_aggregate_target mismatch: Previous 783M
Current 501M
Please check the parameter in the current CDB


Undo mode mismatch: PDB using SHARED undo.  CDB using LOCAL
undo.
Either create an undo tablespace in the PDB or be aware that the
CDB will not look at undo in the PDB.



6 rows selected.


SQL>
```

Q/ Which of the 6 messages are important to fix before upgrading?

A/ Some of the messages refer to CDB's version and `COMPATIBLE` parameter mismatch which are going to be fixed by upgrading the PDB. Other messages refer to initialization parameter values which can be updated after the upgrade. APEX version of the plugged PDB is lower than the APEX version recently installed in the target CDB. You'll have to upgrade the PDB APEX version to the one of the target CDB. Shared undo mode used in the PDB does not have an

UNDO tablespace. Will there be an UNDO tablespace created after the upgrade operation in the CDB using local undo mode? You do not want to set the target CDB to shared undo mode. Therefore, you will verify that an undo tablespace in the upgraded PDB will have been created.

c. Plug `pdb1` into `cdb2` as a regular PDB. Copy the original files into the `/u02/app/oracle/oradata/cdb2/pdb1` directory.

```
SQL> !mkdir /u02/app/oracle/oradata/cdb2/pdb1
```

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SESSION SET DB_CREATE_FILE_DEST =
                '/u02/app/oracle/oradata/cdb2/pdb1';
  2
Session altered.

SQL> CREATE PLUGGABLE DATABASE pdb1
            USING '/tmp/xmlfilePDB1.xml'
            FILE_NAME_CONVERT =
                          ('/u02/app/oracle/oradata/cdb1/pdb1',
                           '/u02/app/oracle/oradata/cdb2/pdb1')
            COPY;
    2    3    4    5    6

Pluggable database created.

SQL>
```

5. Open `pdb1` in `cdb2` in `UPGRADE` mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN UPGRADE;

Pluggable database altered.

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB2                           READ WRITE NO
         4 HR_ROOT                        READ WRITE NO
         5 OPERATIONS                     READ WRITE NO
         6 PDB1                           MIGRATE    YES

SQL>
```

6. Set the archivelog recovery destination size to a large value.

```
SQL> ALTER SYSTEM SET db_recovery_file_dest_size='40G'
           SCOPE = both;
  2
System altered.

SQL> EXIT
$
```

7. Upgrade `pdb1` in `cdb2`. The `-c` parameter is used to list the container(s) for which recommendations were provided before upgrade.

```
$ cd $ORACLE_HOME/rdbms/admin
$ $ORACLE_HOME/perl/bin/perl catctl.pl -c 'PDB1' catupgrd.sql
Argument list for [catctl.pl]
Run in                     c = PDB1
Do not run in              C = 0
Input Directory            d = 0
Echo OFF                   e = 1
Simulate                   E = 0
Forced cleanup             F = 0
Log Id                     i = 0
Child Process              I = 0
Log Dir                    l = 0
Priority List Name         L = 0
Upgrade Mode active        M = 0
SQL Process Count          n = 0
SQL PDB Process Count      N = 0
Open Mode Normal           o = 0
Start Phase                p = 0
End Phase                  P = 0
Reverse Order              r = 0
AutoUpgrade Resume         R = 0
Script                     s = 0
Serial Run                 S = 0
RO User Tablespaces        T = 0
Display Phases             y = 0
Debug catcon.pm            z = 0
Debug catctl.pl            Z = 0

catctl.pl VERSION: [12.2.0.1.0]
           STATUS: [production]
            BUILD: [RDBMS_12.2.0.1.0_LINUX.X64_160905]
```

```
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
/u01/app/oracle/product/12.2.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/12.2.0/dbhome_1]

Analyzing file
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/catupgrd.sql

Log file directory = [/tmp/cfgtoollogs/upgrade20160909122350]

catcon: ALL catcon-related output will be written to
[/tmp/cfgtoollogs/upgrade20160909122350/catupgrd_catcon_25459.ls
t]
catcon: See
[/tmp/cfgtoollogs/upgrade20160909122350/catupgrd*.log] files for
output generated by scripts
catcon: See
[/tmp/cfgtoollogs/upgrade20160909122350/catupgrd_*.lst] files
for spool files, if any

Number of Cpus        = 1
Database Name         = cdb2
DataBase Version      = 12.2.0.1.0
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354/catupgrd_catcon_25459.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354/catupgrd*.log] files for output generated by
scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354/catupgrd_*.lst] files for spool files, if any

Log file directory =
[/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354]

PDB Parallel SQL Process Count = [2] is higher or equal to CPU
Count = [1]
Concurrent PDB Upgrades defaulting to CPU Count [1]
Parallel SQL Process Count (PDB)      = 2
Parallel SQL Process Count (CDB$ROOT) = 4
Concurrent PDB Upgrades               = 1
```

```
Generated PDB Inclusion:[PDB1]
CDB$ROOT  Open Mode = [OPEN]

Start processing of PDB1
[/u01/app/oracle/product/12.2.0/dbhome_1/perl/bin/perl catctl.pl
-c 'PDB1' -I -i pdb1 -n 2 -l
/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrade
20160909122354 catupgrd.sql]

Argument list for [catctl.pl]
Run in                c = PDB1
Do not run in         C = 0
Input Directory       d = 0
Echo OFF              e = 1
Simulate              E = 0
Forced cleanup        F = 0
Log Id                i = pdb1
Child Process         I = 1
Log Dir               l =
/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrade
20160909122354
Priority List Name    L = 0
Upgrade Mode active   M = 0
SQL Process Count     n = 2
SQL PDB Process Count N = 0
Open Mode Normal      o = 0
Start Phase           p = 0
End Phase             P = 0
Reverse Order         r = 0
AutoUpgrade Resume    R = 0
Script                s = 0
Serial Run            S = 0
RO User Tablespaces   T = 0
Display Phases        y = 0
Debug catcon.pm       z = 0
Debug catctl.pl       Z = 0

catctl.pl VERSION: [12.2.0.1.0]
           STATUS: [production]
            BUILD: [RDBMS_12.2.0.1.0_LINUX.X64_160905]
```

```
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
/u01/app/oracle/product/12.2.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/12.2.0/dbhome_1]

Analyzing file
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/catupgrd.sql

Log file directory =
[/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354]

catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354/catupgrdpdb1_catcon_25613.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354/catupgrdpdb1*.log] files for output generated by
scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354/catupgrdpdb1_*.lst] files for spool files, if
any

Number of Cpus        = 1
Database Name         = cdb2
DataBase Version      = 12.2.0.1.0
Generated PDB Inclusion:[PDB1]
CDB$ROOT  Open Mode = [OPEN]
Components in [PDB1]
    Installed [APEX APS CATALOG CATJAVA CATPROC CONTEXT DV
JAVAVM OLS ORDIM OWM SDO XDB XML XOQ]
Not Installed [EM MGW ODM RAC WK]

------------------------------------------------------
Phases [0-117]         Start Time:[2016_09_09 12:24:14]
Container Lists Inclusion:[PDB1] Exclusion:[NONE]
------------------------------------------------------
***********   Executing Change Scripts   ***********
Serial   Phase #:0    [PDB1] Files:1
…
****************   Upgrading APEX   ****************
Restart  Phase #:105  [PDB1] Files:1    Time: 1s
Serial   Phase #:106  [PDB1] Files:1    Time: 2633s
```

```
Restart  Phase #:107  [PDB1] Files:1    Time: 1s
***********   Final Component scripts    ***********
Serial   Phase #:108  [PDB1] Files:1    Time: 3s
*************   Final Upgrade scripts   ************
Serial   Phase #:109  [PDB1] Files:1    Time: 352s
**********   End PDB Application Upgrade   *********
Serial   Phase #:110  [PDB1] Files:1    Time: 3s
*******************   Migration   ******************
Serial   Phase #:111  [PDB1] Files:1    Time: 117s
Serial   Phase #:112  [PDB1] Files:1    Time: 7s
Serial   Phase #:113  [PDB1] Files:1    Time: 85s
*****************   Post Upgrade   *****************
Serial   Phase #:114  [PDB1] Files:1    Time: 58s
****************   Summary report   ****************
Serial   Phase #:115  [PDB1] Files:1    Time: 4s
Serial   Phase #:116  [PDB1] Files:1    Time: 5s
Serial   Phase #:117  [PDB1] Files:1     Time: 0s

------------------------------------------------------
Phases [0-117]         End Time:[2016_09_09 14:37:42]
Container Lists Inclusion:[PDB1] Exclusion:[NONE]
------------------------------------------------------

Grand Total Time: 8033s [PDB1]

 LOG FILES:
(/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354/catupgrdpdb1*.log)

Upgrade Summary Report Located in:
/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrade
20160909122354/upg_summary.log

Total Upgrade Time:          [0d:2h:13m:53s]

     Time: 8059s For PDB(s)

Grand Total Time: 8059s

 LOG FILES:
(/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/upgrad
e20160909122354/catupgrd*.log)
```

```
Grand Total Upgrade Time:    [0d:2h:14m:19s]
$
```

8. Finally execute the `postupgrade_fixups_PDB1.sql`and `utlrp.sql` scripts in the upgraded PDB to complete the upgrade step.
   a. First open the upgraded PDB.

```
$ sqlplus / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN;

Pluggable database altered.

SQL> EXIT
$
```

   b. Execute the postupgrade scripts, namely `$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups_PDB1.sql` and `$ORACLE_HOME/rdbms/admin/utlrp.sql`.

