



Integrated Cloud Applications & Platform Services



# Oracle Database: Managing Multitenant Architecture

Activity Guide

D105924GC10 | D106061

Learn more from Oracle University at [education.oracle.com](https://education.oracle.com)

ORACLE®

**Author**

Dominique Jeunot

**Technical Contributors  
and Reviewers**

Jean-François Verrier

Sravanti Tatiraju

Veerabhadra Rao Putrevu

**Graphic Editor**

Kavya Bellur

**Editors**

Moushmi Mukherjee

Adrita Biswas

**Publishers**

Sujatha Nagendra

Srividya Rameshkumar

Michael Sebastian Almeida

Raghunath M

1002012019

**Copyright © 2019, Oracle and/or its affiliates. All rights reserved.**

**Disclaimer**

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

**Restricted Rights Notice**

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

**U.S. GOVERNMENT RIGHTS**

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

**Trademark Notice**

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

# Table of Contents

---

<b>Course Practice Environment: Security Credentials .....</b>	<b>7</b>
Course Practice Environment: Security Credentials.....	8
<b>Practices for Lesson 1: CDB Basics.....</b>	<b>9</b>
Practices for Lesson 1: Overview .....	10
Practice 1-1: Discovering Practices Environment.....	12
Practice 1-2: Adding a CDB as a New Target in Enterprise Manager Cloud Control.....	13
Practice 1-3: Checking Named Credentials.....	15
Practice 1-4: Using Enterprise Manager Express.....	17
<b>Practices for Lesson 2: CDB and Regular PDBs.....</b>	<b>21</b>
Practices for Lesson 2: Overview .....	22
Practice 2-1: Exploring CDB Architecture and Structures .....	23
Practice 2-2: Creating a New CDB .....	34
Practice 2-3: Creating a New PDB .....	38
<b>Practices for Lesson 3: Application PDBs and Application Installation .....</b>	<b>45</b>
Practices for Lesson 3: Overview .....	46
Practice 3-1: Installing an Application in an Application Container.....	47
Practice 3-2: Upgrading an Application in an Application Container .....	56
Practice 3-3: Querying Data Across Application PDBs in CDB .....	62
<b>Practices for Lesson 4: PDB Creation.....</b>	<b>79</b>
Practices for Lesson 4: Overview .....	80
Practice 4-1: Cloning Remote PDBs in Hot Mode .....	81
Practice 4-2: Cloning an Application Container .....	88
Practice 4-3: Unplugging and Plugging Application Containers .....	95
Practice 4-4: Converting a Regular PDB to an Application PDB .....	102
Practice 4-5: Relocating PDBs.....	109
Practice 4-6: Querying Data Across CDBs by Using Proxy PDBs .....	117
Practice 4-7: Dropping Unnecessary PDBs.....	127
<b>Practices for Lesson 5: CDB and PDB Management.....</b>	<b>131</b>
Practices for Lesson 5: Overview .....	132
Practice 5-1: Starting up and Shutting down a CDB.....	133
Practice 5-2: Opening and Closing PDBs .....	138
Practice 5-3: Renaming a PDB .....	144
Practice 5-4: Setting Parameter Values for PDBs .....	147
Practice 5-5: Renaming PDB Services .....	151

<b>Practices for Lesson 6: Storage.....</b>	<b>157</b>
Practices for Lesson 6: Overview .....	158
Practice 6-1: Managing Permanent and Temporary Tablespaces .....	159
Practice 6-2: Managing UNDO Tablespaces .....	166
<b>Practices for Lesson 7: Security.....</b>	<b>169</b>
Practices for Lesson 7: Overview .....	170
Practice 7-1: Managing Common and Local Users, Privileges, and Roles .....	171
Practice 7-2: Managing Common and Local Objects in Application Containers .....	216
Practice 7-3: Enabling Common Users to View Information About PDB Objects .....	226
Practice 7-4: Applying Recorded Statements in Application PDBs .....	230
Practice 7-5: Managing PDB Lockdown Profiles .....	237
Practice 7-6: Auditing Operations in PDBs .....	256
Practice 7-7: Protecting Application Common Objects with Database Vault Common Realms.....	269
Practice 7-8: Managing PDB Keystores .....	282
Practice 7-9: Unplugging and Plugging Encrypted PDBs .....	283
<b>Practices for Lesson 8: Backup and Duplicate .....</b>	<b>289</b>
Practices for Lesson 8: Overview .....	290
Practice 8-1: RMAN Whole CDB Backup .....	291
Practice 8-2: RMAN PDB Backup.....	296
Practice 8-3: Duplicating a PDB into an Existing CDB .....	300
Practice 8-4: Duplicating an On-Premises CDB for Cloud .....	305
<b>Practices for Lesson 9: Recovery and Flashback.....</b>	<b>323</b>
Practices for Lesson 9: Overview .....	324
Practice 9-1: RMAN Recovery from SYSTEM PDB Datafile Loss .....	325
Practice 9-2: RMAN Recovery from Nonessential PDB Datafile Loss .....	341
Practice 9-3: PDB PITR .....	344
Practice 9-4: Recovering a Plugged Non-CDB by Using Preplugin Backups .....	349
Practice 9-5: Recovering a Plugged PDB by Using Preplugin Backups .....	350
Practice 9-6: Flashing Back an Application Upgrade by Using Restore Points.....	364
Practice 9-7: Managing and Using PDB Snapshots .....	371
Practice 9-8: Switching Over Refreshable Cloned PDBs .....	372
<b>Practices for Lesson 10: Performance .....</b>	<b>373</b>
Practices for Lesson 10: Overview .....	374
Practice 10-1: Monitoring Performance at CDB and PDB Levels.....	375
Practice 10-2: Getting Performance ADDM Recommendations at CDB and PDB Levels .....	380
Practice 10-3: Monitoring and Tuning SQL Executions at PDB Level.....	392
Practice 10-4: Configuring CDB Fleet.....	406
<b>Practices for Lesson 11: Resources Allocation .....</b>	<b>415</b>
Practices for Lesson 11: Overview .....	416