```
$ cd $ORACLE_HOME/rdbms/admin
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c PDB1 -b postupgrade
$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups_PDB1
.sql
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_catcon_30525.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_*.lst] files for spool files, if any
catcon.pl: completed successfully
$
```

Note: Use the `-b` mandatory parameter for the base name of log files.

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c PDB1 -b postupgrade
$ORACLE_HOME/rdbms/admin/utlrp.sql
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_catcon_32552.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_*.lst] files for spool files, if any
catcon.pl: completed successfully
```

```
$
```

c. Run `utlu122s.sql` to verify that all issues have been fixed.

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c PDB1 -b postupgrade
$ORACLE_HOME/rdbms/admin/utlu122s.sql
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_catcon_2587.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_*.lst] files for spool files, if any
catcon.pl: completed successfully
$
```

Q/ How do you get details about issues fix?

A/ In the output above, you know the directory where the output log files are generated and their names, `/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade*.log`. Read the `/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade0.log` file.

```
$ more
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade0
.log
SQL*Plus: Release 12.2.0.1.0 Production on Fri Sep 9 15:09:24
2016


Copyright (c) 1982, 2016, Oracle.  All rights reserved.
SQL> Connected.
SQL>   2
Session altered.

SQL>   2
Session altered.

SQL>   2
Session altered.

SQL> SQL>
SQL>   2
Session altered.

SQL> SQL>   2
Session altered.
```

```
NOW_CONNECTED_TO
---------------------------------------------------------------
==== Current Container = PDB1 Id = 5 ====

SQL>
NOW_CONNECTED_TO
---------------------------------------------------------------
==== Current Container = PDB1 Id = 5 ====

SQL>   2
CATCONSECTION
-----------------------------------
==== CATCON EXEC IN CONTAINERS ====

SQL>
BEGIN_RUNNING
---------------------------------------------------------------
====
@/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/utlu122s.sq
l Container:PDB1 Id:5 16-09-09 03:09:25 Proc:0 ====

SQL>
BEGIN_RUNNING
---------------------------------------------------------------
====
@/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/utlu122s.sq
l Container:PDB1 Id:5 16-09-09 03:09:25 Proc:0 ====

SQL>   2
Session altered.

SQL>   2
Session altered.

SQL>

Oracle Database 12.2 Post-Upgrade Status Tool           09-09-
2016 15:09:26
                                [PDB1]

Component                               Current         Version
Elapsed Time
```

```
Name                                        Status          Number
HH:MM:SS


Oracle Server                                 VALID      12.2.0.1.0
00:31:52
JServer JAVA Virtual Machine                  VALID      12.2.0.1.0
00:12:34
Oracle Real Application Clusters     OPTION OFF      12.2.0.1.0
00:00:00
Oracle Workspace Manager                      VALID      12.2.0.1.0
00:01:57
OLAP Analytic Workspace                       VALID      12.2.0.1.0
00:00:35
Oracle OLAP API                               VALID      12.2.0.1.0
00:00:27
Oracle Label Security                         VALID      12.2.0.1.0
00:00:18
Oracle XDK                                    VALID      12.2.0.1.0
00:03:34
Oracle Text                                   VALID      12.2.0.1.0
00:01:10
Oracle XML Database                           VALID      12.2.0.1.0
00:04:18
Oracle Database Java Packages                 VALID      12.2.0.1.0
00:00:24
Oracle Multimedia                             VALID      12.2.0.1.0
00:04:56
Oracle Application Express                    VALID     5.0.4.00.12
00:43:47
Oracle Database Vault                         VALID      12.2.0.1.0
00:01:20
Final Actions
00:07:51
Post Upgrade
00:00:54
Post Compile
00:17:47


Total Upgrade Time: 02:28:49 [PDB1]


Database time zone version is 18. It is older than current
release time
zone version 26. Time zone upgrade is needed using the DBMS_DST
package.
```

```
Summary Report File =
/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/cdb2/u
pgrade20160909122354/upg_summary.log

15:09:27 SQL>
15:09:27 SQL>
END_RUNNING
----------------------------------------------------------------
====
@/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/utlu122s.sq
l Container:PDB1 Id:5 16-09-09 03:09:27 Proc:0 ====

1 row selected.

Elapsed: 00:00:00.01
15:09:27 SQL>
END_RUNNING
----------------------------------------------------------------
====
@/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/utlu122s.sq
l Container:PDB1 Id:5 16-09-09 03:09:27 Proc:0 ====



1 row selected.

Elapsed: 00:00:00.00
15:09:27 SQL> 15:09:27 SQL>
15:09:27 SQL> 15:09:27   2
Session altered.

Elapsed: 00:00:00.01
15:09:27 SQL> 15:09:27 SQL>
15:09:27 SQL> ========== PROCESS ENDED ==========
15:09:27 SQL> Disconnected from Oracle Database 12c Enterprise
Edition Release 1
2.2.0.1.0 - 64bit Production
$
```

9. In *session 12.1*, now that the PDB is successfully upgraded to 12.2, you drop the source 12.1 PDB. If it had not been successfully upgraded, it was still possible to plug the source PDB back to the source CDB.

```
$ sqlplus / AS SYSDBA

SQL> DROP PLUGGABLE DATABASE pdb1 INCLUDING DATAFILES;
```

```
Pluggable database dropped.

SQL> EXIT
$
```

10. Your final goal was to upgrade the regular `pdb1` as the `research` application PDB in the `hr_root` application container in `cdb2`.
    In *session 12.2*, use the `script_pdb1_convert.sql` to unplug `pdb1` and plug it as `research` in the `hr_root` application container in `cdb2`. This conversion operation is identical to Practice 4-3.

```
$ sqlplus / AS SYSDBA

SQL> @$HOME/labs/APP/script_pdb1_convert
…
Pluggable database dropped.

SQL> CONNECT hr_mgr@research
Enter password: ******
Connected.
SQL> SELECT count(*) FROM hr_mgr.mgr_tab;

  COUNT(*)
----------
         0

1 row selected.

SQL>
```

Verify that the `test.bigtab` table that existed before the upgrade is still present after the upgrade.

```
SQL> SELECT count(*) FROM test.bigtab;

  COUNT(*)
----------
     10000

1 row selected.

SQL>
```

Q1/ Which last manual operation were you asked to do after the upgrade?

A1/ Verify that the plugged PDB holds an undo tablespace.

```
SQL> CONNECT / AS SYSDBA
Connected.
```

```
SQL> SELECT property_name, property_value
      FROM   database_properties
      WHERE  property_name = 'LOCAL_UNDO_ENABLED';
  2    3
PROPERTY_NAME      PROPERTY_VALUE
------------------ --------------
LOCAL_UNDO_ENABLED TRUE

SQL> CONNECT sys@research AS SYSDBA
Enter password: ******
Connected.
SQL> SELECT tablespace_name, contents FROM dba_tablespaces
     WHERE  contents = 'UNDO';
  2
TABLESPACE_NAME                CONTENTS
------------------------------ ---------------------
UNDO_1                         UNDO

SQL> EXIT
$
```

Q2/ Is there another method to upgrade `pdb1`?

A2/ Data Pump export / import allows the data migration of a 12.1.0.2 PDB into a 12.2.0.1 CDB. If the PDB contains a huge amount of data, the conventional export / import will last a long time. A FULL Transportable export / import of the PDB will be faster. DBUA can also upgrade a PDB.

11. Remove all preupgrade and postupgrade scripts and log files.

```
$ cd /u01/app/oracle/cfgtoollogs/cdb1
$ rm -rf preupgrade
$
```

Practice 7-2: Plugging Remote PDBs Through XTTS

Overview

In this practice, you will perform a cross-platform PDB transport because you have to transport `pdb_orcl` from a CDB from a Linux platform to a CDB on a Solaris platform. The PDB transport from a platform to another endian platform consists of backing up by unplugging and then restoring by plugging. In the course setup, there is no Solaris platform available. You will imagine the Solaris platform to be the same server that you are practicing with, and hence the destination platform is still a Linux host.

Tasks

1. Before starting the practice, startup `ORCL` and execute the `$HOME/labs/APP/setup_pdb_orcl.sh` shell script creates the `pdb_orcl` regular PDB.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
SQL> STARTUP
ORACLE instance started.

Total System Global Area  884998144 bytes
Fixed Size                  4586464 bytes
Variable Size             398459936 bytes
Database Buffers          473956352 bytes
Redo Buffers                7995392 bytes
Database mounted.
Database opened.
SQL> EXIT
$
$ $HOME/labs/APP/setup_pdb_orcl.sh
…
$
```

2. Before backing up the PDB, ensure the prerequisites are satisfied.
   a. The database compatibility should equal or greater than12.2

```
$ sqlplus / AS SYSDBA

SQL> SHOW PARAMETER COMPATIBLE

NAME                            TYPE        VALUE
------------------------------ ----------- --------------------
compatible                      string      12.2.0
noncdb_compatible               boolean     FALSE
SQL>
```

b. The PDB must be closed before transportation starts.

```
SQL> ALTER PLUGGABLE DATABASE pdb_orcl CLOSE;

Pluggable database altered.

SQL> EXIT
$
```

3. Determine the location of the conversion. In the first case, you decide to convert the PDB datafiles on the source platform.

a. Back up and convert the data files on the source platform.

```
$ rm -rf /home/oracle/backup
$ mkdir -p /home/oracle/backup/ORCL
$
```

1) Try as if you would unplug, back up, and convert the files for a Solaris[tm] OE (64-bit) which has big endianess.

```
$ rman target /

connected to target database: ORCL (DBID=1434391901)

RMAN> BACKUP TO PLATFORM 'Solaris[tm] OE (64-bit)'
             UNPLUG INTO '/tmp/pdb_orcl.xml'
             PLUGGABLE DATABASE pdb_orcl
             FORMAT
'/home/oracle/backup/ORCL/transportsolaris_%U';
2> 3> 4>

Starting backup at 22-MAR-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=873 device type=DISK
RMAN-00571: ===================================================
RMAN-00569: =========== ERROR MESSAGE STACK FOLLOWS ===========
RMAN-00571: ===================================================
RMAN-03002: failure of backup command at 03/22/2016 10:18:32
RMAN-08421: running UNPLUG or PLUG for cross-endian platform not
supported
$
```

Q1/ Why does it fail?
A1/ Cross-platform transportable PDB is possible only when the source and destinations platforms have the same endian format. Because `SYSTEM` tablespaces cannot be converted across endianness, the solution would be full transportable export/import, using RMAN CONVERT on the data files for the endianness conversion.