Practice 11-1: Managing PDB Performance Profiles.....	417
Practice 11-2: Managing Resource Allocation Between PDBs .....	426
Practice 11-3: Avoiding Excessive Session PGA Memory Usage in PDBs .....	432
<b>Practices for Lesson 12: Data Movement .....</b>	<b>437</b>
Practices for Lesson 12: Overview .....	438
Practice 12-1: Performing a Full Transportable Export/Import from a 12c Non-CDB into an 18c PDB .....	439
Practice 12-2: Performing a Full Transportable Export/Import from a 12c PDB into an 18c PDB .	443
<b>Practices for Lesson 13: Upgrade Methods .....</b>	<b>447</b>
Practices for Lesson 13: Overview .....	448
Practice 13-1: Upgrading a 12.2 Regular PDB to an 18c Application PDB .....	449
Practice 13-2: Plugging Remote PDBs Through XTTs .....	476
Practice 13-3: Upgrading a 12.2 CDB to an 18c CDB .....	487

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.

**Course Practice  
Environment: Security  
Credentials**

## 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
VM1 (for on-premises DBs)		
ORCL and PDB1	Any user	Welcome_1
Other PDBs created in ORCL	Any user	Welcome_1
Other CDBs created and PDBs	Any user	Welcome_1
VM2 (for Enterprise Manager Control)		
cdbem and PDBEM	sysman	Oracle123

## **Practices for Lesson 1: CDB Basics**

## Practices for Lesson 1: Overview

### Practices Overview

All practices are independent from one lesson to another.

1. Your system currently has two VMs:

- VM1 dedicated to RDBMS: Both Oracle Database 12.2.0.1 and 18c are installed, with three pre-created databases.
  - The `NONCDB` database is the 12.2.0.1 database (a non-CDB).
  - The `cdb12` database is the 12.2.0.1 database (a CDB) with one PDB, `PDB12`.
  - The `ORCL` database is the 18c CDB with one PDB, `PDB1`.
  - Net service names for any of the future CDBs and PDBs that you will create through the practices are already logged in the  
`/home/oracle/labs/admin/tnsnames_host01.ora` file.  
`/u01/app/oracle/product/18.1.0/dbhome_1/network/admin/tnsnames.ora` file
- VM2 dedicated to EM CC on which you will act as an Enterprise Manager administrator:
  - 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).

To clean up your CDB and PDBs at the beginning of the practices of each lesson, on VM1, you execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script. The shell script drops all PDBs that may have been created by any of the practices and finally recreates `PDB1`.

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

In case you need to recreate the **ORCL CDB and its PDB1 PDB**, use the `/home/oracle/labs/admin/recreate_ORCL.sh` shell script.

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

In case you need to **create** or **recreate the CDB18 CDB without its PDB18 PDB**, use the /home/oracle/labs/creation/create\_empty\_CDB18.sh shell script.

```
$ $HOME/labs/creation/create_empty_CDB18.sh  
...  
$
```

In case you need to **create** or **recreate the CDB18 CDB and its PDB18 PDB**, use the /home/oracle/labs/creation/create\_CDB18.sh shell script.

```
$ $HOME/labs/creation/create_CDB18.sh  
...  
$
```

## Practice 1-1: Discovering Practices Environment

---

### Tasks

- Discover the system environment for the practices of this course.

The configuration for the pre-created databases matches the preconfigured directories on an Oracle Cloud compute node associated with the pre-created database of a database deployment (or Database Cloud Service instance), whichever Oracle Database release the database is pre-created in.

- Non-CDB and CDB datafiles in /u02/app/oracle/oradata/
  - CDB root datafiles in /u02/app/oracle/oradata/<dbname>
  - PDB datafiles in /u02/app/oracle/oradata/<dbname>/PDB1 (the preconfigured PDB on Oracle Cloud is PDB1 and therefore the subdirectory is PDB1)
  - Controlfiles in  
/u02/app/oracle/oradata/<dbname> and /u03/app/oracle/fast\_recovery\_area/<dbname>
  - All redo log files in /u04/app/oracle/redo/ORCL (the preconfigured 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/<dbname>
  - Password and init files in \$ORACLE\_HOME/dbs
  - Diagnostics files in  
/u01/app/oracle/diag/rdbms/<dbname\_lower\_case>/<dbname>/...
  - TDE wallet in /u01/app/oracle/admin/<dbname>/tde\_wallet
  - Net files in \$ORACLE\_HOME/network/admin
- Copy the /home/oracle/labs/admin/tnsnames\_host01.ora file to /u01/app/oracle/product/18.1.0/dbhome\_1/network/admin/tnsnames.ora file.

If later in the practices, you create other PDBs than those defined in the /home/oracle/labs/admin/tnsnames\_host01.ora file, you can still update and copy the /home/oracle/labs/admin/tnsnames\_host01.ora file to the /u01/app/oracle/product/18.1.0/dbhome\_1/network/admin/tnsnames.ora file.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL
The Oracle base has been set to /u01/app/oracle
$
$ cp /home/oracle/labs/admin/tnsnames_host01.ora
/u01/app/oracle/product/18.1.0/dbhome_1/network/admin/tnsnames.or
a
$ lsnrctl reload
$
```

## Practice 1-2: Adding a CDB as a New Target in Enterprise Manager Cloud Control

### Overview

In this practice, you will access Oracle Enterprise Manager Cloud Control 13c as the `sysman` user and select **Summary** as your home page. Then you will add the `cdb12` CDB as a monitored target.

### Tasks

1. On VM2, click the Firefox icon to open a browser to access the Enterprise Manager Cloud Control console.
2. Enter the URL for Cloud Control: <https://localhost:7802/em>.

If the OMS is not started, start it as follows:

```
$ export OMS_HOME=/u01/app/oracle/product/middleware/oms
$ $OMS_HOME/bin/emctl stop oms -all -force
$ $OMS_HOME/bin/emctl start oms
...
Starting Oracle Management Server...
Starting WebTier...
WebTier Successfully Started
Oracle Management Server Successfully Started
Oracle Management Server is Up
WARNING: Limit of open file descriptors is found to be 1024.
The OMS has been started but it may run out of descriptors under
heavy usage.
For proper functioning of OMS, please set "ulimit -n" to be at
least 4096.
$
```

3. If you receive a Secure Connection Failed message, you need to add a security 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. The Enterprise Manager Cloud Control console appears.
5. Enter `sysman` in the User Name field and the password in the Password field. Then click Login.
6. The first time a new user logs in to Enterprise Manager, the user is prompted to accept the license agreement. You have to accept only once. During subsequent logins, the license agreement page will not appear.

7. In the Enterprise Summary page, add the `cdb12` Database Instance as a new target in Enterprise Manager Cloud Control.
  - a. On the top right corner of Enterprise Manager, select Setup > Add Target > Add Targets Manually.
  - b. Under Add Targets Manually, select Add Non-Host Targets Using Guided Process and click Add Using Guided Process. Then under Guided Discovery, select Oracle Database, Listener and Automatic Storage Management. Click the Add button.
  - c. In Specify Host or Cluster, click the magnifying glass to find the host (`host01.us.oracle.com`) where `cdb12` runs. Select your host, click Select, and then click Next.
  - d. In the Databases list, uncheck all databases except `cdb12`.
    - 1) Unlock the `DBSNMP` user. This user is the monitoring user used to test the connection once the target is being added. On VM1, open a terminal window.

```
$ . oraenv
ORACLE_SID = [oracle] ? cdb12
The Oracle base has been set to /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER USER dbsnmp IDENTIFIED BY password ACCOUNT UNLOCK
      CONTAINER=ALL;
2
User altered.