Q2/ Would a cross-platform transportable PDB to Solaris Operating System (x86-64) be successful?

```
RMAN> BACKUP TO PLATFORM 'Solaris Operating System (x86-64)'
             UNPLUG INTO '/tmp/pdb_orcl.xml'
             PLUGGABLE DATABASE pdb_orcl
             FORMAT
'/home/oracle/backup/ORCL/transportsolaris_%U';
2> 3> 4>

Starting backup at 22-MAR-16
using channel ORA_DISK_1
running UNPLUG on the specified pluggable database: PDB_ORCL

UNPLUG file path : /tmp/pdb_orcl.xml
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00155
name=/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95
E0532633960AB2C1/datafile/o1_mf_sysaux_cgzor9oz_.dbf
input datafile file number=00154
name=/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95
E0532633960AB2C1/datafile/o1_mf_system_cgzor9kx_.dbf
input datafile file number=00156
name=/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95
E0532633960AB2C1/datafile/o1_mf_undotbs1_cgzor9p0_.dbf
channel ORA_DISK_1: starting piece 1 at 22-MAR-16
channel ORA_DISK_1: finished piece 1 at 22-MAR-16
piece
handle=/home/oracle/backup/ORCL/transportsolaris_3jr141ek_1_1
tag=TAG20160322T101955 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:26
Finished backup at 22-MAR-16



RMAN> EXIT
$
```

A2/ Yes, it is successful because the Solaris Operating System (x86-64) has the same endian format as Linux, little endian format.

2) Because we do not have a Solaris Operating System (x86-64) target, we will back up and convert the files for Linux, which has little endianess like the source host.

   Q/ If you immediately executed the `BACKUP UNPLUG INTO /tmp/pdb_orcl.xml` using the xml file recently created for Solaris, the backup would fail with the following error `ORA-65343: cannot unplug pluggable database PDB_ORCL`.

A/ The PDB is considered unplugged by the first `BACKUP UNPLUG INTO` that succeeded.

```
$ $HOME/labs/APP/setup_pdb_orcl.sh
…
$ sqlplus / as sysdba
SQL> ALTER PLUGGABLE DATABASE pdb_orcl CLOSE;

Pluggable database altered.

SQL> EXIT
$ rm -rf /home/oracle/backup
$ mkdir -p /home/oracle/backup/ORCL
$ rman target /

connected to target database: ORCL (DBID=1434391901)

RMAN> BACKUP TO PLATFORM 'Linux x86 64-bit'
            UNPLUG INTO '/tmp/pdb_orcl.xml'
            PLUGGABLE DATABASE pdb_orcl
            FORMAT '/home/oracle/backup/ORCL/transport_%U';
2> 3> 4>

Starting backup at 22-MAR-16
using channel ORA_DISK_1
running UNPLUG on the specified pluggable database: PDB_ORCL
UNPLUG file path : /tmp/pdb_orcl.xml
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00155
name=/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95
E0532633960AB2C1/datafile/o1_mf_sysaux_cgzor9oz_.dbf
input datafile file number=00154
name=/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95
E0532633960AB2C1/datafile/o1_mf_system_cgzor9kx_.dbf
input datafile file number=00156
name=/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95
E0532633960AB2C1/datafile/o1_mf_undotbs1_cgzor9p0_.dbf
channel ORA_DISK_1: starting piece 1 at 22-MAR-16
channel ORA_DISK_1: finished piece 1 at 22-MAR-16
piece handle=/home/oracle/backup/ORCL/transport_3kr141ie_1_1
tag=TAG20160322T102159 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 22-MAR-16
```

```
RMAN> EXIT
$
```

Q/ What is the new `UNPLUG INTO` clause useful for?

A/ The new `UNPLUG INTO` clause creates the XML file containing the metadata of the PDB—tablespaces list, datafiles list, options, and parameters values.

b. Because the source and target PDB are the same host due to our practice configuration, drop the source PDB before plugging the PDB in the target CDB. If the destination host was another host than the source host, you would transfer the files to the destination host, the backupset and xml file.

```
$ sqlplus / AS SYSDBA

SQL> DROP PLUGGABLE DATABASE pdb_orcl including datafiles;

Pluggable database dropped.

SQL> EXIT
$
```

c. In the target CDB, restore the converted data files by plugging.

1) Create a directory for the plugged PDB.

```
$ mkdir /u01/app/oracle/oradata/ORCL/pdb_orcl_2
$
```

2) Restore the converted data files.

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

connected to target database: ORCL (DBID=1434391901)

RMAN> ALTER SYSTEM SET
DB_CREATE_FILE_DEST='/u01/app/oracle/oradata/ORCL/pdb_orcl_2';

using target database control file instead of recovery catalog
Statement processed

RMAN> RESTORE FROM PLATFORM 'Linux x86 64-bit'
              USING '/tmp/pdb_orcl.xml'
              FOREIGN pluggable database pdb_orcl to new
              FROM BACKUPSET
'/home/oracle/backup/ORCL/transport_3kr141ie_1_1';
2> 3> 4>
```

```

Starting restore at 22-MAR-16
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 all foreign files in backup piece
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/ORCL/transport_3kr141ie_1_1
channel ORA_DISK_1: restoring foreign file 155 to
/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_sysa
ux_ch27pykc_.dbf
channel ORA_DISK_1: restoring foreign file 154 to
/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_syst
em_ch27pymb_.dbf
channel ORA_DISK_1: restoring foreign file 156 to
/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_undo
tbs1_ch27pyoo_.dbf
channel ORA_DISK_1: foreign piece
handle=/home/oracle/backup/ORCL/transport_3kr141ie_1_1
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:16
channel ORA_DISK_1: plugging file 154 for
/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95E0532
633960AB2C1/datafile/o1_mf_system_cgzor9kx_.dbf
channel ORA_DISK_1: plugging file 155 for
/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95E0532
633960AB2C1/datafile/o1_mf_sysaux_cgzor9oz_.dbf
channel ORA_DISK_1: plugging file 156 for
/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95E0532
633960AB2C1/datafile/o1_mf_undotbs1_cgzor9p0_.dbf
channel ORA_DISK_1: plugging file 7 for
/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95E0532
633960AB2C1/datafile/o1_mf_temp_cgzor9p0_.dbf

Performing import of metadata...
Finished restore at 22-MAR-16

RMAN>
```

d. Open the transported PDB.

```
RMAN> ALTER PLUGGABLE DATABASE pdb_orcl OPEN;

Statement processed
```

```
RMAN> EXIT
$
```

e. Display the `test.bigtab` data.

```
$ sqlplus test@pdb_orcl

SQL> SELECT count(*) FROM test.bigtab;

  COUNT(*)
----------
     10000

SQL> EXIT
$
```

4. In the second case, you decide to convert the PDB datafiles on the destination platform.

a. Back up and convert the data files on the source platform.

```
$ rm /home/oracle/backup/ORCL/transport*
$ rm /tmp/pdb_orcl.xml
$ rman target /

connected to target database: ORCL (DBID=1434391901)

RMAN> BACKUP TO PLATFORM 'Linux x86 64-bit'
             UNPLUG INTO '/tmp/pdb_orcl.xml'
             PLUGGABLE DATABASE pdb_orcl
             FORMAT '/home/oracle/backup/ORCL/transport_%U';
2> 3> 4>
Starting backup at 22-MAR-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=141 device type=DISK
running UNPLUG on the specified pluggable database: PDB_ORCL
RMAN-00571: ===========================================================
RMAN-00569: =========== ERROR MESSAGE STACK FOLLOWS ===========
RMAN-00571: ===========================================================
RMAN-03002: failure of backup command at 03/22/2016 10:34:18
ORA-65025: Pluggable database PDB_ORCL is not closed on all
instances.

RMAN>
```

Q/ Why does the backup operation require the PDB to be closed?

A/ The backup operation uses the new `UNPLUG` clause, which requires the PDB to be closed.

```
RMAN> ALTER PLUGGABLE DATABASE pdb_orcl CLOSE;

Statement processed

RMAN> BACKUP TO PLATFORM 'Linux x86 64-bit'
             UNPLUG INTO '/tmp/pdb_orcl.xml'
             PLUGGABLE DATABASE pdb_orcl
             FORMAT '/home/oracle/backup/ORCL/transport_%U';
2> 3> 4>
Starting backup at 22-MAR-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=262 device type=DISK
running UNPLUG on the specified pluggable database: PDB_ORCL
UNPLUG file path : /tmp/pdb_orcl.xml
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00177
name=/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf
_sysaux_ch27pykc_.dbf
input datafile file number=00176
name=/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf
_system_ch27pymb_.dbf
input datafile file number=00178
name=/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf
_undotbs1_ch27pyoo_.dbf
channel ORA_DISK_1: starting piece 1 at 22-MAR-16
channel ORA_DISK_1: finished piece 1 at 22-MAR-16
piece handle=/home/oracle/backup/ORCL/transport_3lr142ao_1_1
tag=TAG20160322T103457 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:25
Finished backup at 22-MAR-16

RMAN>
```

b. Because the source and target PDB are the same host due to our practice configuration, drop the source PDB before plugging the PDB in the target CDB. If the destination host was another host than the source host, you would transfer the files to the destination host, the backupset and xml file.