SQL> EXIT
$
```

- 2) Back to the Enterprise Manager page on VM2, enter the password for Monitor Password.
  - e. Click the Test Connection button. Click OK. Click Next.
  - f. Click Save to complete the operation and finally click Close.

## Practice 1-3: Checking Named Credentials

---

### Overview

In this practice, because you are going to work on ORCL in most of the practices, you verify the existence of the CREDORCL credential used for any connection as SYS user sharable in the multitenant container database instance ORCL.

### Tasks

1. On VM2, navigate to Setup > Security > Named Credentials. You can see the CREDORCL database instance credential with the following values:

Field	Choice or Value
<b>General Properties</b>	
Credential Name	CREDORCL
Credential Owner	SYSMAN
Authenticating Target Type	Database Instance
Credential Type	Database Credentials
Target Name	ORCL
Target Username	SYS

2. Test if the named credential works when you connect to the ORCL target. Click Targets and then select Databases.
3. Click ORCL.
4. Click Administration, then Storage, and then Tablespaces. The named credential CREDORCL is displayed.
5. Click Login if you accept this named credential to log in to the ORCL multitenant container database; otherwise choose New to define new login username and password.
6. When you click Logout, choose “Logout of Enterprise Manager and all targets” and click Logout.
7. Still on VM2, open a terminal window and verify that the monitoring information stored in the Management Repository is stored in the pdbem PDB.

```
$ . oraenv
ORACLE_SID = [oracle] ? cdbem
The Oracle base has been set to /u01/app/oracle
$ sqlplus sysman@pdbem

Enter password: password

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
-----
CREDORCL          SYSMAN             ORCL
SQL> EXIT
$
```

## Practice 1-4: Using Enterprise Manager Express

---

### Overview

In this practice, you cover the following things:

- Determine the HTTPS port number on which ORCL listens to Oracle Enterprise Manager Database Express (EM Express).
- Enable the global port for your ORCL CDB so that you can start EM Express for a PDB.
- Start EM Express for the ORCL CDB.
- Start EM Express for PDB1.
- Explore the differences in the EM Express interface when connected to a CDB versus a PDB.

### Tasks

1. You are logged in to VM1 as the oracle user. Open a terminal window and use the oraenv script to set the ORACLE\_SID value to ORCL.

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

2. Determine the port number on which your CDB listens to EM Express.

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

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

- b. Execute the DBMS\_XDB\_CONFIG.GETHTTPSPORT procedure to view the HTTPS port number configured on the CDB by DBCA for EM Express. The query should return port 5500. A value of zero indicates that an EM Express port is not configured, and you'll need to manually configure one. All communication between your CDB and EM Express is encrypted, by default.

**Note:** You can execute the DBMS\_XDB\_CONFIG.GETHTTPPORT procedure to check for unencrypted protocols being used. This is good for security audits and will return a port number of zero.

```
SQL> SELECT dbms_xdb_config.gethttpsport() FROM dual;

DBMS_XDB_CONFIG.GETHTTPSPORT()
-----
5500
SQL>
```

3. Enable the global port for the CDB.

```
SQL> EXEC dbms_xdb_config.SetGlobalPortEnabled(TRUE)  
PL/SQL procedure successfully completed.  
SQL>
```

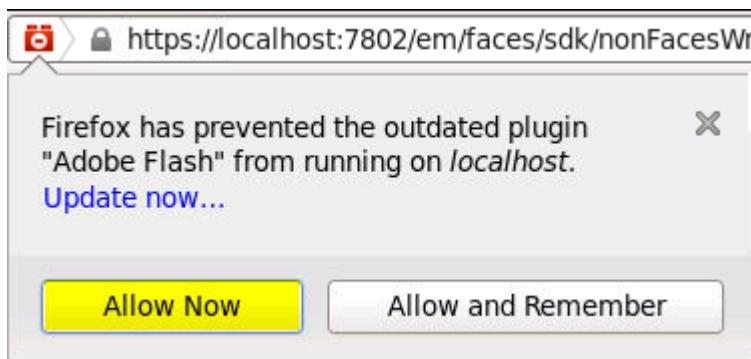
4. Start EM Express for the ORCL CDB. To do this, open a Firefox web browser and enter the URL <https://localhost:5500/em>.

The first time you enter the URL for EM Express in your web browser, your browser may display warning messages. You'll need to enter a security exception for the EM Express URL in your web browser.

5. If needed, enter a security exception for the EM Express URL. VM1 is set up so that you don't have to enter a security exception.
  - a. On the This Connection is Untrusted page displayed, expand **I Understand the Risks** and click **Add Exception**.
  - b. In the Add Security Exception dialog box, retain the default settings and click **Confirm Security Exception**.
  - c. If you do see the following plugin message,



allow the Adobe Flash plugin. To the left of the URL, left-click the icon, select **Allow and Remember**, and click **Allow and Remember**.



6. On the Login page for EM Express, enter the user name **sys** and the password as specified by your instructor. Leave the Container Name box empty. Select the **as sysdba** check box. Click **Login**. Remember, to manage a CDB root with EM Express, you must log in as a user with the **SYSDBA** privilege.
7. View the EM Express Home page for the CDB. The Status column provides information about:
  - Up time duration
  - Instance type (single instance named **ORCL**; a CDB with 1 PDB)
  - Database version (18.1.0.0.0)
  - Database name (**ORCL**)
  - Instance name (**ORCL**)
  - Platform name (Linux x86 64-bit)
  - Host name (**host01**)
  - Thread number (1)
  - Whether the archiver process is started or stopped (currently started)

Also notice that you can view configuration, storage, security, and performance information.

8. Click the tabs and menus to review the interface.
9. In the upper right corner, click **Log Out**.
10. Start EM Express for **PDB1**. To do this, on the Login page for EM Express, enter the user name **sys** and the password as specified by your instructor. Enter **PDB1** as the container name. Select the **as sysdba** check box. Click **Login**. If an error message tells you that you don't have sufficient privileges, switch to the terminal session left open in step 3 and open **PDB1**.

```
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN;  
  
Pluggable database altered.  
  
SQL> EXIT  
$
```

Click Login again.

11. View the EM Express Home page for **PDB1**. Notice that in the upper left corner, the Home page identifies the container as **PDB1**. In the Status column, you have a field named Container Name, which also indicates the container to which you are currently connected is **PDB1**.
12. Browse the Configuration, Storage, Security, and Performance menus.
13. Click **Log Out** and close the browser window.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.

## **Practices for Lesson 2: CDB and Regular PDBs**

## Practices for Lesson 2: Overview

---

### Practices Overview

In this practice, you will explore and get familiar with the architecture and structures of CDBs and PDBs and create a CDB and PDBs by using different tools like SQL\*Plus, SQL Developer, and Database Configuration Assistant (DBCA).

## Practice 2-1: Exploring CDB Architecture and Structures

### Overview

In this practice, you will explore the architecture and structures of ORCL and its pluggable database (PDB).

### Tasks

1. Before starting the practice, execute the \$HOME/labs/creation/glogin\_2.sh shell script. It sets formatting for all columns selected in queries.

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

2. Explore the ORCL instance, the background processes, and the multitenant container database.
  - a. Use the pgrep Unix command.

```
$ pgrep -lf ORCL  
4715 ora_pmon_ORCL  
4722 ora_clmn_ORCL  
4731 ora_psp0_ORCL  
4780 ora_vktm_ORCL  
4800 ora_gen0_ORCL  
4806 ora_mman_ORCL  
4814 ora_gen1_ORCL  
4824 ora_diag_ORCL  
4828 ora_ofsd_ORCL  
4839 ora_dbrm_ORCL  
4846 ora_vkrm_ORCL  
4850 ora_svcb_ORCL  
4857 ora_pman_ORCL  
4865 ora_dia0_ORCL  
4877 ora_dbw0_ORCL  
4895 ora_lgwr_ORCL  
4909 ora_ckpt_ORCL  
4919 ora_lg00_ORCL  
4925 ora_smon_ORCL  
4927 ora_lg01_ORCL  
4929 ora_smco_ORCL  
4932 ora_reco_ORCL  
4936 ora_lreg_ORCL  
4940 ora_pxmn_ORCL  
4946 ora_mmon_ORCL
```

```
4948 ora_mmn1_ORCL
4950 ora_d000_ORCL
4952 ora_s000_ORCL
4954 ora_tmon_ORCL
5128 ora_arc0_ORCL
5130 ora_tt00_ORCL
5132 ora_tt01_ORCL
5134 ora_arcl_ORCL
5136 ora_arc2_ORCL
5138 ora_arc3_ORCL
5140 ora_tt02_ORCL
5186 ora_aqpc_ORCL
5190 ora_p000_ORCL
5192 ora_p001_ORCL
5194 ora_p002_ORCL
5196 ora_p003_ORCL
5198 ora_p004_ORCL
5200 ora_p005_ORCL
5202 ora_p006_ORCL
5204 ora_p007_ORCL
5350 ora_cjq0_ORCL
5376 ora_qm02_ORCL
5382 ora_q002_ORCL
5384 ora_q003_ORCL
21903 ora_w005_ORCL
22118 ora_w003_ORCL
23332 ora_w002_ORCL
26975 ora_w006_ORCL
28207 ora_w000_ORCL
28213 ora_w007_ORCL
28352 ora_w004_ORCL
28384 ora_w001_ORCL
28939 ora_p008_ORCL
28941 ora_p009_ORCL
28943 ora_p00a_ORCL
28945 ora_p00b_ORCL
28947 ora_p00c_ORCL
28949 ora_p00d_ORCL
28951 ora_p00e_ORCL
28953 ora_p00f_ORCL
29271 ora_qm03_ORCL
$
```

- b. Connect to the multitenant container database ORCL.

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

SQL>
```

- c. Check if the database is a multitenant container database.