```
RMAN> DROP PLUGGABLE DATABASE pdb_orcl including datafiles;

Statement processed

SQL> EXIT
$
```

c. In the target CDB, convert and restore the data files by plugging.

1) Create a directory for the plugged PDB.

```
$ rm -rf mkdir /u02/app/oracle/oradata/ORCL/pdb_orcl_2
$ mkdir /u02/app/oracle/oradata/ORCL/pdb_orcl_2
$
```

2) Restore the converted data files.

```
$ ls /home/oracle/backup/ORCL/transport*
/home/oracle/backup/ORCL/transport_3lr142ao_1_1
$ rman target /

connected to target database: ORCL (DBID=1434391901)

RMAN> ALTER SYSTEM SET
DB_CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/pdb_orcl_2';

using target database control file instead of recovery catalog
Statement processed

RMAN> RESTORE FROM PLATFORM 'Linux x86 64-bit'
              USING '/tmp/pdb_orcl.xml'
              FOREIGN pluggable database pdb_orcl to new
              FROM BACKUPSET
             '/home/oracle/backup/ORCL/transport_3lr142ao_1_1';
2> 3> 4> 5>
Starting restore at 22-MAR-16
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=508 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 all foreign files in backup piece
channel ORA_DISK_1: reading from backup piece
/home/oracle/backup/ORCL/transport_3lr142ao_1_1
channel ORA_DISK_1: restoring foreign file 177 to
/u02/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_sysa
ux_ch28bvqy_.dbf
channel ORA_DISK_1: restoring foreign file 176 to
/u02/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_syst
em_ch28bvw9_.dbf
channel ORA_DISK_1: restoring foreign file 178 to
/u02/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_undo
tbs1_ch28bw54_.dbf
```

```
channel ORA_DISK_1: foreign piece
handle=/home/oracle/backup/ORCL/transport_3lr142ao_1_1
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:56
channel ORA_DISK_1: plugging file 176 for
/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_syst
em_ch27pymb_.dbf
channel ORA_DISK_1: plugging file 177 for
/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_sysa
ux_ch27pykc_.dbf
channel ORA_DISK_1: plugging file 178 for
/u01/app/oracle/oradata/ORCL/pdb_orcl_2/ORCL/datafile/o1_mf_undo
tbs1_ch27pyoo_.dbf
channel ORA_DISK_1: plugging file 7 for
/u02/app/oracle/oradata/ORCL/pdb_orcl/ORCL/2E8E2ADC3CC22E95E0532
633960AB2C1/datafile/o1_mf_temp_cgzor9p0_.dbf

Performing import of metadata...
Finished restore at 22-MAR-16

RMAN>
```

d. Open the transported PDB.

```
RMAN> ALTER PLUGGABLE DATABASE pdb_orcl OPEN;

Statement processed

RMAN> EXIT
$
```

e. Display the `test.bigtab` data.

```
$ sqlplus test@pdb_orcl

SQL> SELECT count(*) FROM test.bigtab;

  COUNT(*)
----------
     10000

SQL> EXIT
$
```

Practice 7-3: Upgrading a 12.1.0.2 CDB to a 12.2.0.1 CDB *(Optional)*

Overview

In this practice, you will upgrade the Oracle Database 12.1.0.2 `cdb1` to Oracle Database 12.2.0.1. The operation takes a long time.

Once the CDB upgrade operation is started at task 3, you can complete other practices that take place in `ORCL` database.

Tasks

1. In *session 12.1*, prepare `cdb1` to be upgraded to Oracle Database 12.2.0.1.
   a. Execute the `setup_upgrade_CDB.sh` script to create two regular PDBs, `pdb1_1` and `pdb1_2` in `cdb1` that will be upgraded with the CDB.

```
$ $HOME/labs/APP/setup_upgrade_CDB.sh
…
$
```

   b. Execute the Pre-Upgrade Information Tool in the 12.1.0.2 `cdb1` by executing the `preupgrd.sql` script.

```
$ . oraenv
ORACLE_SID = [cdb2] ? cdb1
The Oracle base remains unchanged with value /u01/app/oracle
$
$ rm -rf /u01/app/oracle/cfgtoollogs/cdb1/preupgrade
$ cd /u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin
$ $ORACLE_HOME/jdk/bin/java -jar preupgrade.jar TERMINAL TEXT
Container:CDB$ROOT is in: READ WRITE status and it will be
processed.
Container:PDB$SEED is in: READ ONLY status and it will be
processed.
Container:PDB1_1 is in: READ WRITE status and it will be
processed.
Container:PDB1_2 is in: READ WRITE status and it will be
processed.


Processing: CDB$ROOT...


Processing: PDB$SEED...


Processing: PDB1_1...


Processing: PDB1_2...


Report generated by Oracle Database Pre-Upgrade Information Tool
Version 12.2.0.1.0
```

```
Upgrade-To version: 12.2.0.1.0
=======================================
Status of the database prior to upgrade
=======================================

      Database Name:  CDB1
     Container Name:  CDB$ROOT
       Container ID:  1
            Version:  12.1.0.2.0
         Compatible:  12.1.0.2.0
          Blocksize:  8192
           Platform:  Linux x86 64-bit
      Timezone File:  18
  Database log mode:  ARCHIVELOG
           Readonly:  FALSE

  Oracle Component              Upgrade Action   Current Status
  ----------------              --------------   --------------
  Oracle Server                  [to be upgraded]  VALID
  JServer JAVA Virtual Machine   [to be upgraded]  VALID
  Oracle XDK for Java            [to be upgraded]  VALID
  Real Application Clusters      [to be upgraded]  OPTION OFF
  Oracle Workspace Manager       [to be upgraded]  VALID
  OLAP Analytic Workspace        [to be upgraded]  VALID
  Oracle Label Security          [to be upgraded]  VALID
  Oracle Database Vault          [to be upgraded]  VALID
  Oracle Text                    [to be upgraded]  VALID
  Oracle XML Database            [to be upgraded]  VALID
  Oracle Java Packages           [to be upgraded]  VALID
  Oracle Multimedia              [to be upgraded]  VALID
  Oracle Spatial                 [to be upgraded]  VALID
  Oracle Application Express     [to be upgraded]  VALID
  Oracle OLAP API                [to be upgraded]  VALID

==============
BEFORE UPGRADE
==============
  Run <preupgradeLogDirPath>/preupgrade_fixups_CDB_ROOT.sql to
complete all of the BEFORE UPGRADE action items below marked
with '(AUTOFIXUP)'.

  REQUIRED ACTIONS
  ================
   + Adjust TABLESPACE SIZES as needed.
                             Auto    12.2.0.1.0
     Tablespace         Size  Extend  Min Size    Action
     ----------       ------  ------- ----------  -----
     SYSAUX           750 MB  ENABLED    1649 MB  None
     SYSTEM           800 MB  ENABLED    1304 MB  None
     TEMP             197 MB  DISABLED    150 MB  None
     UNDOTBS1         175 MB  ENABLED     400 MB  None

     Note that 12.2.0.1.0 minimum sizes are estimates.
     If you plan to upgrade multiple pluggable databases
concurrently, then you must ensure that the UNDO tablespace size
is equal to at least the number of pluggable databases that you
upgrade concurrently, multiplied by that minimum.  Failing to
allocate sufficient space can cause the upgrade to fail.



  RECOMMENDED ACTIONS
  ===================
+ (AUTOFIXUP) Gather SYS schema and stale data dictionary
statistics prior to database upgrade in off-peak time using:


      EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;


      EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS('SYS');


     Dictionary statistics do not exist or are stale (not up-to-
date).


     Dictionary statistics help the Oracle optimizer find
efficient SQL execution plans and are essential for proper
upgrade timing. Oracle recommends gathering dictionary
statistics in the last 24 hours before database upgrade.
  INFORMATION ONLY
  ================
   + Consider upgrading APEX manually, before the database
upgrade.
     The database contains APEX version 4.2.5.00.08 and will
need to be upgraded to at least version 5.0.3.00.02.
     To reduce database upgrade time, you can upgrade APEX
manually before the database upgrade.  Refer to My Oracle
Support Note 1088970.1 for information on APEX installation
upgrades.
```

```
=============
AFTER UPGRADE
=============

  Run <preupgradeLogDirPath>/postupgrade_fixups_CDB_ROOT.sql to
complete all of the AFTER UPGRADE action items below marked with
'(AUTOFIXUP)'.

  REQUIRED ACTIONS
  ================
  None

  RECOMMENDED ACTIONS
  ===================
   + Upgrade the database time zone version using the DBMS_DST
package.
     The database is using timezone datafile version 18 and the
target 12.2.0.1.0 database ships with timezone datafile version
25.
     Oracle recommends using the most recent timezone data.  For
further information, refer to My Oracle Support Note 1509653.1.

   + (AUTOFIXUP) Gather dictionary statistics after the upgrade
using the command:

       EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

     Oracle recommends gathering dictionary statistics after
upgrade.
     Dictionary statistics provide essential information to the
Oracle optimizer to help it find efficient SQL execution plans.
After a database upgrade, statistics need to be re-gathered as
there can now be tables that have significantly changed during
the upgrade or new tables that do not have statistics gathered
yet.