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

NAME          CDB CON_ID
-----
ORCL          YES     0

SQL>
```

- d. Check the instance name.

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

INSTANCE_NAME      STATUS      CON_ID
-----
ORCL              OPEN       0

SQL> EXIT
$
```

### 3. Explore the services.

- a. Check if the listener is started.

```
$ lsnrctl status
...
Services Summary...
...
Service "ORCL" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
Service "ORCLXDB" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
Service "pdb1" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
The command completed successfully
$
```

The listener is already started. If it were not started, you would use the following command to start the listener:

```
$ lsnrctl start
...
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc) (KEY=EXTPROC1521)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=yourserver) (PORT=1521))
)
The listener supports no services
The command completed successfully
$
```

- List the services automatically created for each container.

```
$ sqlplus / AS SYSDBA

SQL> SELECT name, con_id FROM v$services;

NAME          CON_ID
-----
ORCLXDB          1
ORCL            1
SYS$BACKGROUND    1
SYS$USERS         1
pdb1              3

SQL>
```

Notice that the PDB\$SEED service is not listed. The user should not connect to this service as there should not be any operation performed on this container, which is reserved as a template to create other PDBs.

- Display the pluggable databases.

- Use the new view V\$PDBS.

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

CON_ID NAME          OPEN_MODE
-----
2  PDB$SEED        READ ONLY
3  PDB1           READ WRITE

SQL>
```

Notice that the seed PDB is in READ ONLY open mode.

- b. Use the new commands `SHOW CON_NAME` and `CON_ID` to know which container you are connected to.

```
SQL> SHOW con_name

CON_NAME
-----
CDB$ROOT

SQL> SHOW con_id

CON_ID
-----
1

SQL>
```

You can also use the `SYS_CONTEXT` function to view the `CON_NAME` and `CON_ID` attributes of your session context.

```
SELECT sys_context('userenv', 'CON_NAME') FROM dual;
SELECT sys_context('userenv', 'CON_ID') FROM dual;
```

5. View some of the new family of views `CDB_xxx`:

```
SQL> SELECT pdb_id, pdb_name, dbid, con_id FROM cdb_pdbs;

PDB_ID PDB_NAME          DBID CON_ID
----- -----
  3 PDB1            3637697889      3
  2 PDB$SEED        814247809      2

SQL>
```

The `PDB_ID` number 2 is always assigned to the seed PDB because it is the second container to be created after the CDB root container (`CON_ID 1`).

6. Check all the files of the CDB.

- a. View the redo log files of the CDB.

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

GROUP# CON_ID MEMBER
----- -----
  4      0 /u04/app/oracle/redo/ORCL/redo01.log
  5      0 /u04/app/oracle/redo/ORCL/redo02.log
  6      0 /u04/app/oracle/redo/ORCL/redo03.log

SQL>
```

The CON\_ID value 0 refers to the CDB instance.

- View the control files of the CDB.

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

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

SQL>
```

- View all the datafiles of the CDB, including those of the CDB root and all PDBs.

- With CDB\_DATA\_FILES view:

```
SQL> SELECT file_name, tablespace_name, con_id CID
      FROM cdb_data_files ORDER BY con_id;

FILE_NAME                               TABLESPA CID
-----
/u02/app/oracle/oradata/ORCL/system01.dbf    SYSTEM     1
/u02/app/oracle/oradata/ORCL/users01.dbf      USERS     1
/u02/app/oracle/oradata/ORCL/undotbs01.dbf   UNDOTBS1  1
/u02/app/oracle/oradata/ORCL/sysaux01.dbf    SYSAUX    1
/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf SYSTEM    3
/u02/app/oracle/oradata/ORCL/PDB1/users01.dbf  USERS    3
/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf UNDOTBS1 3
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf SYSAUX    3

8 rows selected.

SQL>
```

There are the SYSTEM, SYSAUX, and UNDO datafiles and a tempfile for the CDB seed.

- Ensure that you are connected to the CDB root; then use the DBA\_DATA\_FILES view.

```
SQL> COL file_name FORMAT A42
SQL> SELECT file_name, tablespace_name, file_id
      FROM dba_data_files;

FILE_NAME                               TABLESPA FILE_ID
-----
/u02/app/oracle/oradata/ORCL/users01.dbf    USERS      7
/u02/app/oracle/oradata/ORCL/undotbs01.dbf  UNDOTBS1  4
/u02/app/oracle/oradata/ORCL/system01.dbf    SYSTEM     1
```

```
/u02/app/oracle/oradata/ORCL/sysaux01.dbf    SYSAUX
```

3

SQL>

Notice that only the root datafiles are listed.

- f. Now use the V\$TABLESPACE and V\$DATAFILE view.

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

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

11 rows selected.

SQL>

- g. List the tempfiles of the CDB.