Report generated by Oracle Database Pre-Upgrade Information Tool
Version 12.2.0.1.0

Upgrade-To version: 12.2.0.1.0

=======================================
Status of the database prior to upgrade
=======================================
      Database Name:  CDB1
     Container Name:  PDB$SEED
       Container ID:  2
            Version:  12.1.0.2.0
         Compatible:  12.1.0.2.0
          Blocksize:  8192
           Platform:  Linux x86 64-bit
      Timezone File:  18
  Database log mode:  ARCHIVELOG
           Readonly:  FALSE

  Oracle Component              Upgrade Action    Current Status
  ----------------              --------------    --------------
  Oracle Server                 [to be upgraded]  VALID
  JServer JAVA Virtual Machine  [to be upgraded]  VALID
  Oracle XDK for Java           [to be upgraded]  VALID
  Real Application Clusters     [to be upgraded]  OPTION OFF
  Oracle Workspace Manager      [to be upgraded]  VALID
  OLAP Analytic Workspace       [to be upgraded]  VALID
  Oracle Label Security         [to be upgraded]  VALID
  Oracle Database Vault         [to be upgraded]  VALID
  Oracle Text                   [to be upgraded]  VALID
  Oracle XML Database           [to be upgraded]  VALID
  Oracle Java Packages          [to be upgraded]  VALID
  Oracle Multimedia             [to be upgraded]  VALID
  Oracle Spatial                [to be upgraded]  VALID
  Oracle Application Express     [to be upgraded]  VALID
  Oracle OLAP API                [to be upgraded]  VALID


==============
BEFORE UPGRADE
==============
  Run <preupgradeLogDirPath>/preupgrade_fixups_PDB_SEED.sql to
complete all of the BEFORE UPGRADE action items below marked
with '(AUTOFIXUP)'.


  REQUIRED ACTIONS
  ================
   + Adjust TABLESPACE SIZES as needed.
                              Auto    12.2.0.1.0
     Tablespace         Size  Extend  Min Size    Action
     ----------       ------  ------- ----------  -----
     SYSAUX           750 MB  ENABLED    1649 MB  None
     SYSTEM           800 MB  ENABLED    1304 MB  None
     TEMP             197 MB  DISABLED    150 MB  None
     UNDOTBS1         175 MB  ENABLED     400 MB  None



     Note that 12.2.0.1.0 minimum sizes are estimates.
     If you plan to upgrade multiple pluggable databases
concurrently, then you must ensure that the UNDO tablespace size
is equal to at least the number of pluggable databases that you
upgrade concurrently, multiplied by that minimum.  Failing to
allocate sufficient space can cause the upgrade to fail.


  RECOMMENDED ACTIONS
  ===================
   + (AUTOFIXUP) Gather SYS schema and stale data dictionary
statistics prior to database upgrade in off-peak time using:


      EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;


      EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS('SYS');


     Dictionary statistics do not exist or are stale (not up-to-
date).
     Dictionary statistics help the Oracle optimizer find
efficient SQL execution plans and are essential for proper
upgrade timing. Oracle recommends gathering dictionary
statistics in the last 24 hours before database upgrade.


  INFORMATION ONLY
  ================
   + Consider upgrading APEX manually, before the database
upgrade.
     The database contains APEX version 4.2.5.00.08 and will
need to be upgraded to at least version 5.0.3.00.02.
     To reduce database upgrade time, you can upgrade APEX
manually before the database upgrade.  Refer to My Oracle
Support Note 1088970.1 for information on APEX installation
upgrades.


=============
AFTER UPGRADE
=============
  Run <preupgradeLogDirPath>/postupgrade_fixups_PDB_SEED.sql to
complete all of the AFTER UPGRADE action items below marked with
'(AUTOFIXUP)'.
```

```
  REQUIRED ACTIONS
  ================
  None

  RECOMMENDED ACTIONS
  ===================
   + Upgrade the database time zone version using the DBMS_DST
package.
     The database is using timezone datafile version 18 and the
target 12.2.0.1.0 database ships with timezone datafile version
25.
     Oracle recommends using the most recent timezone data.  For
further information, refer to My Oracle Support Note 1509653.1.

   + (AUTOFIXUP) Gather dictionary statistics after the upgrade
using the command:

       EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;

     Oracle recommends gathering dictionary statistics after
upgrade.

     Dictionary statistics provide essential information to the
Oracle optimizer to help it find efficient SQL execution plans.
After a database upgrade, statistics need to be re-gathered as
there can now be tables that have significantly changed during
the upgrade or new tables that do not have statistics gathered
yet.

Report generated by Oracle Database Pre-Upgrade Information Tool
Version 12.2.0.1.0

Upgrade-To version: 12.2.0.1.0
=======================================
Status of the database prior to upgrade
=======================================

      Database Name:  CDB1
     Container Name:  PDB1_1
       Container ID:  3
            Version:  12.1.0.2.0
         Compatible:  12.1.0.2.0
          Blocksize:  8192
           Platform:  Linux x86 64-bit
      Timezone File:  18
  Database log mode:  ARCHIVELOG
           Readonly:  FALSE

  Oracle Component              Upgrade Action    Current Status
  ----------------              --------------    --------------
  Oracle Server                 [to be upgraded]  VALID
  JServer JAVA Virtual Machine  [to be upgraded]  VALID
  Oracle XDK for Java           [to be upgraded]  VALID
  Real Application Clusters     [to be upgraded]  OPTION OFF
  Oracle Workspace Manager      [to be upgraded]  VALID
  OLAP Analytic Workspace       [to be upgraded]  VALID
  Oracle Label Security         [to be upgraded]  VALID
  Oracle Database Vault         [to be upgraded]  VALID
  Oracle Text                   [to be upgraded]  VALID
  Oracle XML Database           [to be upgraded]  VALID
  Oracle Java Packages          [to be upgraded]  VALID
  Oracle Multimedia             [to be upgraded]  VALID
  Oracle Spatial                [to be upgraded]  VALID
  Oracle Application Express    [to be upgraded]  VALID
  Oracle OLAP API                 to be upgraded]  VALID


==============
BEFORE UPGRADE
==============
  Run <preupgradeLogDirPath>/preupgrade_fixups_PDB1_1.sql to
complete all of the BEFORE UPGRADE action items below marked
with '(AUTOFIXUP)'.

  REQUIRED ACTIONS
  ================
   + Adjust TABLESPACE SIZES as needed.
                             Auto    12.2.0.1.0
     Tablespace         Size  Extend  Min Size    Action
     ----------       ------  ------- ----------  -----
     SYSAUX           750 MB  ENABLED    1649 MB  None
     SYSTEM           800 MB  ENABLED    1304 MB  None
     TEMP             197 MB  DISABLED    150 MB  None
     UNDOTBS1         175 MB  ENABLED     400 MB  None

     Note that 12.2.0.1.0 minimum sizes are estimates.
     If you plan to upgrade multiple pluggable databases
concurrently, then you must ensure that the UNDO tablespace size
```

```
is equal to at least the number of pluggable databases that you
upgrade concurrently, multiplied by that minimum.  Failing to
allocate sufficient space can cause the upgrade to fail.


  RECOMMENDED ACTIONS
  ===================
   + (AUTOFIXUP) Gather SYS schema and stale data dictionary
statistics prior to database upgrade in off-peak time using:


      EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;


      EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS('SYS');


     Dictionary statistics do not exist or are stale (not up-to-
date).


     Dictionary statistics help the Oracle optimizer find
efficient SQL execution plans and are essential for proper
upgrade timing. Oracle recommends gathering dictionary
statistics in the last 24 hours before database upgrade.


  INFORMATION ONLY
  ================
   + Consider upgrading APEX manually, before the database
upgrade.
     The database contains APEX version 4.2.5.00.08 and will
need to be upgraded to at least version 5.0.3.00.02.
     To reduce database upgrade time, you can upgrade APEX
manually before the database upgrade.  Refer to My Oracle
Support Note 1088970.1 for information on APEX installation
upgrades.


=============
AFTER UPGRADE
=============
  Run <preupgradeLogDirPath>/postupgrade_fixups_PDB1_1.sql to
complete all of the AFTER UPGRADE action items below marked with
'(AUTOFIXUP)'.


  REQUIRED ACTIONS
  ================
  None


  RECOMMENDED ACTIONS
  ===================
   + Upgrade the database time zone version using the DBMS_DST
package.
     The database is using timezone datafile version 18 and the
target 12.2.0.1.0 database ships with timezone datafile version
25.
     Oracle recommends using the most recent timezone data.  For
further information, refer to My Oracle Support Note 1509653.1.


Report generated by Oracle Database Pre-Upgrade Information Tool
Version 12.2.0.1.0


Upgrade-To version: 12.2.0.1.0
=======================================
Status of the database prior to upgrade
=======================================


      Database Name:  CDB1
     Container Name:  PDB1_2
       Container ID:  3
            Version:  12.1.0.2.0
         Compatible:  12.1.0.2.0
          Blocksize:  8192
           Platform:  Linux x86 64-bit
      Timezone File:  18
  Database log mode:  ARCHIVELOG
           Readonly:  FALSE

  Oracle Component            Upgrade Action    Current Status
  ----------------            --------------    --------------
  Oracle Server                  [to be upgraded]  VALID
  JServer JAVA Virtual Machine   [to be upgraded]  VALID
  Oracle XDK for Java            [to be upgraded]  VALID
  Real Application Clusters      [to be upgraded]  OPTION OFF
  Oracle Workspace Manager       [to be upgraded]  VALID
  OLAP Analytic Workspace        [to be upgraded]  VALID
  Oracle Label Security          [to be upgraded]  VALID
  Oracle Database Vault          [to be upgraded]  VALID
  Oracle Text                    [to be upgraded]  VALID
  Oracle XML Database            [to be upgraded]  VALID
  Oracle Java Packages           [to be upgraded]  VALID
  Oracle Multimedia              [to be upgraded]  VALID
  Oracle Spatial                 [to be upgraded]  VALID
  Oracle Application Express     [to be upgraded]  VALID
  Oracle OLAP API                [to be upgraded]  VALID


==============
BEFORE UPGRADE
==============
  Run <preupgradeLogDirPath>/preupgrade_fixups_PDB1_2.sql to
complete all of the BEFORE UPGRADE action items below marked
with '(AUTOFIXUP)'.