```
SQL> COL file_name FORMAT A57
SQL> SELECT file_name, tablespace_name FROM cdb_temp_files;
```

FILE_NAME	TABLESPA
/u02/app/oracle/oradata/ORCL/temp01.dbf	TEMP
/u02/app/oracle/oradata/ORCL/PDB1/.../temp_....dbf	TEMP

SQL>

7. List all the users created.

- a. Verify that the SYSTEM user is created.

```
SQL> SELECT username, common, con_id FROM cdb_users
      WHERE username = 'SYSTEM';

USERNAME          COM CON_ID
-----
SYSTEM            YES     3
SYSTEM            YES     1

SQL>
```

Notice that the user SYSTEM exists in all containers as a common user.

- b. List all the common users of the CDB.

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

USERNAME
-----
ANONYMOUS
APPQOSSYS
AUDSYS
CTXSYS
DBSFWUSER
DBSNMP
...
SYS
SYS$UMF
SYSBACKUP
SYSDG
SYSKM
SYSRAC
SYSTEM
WMSYS
XDB
XS$NULL

35 rows selected.

SQL>
```

- c. List the local users in the CDB.

```
SQL> SELECT username, con_id FROM cdb_users
      WHERE common = 'NO';

USERNAME          CON_ID
-----
PDBADMIN           3

SQL>
```

Notice that there is no local user in the CDB root because it is impossible to create any local user in the CDB root.

8. List all the roles and privileges of the CDB.

- a. List all the roles of the CDB.

```
SQL> SELECT role, common, con_id FROM cdb_roles ORDER BY 3;

ROLE          COM CON_ID
-----
CONNECT        YES  1
RESOURCE       YES  1
DBA            YES  1
PDB_DBA        YES  1
AUDIT_ADMIN    YES  1
...
DV_DATAPUMP_NETWORK_LINK  YES  3
DV_POLICY_OWNER  YES  3
DV_REALM_RESOURCE YES  3

172 rows selected.

SQL>
```

Notice that there is no local role in the root container because it is impossible to create any local role in the root.

- b. Ensure that the privileges are neither common nor local by nature.

```
SQL> DESC sys.system_privilege_map
Name          Null?    Type
-----
PRIVILEGE     NOT NULL NUMBER
NAME          NOT NULL VARCHAR2(40)
PROPERTY      NOT NULL NUMBER

SQL> DESC sys.table_privilege_map
```

Name	Null?	Type
PRIVILEGE	NOT NULL	NUMBER
NAME	NOT NULL	VARCHAR2 (40)
SQL>		

Notice that there is no COMMON column.

- c. Verify that the privilege when granted becomes a common or local privilege.

SQL> DESC CDB_SYS_PRIVS		
Name	Null?	Type
GRANTEE		VARCHAR2 (128)
PRIVILEGE		VARCHAR2 (40)
ADMIN_OPTION		VARCHAR2 (3)
<b>COMMON</b>		VARCHAR2 (3)
INHERITED		VARCHAR2 (3)
CON_ID		NUMBER
SQL> DESC CDB_TAB_PRIVS		
Name	Null?	Type
GRANTEE		VARCHAR2 (128)
OWNER		VARCHAR2 (128)
TABLE_NAME		VARCHAR2 (128)
GRANTOR		VARCHAR2 (128)
PRIVILEGE		VARCHAR2 (40)
GRANTABLE		VARCHAR2 (3)
HIERARCHY		VARCHAR2 (3)
<b>COMMON</b>		VARCHAR2 (3)
TYPE		VARCHAR2 (24)
INHERITED		VARCHAR2 (3)
CON_ID		NUMBER
SQL>		

There is a COMMON column.

- d. Notice that the role, though common or local depending on how the role was created, is also, like privileges, granted either commonly or locally.

SQL> SELECT grantee, granted_role, common, con_id FROM cdb_role_privs WHERE grantee='SYSTEM' ;			
GRANTEE	GRANTED_ROLE	COM	CON_ID
SYSTEM	DBA	Y	1

-----			
SYSTEM	DBA	YES	3
SYSTEM	AQ_ADMINISTRATOR_ROLE	YES	3
SYSTEM	DBA	YES	1
SYSTEM	AQ_ADMINISTRATOR_ROLE	YES	1
SQL>	<b>EXIT</b>		
	\$		

## Practice 2-2: Creating a New CDB

---

### Overview

In this practice, you will create a new CDB, `CDB18`, with DBCA in silent mode. The CDB dedicated for test purposes will have the following characteristics:

- The users `SYS` and `SYSTEM` will have the same password as the one used for the same users in `ORCL`.
- You do not use explicit data and redo log file names. Use Oracle Managed Files (OMF) and set the location of these files in the `/u02/app/oracle/oradata/CDB18` directory.
- The CDB root will contain:
  - A default temporary tablespace `TEMP`
  - A default permanent tablespace `USERS`
  - An UNDO tablespace
- The port used for EM Express is `5502`.
- The CDB is created with no PDBs, except the CDB seed.