  REQUIRED ACTIONS
  ================
   + Adjust TABLESPACE SIZES as needed.
                                                Auto
12.2.0.1.0
     Tablespace                        Size     Extend    Min
Size    Action
     ----------                     ----------  --------  ------
----  ------
     SYSAUX                            570 MB  ENABLED
1484 MB  None
     SYSTEM                            260 MB  ENABLED
761 MB  None
     TEMP                               20 MB  ENABLED
150 MB  None


     Note that 12.2.0.1.0 minimum sizes are estimates.
     If you plan to upgrade multiple pluggable databases
concurrently, then you must ensure that the UNDO tablespace size
is equal to at least the number of pluggable databases that you
upgrade concurrently, multiplied by that minimum.  Failing to
allocate sufficient space can cause the upgrade to fail.


  RECOMMENDED ACTIONS
  ===================
   + (AUTOFIXUP) Gather SYS schema and stale data dictionary
statistics prior to database upgrade in off-peak time using:


      EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;


      EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS('SYS');


     Dictionary statistics do not exist or are stale (not up-to-
date).
```

```
     Dictionary statistics help the Oracle optimizer find
efficient SQL execution plans and are essential for proper
upgrade timing. Oracle recommends gathering dictionary
statistics in the last 24 hours before database upgrade.


  INFORMATION ONLY
  ================
   + Consider upgrading APEX manually, before the database
upgrade.
     The database contains APEX version 4.2.5.00.08 and will
need to be upgraded to at least version 5.0.3.00.02.
     To reduce database upgrade time, you can upgrade APEX
manually before the database upgrade.  Refer to My Oracle
Support Note 1088970.1 for information on APEX installation
upgrades.


=============
AFTER UPGRADE
=============
  Run <preupgradeLogDirPath>/postupgrade_fixups_PDB1_2.sql to
complete all of the AFTER UPGRADE action items below marked with
'(AUTOFIXUP)'.


  REQUIRED ACTIONS
  ================
  None


  RECOMMENDED ACTIONS
  ===================
   + Upgrade the database time zone version using the DBMS_DST
package.
     The database is using timezone datafile version 18 and the
target 12.2.0.1.0 database ships with timezone datafile version
25.
     Oracle recommends using the most recent timezone data.  For
further information, refer to My Oracle Support Note 1509653.1.
   + (AUTOFIXUP) Gather dictionary statistics after the upgrade
using the
     command:


       EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;


     Oracle recommends gathering dictionary statistics after
upgrade.
     Dictionary statistics provide essential information to the
Oracle optimizer to help it find efficient SQL execution plans.
```

```
After a database upgrade, statistics need to be re-gathered as
there can now be tables that have significantly changed during
the upgrade or new tables that do not have statistics gathered
yet.

Preupgrade generated files:

/u01/app/oracle/cfgtoollogs/cdb1/preupgrade/preupgrade_fixups.sq
l
/u01/app/oracle/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups.s
ql
$
```

Q1/ Which containers in the CDB does the Pre-Upgrade Information Tool provide actions for?

A1/ The Pre-Upgrade Information Tool provides actions for all containers, the CDB root, CDB seed, and every PDB.

Q2/ What is the structure of the report generated by the Pre-Upgrade Information Tool?

A2/ The structure of the report is as follows:

- CDB$ROOT
- Status of the database prior to upgrade
- BEFORE UPGRADE
  - REQUIRED ACTIONS
  - RECOMMENDED ACTIONS
  - INFORMATION ONLY
- AFTER UPGRADE
  - REQUIRED ACTIONS
  - RECOMMENDED ACTIONS
- PDB$SEED
- Status of the database prior to upgrade
- BEFORE UPGRADE
  - REQUIRED ACTIONS
  - RECOMMENDED ACTIONS
  - INFORMATION ONLY
- AFTER UPGRADE
  - REQUIRED ACTIONS
  - RECOMMENDED ACTIONS
- Any other PDB
- Status of the database prior to upgrade
- BEFORE UPGRADE
  - REQUIRED ACTIONS
  - RECOMMENDED ACTIONS
  - INFORMATION ONLY

- AFTER UPGRADE
  - REQUIRED ACTIONS
  - RECOMMENDED ACTIONS

c. Read the recommendations of the Pre-Upgrade Information Tool performed on `cdb1`.

```
$ cd /u01/app/oracle/cfgtoollogs/cdb1/preupgrade/
$ ls preupgrade*.sql postupgrade*.sql
postupgrade_fixups_CDB_ROOT.sql  preupgrade_fixups_CDB_ROOT.sql
postupgrade_fixups_PDB1_1.sql    preupgrade_fixups_PDB1_1.sql
postupgrade_fixups_PDB1_2.sql    preupgrade_fixups_PDB1_2.sql
postupgrade_fixups_PDB_SEED.sql  preupgrade_fixups_PDB_SEED.sql
postupgrade_fixups.sql           preupgrade_fixups.sql
preupgrade_driver.sql            preupgrade_package.sql
$
```

Q/ Is there only one `preupgrade_fixups.sql` script and one `postupgrade_fixups.sql` script for the `cdb1` upgrade operation?

A/ No. There are as many `preupgrade_fixups.sql` scripts and `postupgrade_fixups.sql` scripts as PDBs, including the CDB root and CDB seed.

d. Execute each **`preupgrade_fixups.sql`** script for each container.

Q/ Based on your experience during the PDB upgrade in Practice 7-1, which faster method can you use to implement the recommended actions for each PDB?
A/ Because the recommended actions related to instance parameters are applicable only at the CDB root level as well as the dictionary statistics collection, execute the actions directly in the CDB root.

```
$ cd $ORACLE_HOME/rdbms/admin
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'CDB$ROOT' -b
preupgrade
$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade/preupgrade_fixups_CDB_R
OOT.sql
catcon: ALL catcon-related output will be written to
preupgrade_catcon_29782.lst
catcon: See preupgrade*.log files for output generated by
scripts
catcon: See preupgrade_*.lst files for spool files, if any
catcon.pl: completed successfully
$
```

e. Back up `cdb1`.

```
$ rman target /

connected to target database: CDB1 (DBID=870223878)

RMAN> ALTER SYSTEM SET db_recovery_file_dest_size='40G'
          SCOPE = both;
2>
```

```
Statement processed

RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
…
RMAN> EXIT
$
```

f. Prepare the PDBs priority list. You want `pdb1_2` be upgraded before `pdb1_1`. Create a SQL script containing the following code. The `SELECT` statement creates the PDBs list sorted and assigned a priority number. The lower priority numbers will be upgraded first. `CDB$ROOT` and `PDB$SEED` are always priorities 1 and 2 and cannot be changed. `CDB$ROOT` will always be processed first, and `PDB$SEED` will always be processed in the first set of upgrades. `catctl.pl` uses the list if you provide the name of the PDB list filename.

g. Create the `/tmp/priority.sql` script containing the following code and execute the SQL script:

```
SET NEWPAGE 0 SPACE 0 PAGESIZE 0 FEEDBACK OFF
SET HEADING OFF VERIFY OFF ECHO OFF TERMOUT OFF
SPOOL /tmp/priority_list.txt
SELECT CON_ID || ',' || NAME FROM V$CONTAINERS;
SPOOL off
```

```
$ sqlplus / AS SYSDBA
SQL> @/tmp/priority.sql
SQL> EXIT
$
```

Q/ How can the priority of `pdb1_1` and `pdb1_2` be changed so that `pdb1_2` is upgraded before `pdb1_1`?

A/ Edit the `/tmp/priority_list.txt` file and change the priority of the PDBs.

```
$ cat /tmp/priority_list.txt
1,CDB$ROOT
2,PDB$SEED
3,PDB1_1
4,PDB1_2
$
```

h. The result should display the following list:

```
1,CDB$ROOT
2,PDB$SEED
3,PDB1_2
4,PDB1_1
```

i. Shut down `cdb1`.

```
$ . oraenv
ORACLE_SID = [cdb1] ? cdb1
```

```
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / AS SYSDBA

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

2. Switch to *Session 12.2* to open `cdb1` in the Oracle Database 12.2.0.1 environment in `UPGRADE` mode.

```
$ . oraenv
ORACLE_SID = [cdb2] ?
The Oracle base remains unchanged with value /u01/app/oracle
$ export ORACLE_SID=cdb1
$ cp /u01/app/oracle/product/12.1.0/dbhome_1/dbs/spfilecdb1.ora
$ORACLE_HOME/dbs
$ sqlplus / AS SYSDBA

Connected to an idle instance.

SQL> STARTUP UPGRADE
ORACLE instance started.

Total System Global Area 666894336 bytes
Fixed Size                 4583960 bytes
Variable Size            390073832 bytes
Database Buffers         264241152 bytes
Redo Buffers               7995392 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE ALL OPEN UPGRADE;

Pluggable database altered.

SQL>
```

Q/ How can you check that the CDB is in UPGRADE mode?

A/ Display the open mode of the PDBs.

```
SQL> SHOW pdbs

```

```
    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       MIGRATE    YES
         3 PDB1_1                         MIGRATE    YES
         5 PDB1_2                         MIGRATE    YES
SQL> EXIT
$
```

3. Upgrade `cdb1`.

```
$ mkdir /tmp/upgrade_cdb1
$ cd $ORACLE_HOME/rdbms/admin
$ $ORACLE_HOME/perl/bin/perl catctl.pl -l /tmp/upgrade_cdb1 -L
/tmp/priority_list.txt catupgrd.sql

Argument list for [catctl.pl]
Run in                c = 0
Do not run in         C = 0
Input Directory       d = 0
Echo OFF              e = 1
Simulate              E = 0
Forced cleanup        F = 0
Log Id                i = 0
Child Process         I = 0
Log Dir               l = /tmp/upgrade_cdb1
Priority List Name    L = /tmp/priority_list.txt
Upgrade Mode active   M = 0
SQL Process Count     n = 0
SQL PDB Process Count N = 0
Open Mode Normal      o = 0
Start Phase           p = 0
End Phase             P = 0
Reverse Order         r = 0
AutoUpgrade Resume    R = 0
Script                s = 0
Serial Run            S = 0
RO User Tablespaces   T = 0
Display Phases        y = 0
Debug catcon.pm       z = 0
Debug catctl.pl       Z = 0

catctl.pl VERSION: [12.2.0.1.0]
           STATUS: [production]
            BUILD: [RDBMS_MAIN_LINUX.X64_160318]
```

```
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
/u01/app/oracle/product/12.2.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/12.2.0/dbhome_1]

Analyzing file ./catupgrd.sql

Log file directory = [/tmp/upgrade_cdb1]

catcon: ALL catcon-related output will be written to
[/tmp/upgrade_cdb1/catupgrd_catcon_2644.lst]
catcon: See [/tmp/upgrade_cdb1/catupgrd*.log] files for output
generated by scripts
catcon: See [/tmp/upgrade_cdb1/catupgrd_*.lst] files for spool
files, if any

Number of Cpus        = 8
Database Name         = cdb1
DataBase Version      = 12.1.0.2.0
Parallel SQL Process Count (PDB)      = 2
Parallel SQL Process Count (CDB$ROOT) = 8
Concurrent PDB Upgrades               = 4
Generated PDB Inclusion:[NONE]
Generated PDB Inclusion:[PDB$SEED PDB1_2 PDB1_1]
Components in [CDB$ROOT]
    Installed [APEX APS CATALOG CATJAVA CATPROC CONTEXT DV
JAVAVM OLS ORDIM OWM SDO XDB XML XOQ]
Not Installed [EM MGW ODM RAC WK]

------------------------------------------------------
Phases [0-108]         Start Time:[2016_03_31 03:32:39]
Container Lists Inclusion:[CDB$ROOT] Exclusion:[NONE]
------------------------------------------------------
***********   Executing Change Scripts   ***********
Serial   Phase #:0    [CDB$ROOT] Files:1
***************   Catalog Core SQL   ***************
Serial   Phase #:1    [CDB$ROOT] Files:5    Time: 645s
Restart  Phase #:2    [CDB$ROOT] Files:1    Time: 0s
***********   Catalog Tables and Views   ***********
Parallel Phase #:3    [CDB$ROOT] Files:19
…
```

```
------------------------------------------------------
Phases [0-108]         End Time:[2016_03_31 06:26:35]
Container Lists Inclusion:[CDB$ROOT] Exclusion:[NONE]
------------------------------------------------------


Start processing of PDB$SEED
[/u01/app/oracle/product/12.2.0/dbhome_1/perl/bin/perl catctl.pl
-l /tmp/upgrade_cdb1 -L /tmp/priority_list.txt -I -i pdb_seed -n
2 -c 'PDB$SEED' catupgrd.sql]

Start processing of PDB1_2
[/u01/app/oracle/product/12.2.0/dbhome_1/perl/bin/perl catctl.pl
-l /tmp/upgrade_cdb1 -L /tmp/priority_list.txt -I -i pdb1_2 -n 2
-c 'PDB1_2' catupgrd.sql]

Start processing of PDB1_1
[/u01/app/oracle/product/12.2.0/dbhome_1/perl/bin/perl catctl.pl
-l /tmp/upgrade_cdb1 -L /tmp/priority_list.txt -I -i pdb1_1 -n 2
-c 'PDB1_1' catupgrd.sql]

Argument list for [catctl.pl]
Run in                c = PDB1_1
Do not run in         C = 0
Input Directory       d = 0
Echo OFF              e = 1
Simulate              E = 0
Forced cleanup        F = 0
Log Id                i = pdb1_1
Child Process         I = 1
Log Dir               l = /tmp/upgrade_cdb1
Priority List Name    L = /tmp/priority_list.txt
Upgrade Mode active   M = 0
SQL Process Count     n = 2
SQL PDB Process Count N = 0

Argument list for [catctl.pl]
Open Mode Normal      o = 0

Argument list for [catctl.pl]
Run in                c = PDB$SEED
Run in                c = PDB1_2
Start Phase           p = 0
Do not run in         C = 0
Do not run in          C = 0
End Phase              P = 0
Input Directory        d = 0
Input Directory        d = 0
Reverse Order          r = 0
Echo OFF               e = 1
Echo OFF               e = 1
Simulate               E = 0
Simulate               E = 0
AutoUpgrade Resume     R = 0
Forced cleanup         F = 0
Forced cleanup         F = 0
Script                 s = 0
Log Id                 i = pdb1_2
Log Id                 i = pdb_seed
Child Process          I = 1
Child Process          I = 1
Log Dir                l = /tmp/upgrade_cdb1
Serial Run             S = 0
Priority List Name     L = /tmp/priority_list.txt
Log Dir                l = /tmp/upgrade_cdb1
RO User Tablespaces    T = 0
Priority List Name     L = /tmp/priority_list.txt
Display Phases         y = 0
Upgrade Mode active    M = 0
Debug catcon.pm        z = 0
Upgrade Mode active    M = 0
SQL Process Count      n = 2
Debug catctl.pl        Z = 0
SQL Process Count      n = 2
SQL PDB Process Count N = 0
SQL PDB Process Count N = 0

catctl.pl VERSION: [12.2.0.1.0]
           STATUS: [production]
            BUILD: [RDBMS_MAIN_LINUX.X64_160318]

Open Mode Normal       o = 0
Open Mode Normal       o = 0
Start Phase            p = 0
End Phase              P = 0
Start Phase            p = 0
```

```
Reverse Order          r = 0
End Phase              P = 0
AutoUpgrade Resume     R = 0
Reverse Order          r = 0
Script                 s = 0
AutoUpgrade Resume     R = 0
Serial Run             S = 0
RO User Tablespaces    T = 0
Script                 s = 0
Display Phases         y = 0
Serial Run             S = 0
Debug catcon.pm        z = 0
RO User Tablespaces    T = 0
Debug catctl.pl        Z = 0
Display Phases         y = 0
Debug catcon.pm        z = 0
Debug catctl.pl        Z = 0

catctl.pl VERSION: [12.2.0.1.0]
           STATUS: [production]
            BUILD: [RDBMS_MAIN_LINUX.X64_160318]

catctl.pl VERSION: [12.2.0.1.0]
           STATUS: [production]
            BUILD: [RDBMS_MAIN_LINUX.X64_160318]

/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
/u01/app/oracle/product/12.2.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
/u01/app/oracle/product/12.2.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/12.2.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/12.2.0/dbhome_1]
/u01/app/oracle/product/12.2.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/12.2.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/12.2.0/dbhome_1]


Analyzing file ./catupgrd.sql
Analyzing file ./catupgrd.sql

Analyzing file ./catupgrd.sql

Log file directory = [/tmp/upgrade_cdb1]

Log file directory = [/tmp/upgrade_cdb1]

Log file directory = [/tmp/upgrade_cdb1]

catcon: ALL catcon-related output will be written to
[/tmp/upgrade_cdb1/catupgrdpdb1_1_catcon_8344.lst]
catcon: ALL catcon-related output will be written to
[/tmp/upgrade_cdb1/catupgrdpdb1_2_catcon_8342.lst]
catcon: See [/tmp/upgrade_cdb1/catupgrdpdb1_1*.log] files for
output generated by scripts
catcon: See [/tmp/upgrade_cdb1/catupgrdpdb1_2*.log] files for
output generated by scripts
catcon: ALL catcon-related output will be written to
[/tmp/upgrade_cdb1/catupgrdpdb_seed_catcon_8340.lst]
catcon: See [/tmp/upgrade_cdb1/catupgrdpdb1_1_*.lst] files for
spool files, if any
catcon: See [/tmp/upgrade_cdb1/catupgrdpdb_seed*.log] files for
output generated by scripts
catcon: See [/tmp/upgrade_cdb1/catupgrdpdb1_2_*.lst] files for
spool files, if any
catcon: See [/tmp/upgrade_cdb1/catupgrdpdb_seed_*.lst] files for
spool files, if any

Number of Cpus        = 8
Number of Cpus        = 8
Number of Cpus        = 8
Database Name         = cdb1
Database Name         = cdb1
Database Name         = cdb1
DataBase Version      = 12.2.0.1.0
DataBase Version      = 12.2.0.1.0
DataBase Version      = 12.2.0.1.0
Generated PDB Inclusion:[PDB1_1]
Generated PDB Inclusion:[PDB$SEED]
Generated PDB Inclusion:[PDB1_2]
CDB$ROOT  Open Mode = [OPEN]
CDB$ROOT  Open Mode = [OPEN]
```

```
CDB$ROOT  Open Mode = [OPEN]
Components in [PDB1_1]
    Installed [APEX APS CATALOG CATJAVA CATPROC CONTEXT DV
JAVAVM OLS ORDIM OWM SDO XDB XML XOQ]
Not Installed [EM MGW ODM RAC WK]

------------------------------------------------------
Phases [0-108]         Start Time:[2016_03_31 06:26:57]
Container Lists Inclusion:[PDB1_1] Exclusion:[NONE]
------------------------------------------------------
Components in [PDB1_2]
    Installed [APEX APS CATALOG CATJAVA CATPROC CONTEXT DV
JAVAVM OLS ORDIM OWM SDO XDB XML XOQ]
Not Installed [EM MGW ODM RAC WK]

------------------------------------------------------
Phases [0-108]         Start Time:[2016_03_31 06:26:57]
Container Lists Inclusion:[PDB1_2] Exclusion:[NONE]
------------------------------------------------------
Components in [PDB$SEED]
    Installed [APEX APS CATALOG CATJAVA CATPROC CONTEXT DV
JAVAVM OLS ORDIM OWM SDO XDB XML XOQ]
Not Installed [EM MGW ODM RAC WK]

------------------------------------------------------
Phases [0-108]         Start Time:[2016_03_31 06:26:57]
Container Lists Inclusion:[PDB$SEED] Exclusion:[NONE]
------------------------------------------------------
***********   Executing Change Scripts   ***********
Serial   Phase #:0    [PDB1_1] Files:1 ***********   Executing
Change Scripts   ***********
Serial   Phase #:0    [PDB1_2] Files:1 ***********   Executing
Change Scripts   ***********
Serial   Phase #:0    [PDB$SEED] Files:1
…
Upgrade Summary Report Located in:
/tmp/upgrade_cdb1/upg_summary.log

Total Upgrade Time:          [0d:6h:52m:40s]
   Time: 555s
****************   Summary report   ****************
Serial   Phase #:106  [PDB$SEED] Files:1    Time: 2s
Serial   Phase #:107  [PDB$SEED] Files:1    Time: 13s
Serial   Phase #:108  [PDB$SEED] Files:1     Time: 0s

------------------------------------------------------
Phases [0-108]         End Time:[2016_03_31 13:25:20]
Container Lists Inclusion:[PDB$SEED] Exclusion:[NONE]
------------------------------------------------------

Grand Total Time: 25115s [PDB$SEED]

 LOG FILES: (/tmp/upgrade_cdb1/catupgrdpdb_seed*.log)

Upgrade Summary Report Located in:
/tmp/upgrade_cdb1/upg_summary.log

Total Upgrade Time:          [0d:0h:41m:51s]

     Time: 3163s For CDB$ROOT
     Time: 6419s For PDB(s)

Grand Total Time: 9582s

 LOG FILES: (/tmp/upgrade_cdb1/catupgrd*.log)

Upgrade Summary Report Located in:
/tmp/upgrade_cdb1/upg_summary.log

Grand Total Upgrade Time:    [0d:2h:39m:42s]
$
```

Q/ Was the priority set for upgrading `pdb1_1` and `pdb1_2` respected? Was `pdb1_2` upgraded before `pdb1_1`?

A/ Yes. Read from the alert.log file.

```
alter pluggable database PDB1_2 upgrade priority 3
Completed: alter pluggable database PDB1_2 upgrade priority 3
alter pluggable database PDB1_1 upgrade priority 4
Completed: alter pluggable database PDB1_1 upgrade priority 4
2016-03-31T06:26:45.971965+00:00
…
Pluggable database PDB1_2 opened in upgrade mode
2016-03-31T06:26:54.475684+00:00
Pluggable database PDB1_1 opened in upgrade mode
PDB1_2(5):Completed: alter pluggable database PDB1_2 open
upgrade
```

```
2016-03-31T06:26:54.644481+00:00
PDB1_1(3):Completed: alter pluggable database PDB1_1 open
upgrade
2016-03-31T06:27:09.541304+00:00
```

If you want to know the scripts and commands performed during the upgrade process, read the `/tmp/upgrade_cdb1/catupgrd*.log` log files.

4. Finally execute the `postupgrade_fixups.sql` and `utlrp.sql` scripts.

```
$ . oraenv
ORACLE_SID = [cdb1] ? cdb1
The Oracle base remains unchanged with value /u01/app/oracle
$ export ORACLE_HOME=/u01/app/oracle/product/12.2.0/dbhome_1
$ sqlplus / AS SYSDBA

SQL> SHOW pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1_1                         MOUNTED
```

```
         5 PDB1_2                         MOUNTED
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> EXIT
$
```

```
$ cd $ORACLE_HOME/rdbms/admin
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'CDB$ROOT' -b
postupgrade
$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups_CDB_
ROOT.sql
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_catcon_24118.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_*.lst] files for spool files, if any
catcon.pl: completed successfully
$
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'PDB$SEED' -b
postupgrade
$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups_PDB_
SEED.sql
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_catcon_5022.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_*.lst] files for spool files, if any
reset_seed_pdb_mode: output produced in exec_DB_script [


    SQL*Plus: Release 12.1.0.2.0 Production on Mon Sep 19
21:05:51 2016


    Copyright (c) 1982, 2014, Oracle.  All rights reserved.


    SQL> Connected.
    SQL>   2
    Session altered.


    SQL>   2
    Pluggable database altered.


    SQL>   2
    Pluggable database altered.


    SQL> SQL>
    SQL> Disconnected from Oracle Database 12c Enterprise
Edition Release 12.2.0.1.0 - 64bit Production
  ] end of output produced in exec_DB_script
reset_seed_pdb_mode: output produced in exec_DB_script [


    SQL*Plus: Release 12.1.0.2.0 Production on Mon Sep 19
21:05:57 2016


    Copyright (c) 1982, 2014, Oracle.  All rights reserved.


    SQL> Connected.
    SQL>   2
    Session altered.
    SQL>   2
    Pluggable database altered.

    SQL>   2
    Pluggable database altered.

    SQL> SQL>
    SQL> Disconnected from Oracle Database 12c Enterprise
Edition Release 12.2.0.1.0 - 64bit Production
  ] end of output produced in exec_DB_script
catcon.pl: completed successfully
$
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'PDB1_1' -b
postupgrade
$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups_PDB1
_1.sql
```

```
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_catcon_28643.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_*.lst] files for spool files, if any
catcon.pl: completed successfully
$
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'PDB1_2' -b
postupgrade
$ORACLE_BASE/cfgtoollogs/cdb1/preupgrade/postupgrade_fixups_PDB1
_2.sql
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_catcon_29017.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_*.lst] files for spool files, if any
catcon.pl: completed successfully
$
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -b postupgrade
$ORACLE_HOME/rdbms/admin/utlrp.sql
```

```
catcon: ALL catcon-related output will be written to
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_catcon_29729.lst]
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts
catcon: See
[/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin/postupgrade
_*.lst] files for spool files, if any
    SQL*Plus: Release 12.1.0.2.0 Production on Mon Sep 19
21:08:22 2016


    Copyright (c) 1982, 2014, Oracle.  All rights reserved.


    SQL> Connected.
    SQL>   2
    Session altered.

    SQL>   2
    Pluggable database altered.

    SQL>   2
    Pluggable database altered.

    SQL> SQL>
    SQL> Disconnected from Oracle Database 12c Enterprise
Edition Release 12.2.0.1.0 - 64bit Production
  ] end of output produced in exec_DB_script
catcon.pl: completed successfully$
```

Q1/ Is there another method to upgrade CDBs?

A1/ DBUA can upgrade a CDB.

A2/ Data Pump export / import can also upgrade a 12.1.0.2 CDB into a 12.2.0.1 CDB with the FULL Transportable or FULL conventional export / import.

5. Update the `cdb1` entry in `/etc/oratab` to set the `ORACLE_HOME` to 12.2.

```
cdb1:/u01/app/oracle/product/12.1.0/dbhome_1:N
➔
cdb1:/u01/app/oracle/product/12.2.0/dbhome_1:N
```

6. Restart the database and open all PDBs. Check that application data is accessible.

```
$ sqlplus / AS SYSDBA


SQL> SHUTDOWN IMMEDIATE
```

```
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.

Total System Global Area 1241513984 bytes
Fixed Size                  4577312 bytes
Variable Size             486541280 bytes
Database Buffers          738197504 bytes
Redo Buffers               12197888 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> CONNECT system@pdb1_1
Enter password: ******
Connected.
SQL> SELECT * FROM pdb1_1_user.smalltab;

LABEL
------------------------------
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab
DATA FROM source_user.smalltab

10 rows selected.

SQL> CONNECT system@pdb1_2
Enter password: ******
Connected.
SQL> SELECT count(*) FROM pdb1_2_user.smalltab;
```

```
  COUNT(*)
----------
        10

SQL> EXIT
$
```

7. Create the new `cdb1` password file in Oracle Database 12.2 environment.

```
$ cd /u01/app/oracle/product/12.2.0/dbhome_1/dbs
$ orapwd file=orapwcdb1 entries=5 password=oracle_4U
$
```

8. To release resources for the next practices, drop the `cdb1` database.

```
$ sqlplus / AS SYSDBA

SQL> SELECT name FROM v$database;

NAME
------------------------------
CDB1
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT RESTRICT
ORACLE instance started.

Total System Global Area 1241513984 bytes
Fixed Size                  4577312 bytes
Variable Size             486541280 bytes
Database Buffers          738197504 bytes
Redo Buffers               12197888 bytes
Database mounted.
SQL> DROP DATABASE;

Database dropped.

SQL> EXIT
$
```