### Tasks

1. Verify that `CDB18` is not already recorded in `/etc/oratab`. If this is the case, remove the entry.

```
$ cat /etc/oratab
#...
CDB18:/u01/app/oracle/product/18.1.0/dbhome_1:Y
NONCDB:/u01/app/oracle/product/12.2.0/dbhome_1:Y
cdb12:/u01/app/oracle/product/12.2.0/dbhome_1:Y
ORCL:/u01/app/oracle/product/18.1.0/dbhome_1:Y
$
```

2. Create the CDB by using DBCA in silent mode. Change the `password` text for each occurrence of the word in the command before executing the command.

```
$ $ORACLE_HOME/bin/dbca -silent -createDatabase -templateName
General_Purpose.dba -gdbname CDB18 -sid CDB18 -
createAsContainerDatabase true -numberOfPDBs 0 -
useLocalUndoForPDBs true -responseFile NO_VALUE -totalMemory
1800 -sysPassword password -systemPassword password -
pdbAdminPassword password -emConfiguration DBEXPRESS -
dbsnmpPassword password -emExpressPort 5502 -enableArchive true
-recoveryAreaDestination /u03/app/oracle/fast_recovery_area -
recoveryAreaSize 15000 -datafileDestination
/u02/app/oracle/oradata
...
Copying database files
1% complete
2% complete
```

```
18% complete
33% complete
Creating and starting Oracle instance
35% complete
40% complete
41% complete
42% complete
46% complete
51% complete
52% complete
53% complete
55% complete
Completing Database Creation
56% complete
58% complete
59% complete
62% complete
63% complete
66% complete
Executing Post Configuration Actions
100% complete
Database creation complete. For details check the logfiles at:
/u01/app/oracle/cfgtoollogs/dbca/CDB18.
Database Information:
Global Database Name:CDB18
System Identifier(SID):CDB18
Look at the log file
"/u01/app/oracle/cfgtoollogs/dbca/CDB18/CDB180.log" for further
details.
$
```

**Q/ Is a new entry added in /etc/oratab?**

```
$ more /etc/oratab
NONCDB:/u01/app/oracle/product/12.2.0/dbhome_1:Y
cdb12:/u01/app/oracle/product/12.2.0/dbhome_1:Y
ORCL:/u01/app/oracle/product/18.1.0/dbhome_1:Y
CDB18:/u01/app/oracle/product/18.1.0/dbhome_1:N
$
```

**A/ Yes.**

3. Verify that the characteristics of the database are satisfied.
- Check that the CDB is a CDB.

```
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> SELECT cdb FROM v$database;

CDB
---
YES

SQL>
```

- Check that the datafiles are located in the /u02/app/oracle/oradata/CDB18 directory.

```
SQL> COL name FORMAT A58
SQL> SELECT name FROM v$datafile ORDER BY 1;

NAME
-----
/u02/app/oracle/oradata/CDB18/pdbseed/sysaux01.dbf
/u02/app/oracle/oradata/CDB18/pdbseed/system01.dbf
/u02/app/oracle/oradata/CDB18/pdbseed/undotbs01.dbf
/u02/app/oracle/oradata/CDB18/sysaux01.dbf
/u02/app/oracle/oradata/CDB18/system01.dbf
/u02/app/oracle/oradata/CDB18/undotbs01.dbf
/u02/app/oracle/oradata/CDB18/users01.dbf

7 rows selected.

SQL>
```

- Verify that the permanent tablespaces USERS and UNDO tablespaces are created and that there is the TEMP temporary tablespace.

```
SQL> SELECT tablespace_name, contents FROM dba_tablespaces;

TABLESPA CONTENTS
-----
SYSTEM      PERMANENT
SYSAUX      PERMANENT
UNDOTBS1    UNDO
```

```
TEMP      TEMPORARY  
USERS     PERMANENT  
  
SQL>
```

- d. Verify the port set for EM Express.

```
SQL> SELECT dbms_xdb_config.gethttpsport() FROM dual;  
  
DBMS_XDB_CONFIG.GETHTTPSPORT()  
-----  
5502  
  
SQL>
```

- e. Set the LOCAL\_LISTENER to NULL.

```
SQL> ALTER SYSTEM SET local_listener = '' SCOPE = BOTH;  
  
System altered.  
  
SQL> EXIT  
$
```

## Practice 2-3: Creating a New PDB

---

### Overview

In this practice, you will create a pluggable database (PDB) from the CDB seed in CDB18 by using SQL\*Plus.

The PDB is named PDB18. The PDB will have the following characteristics:

- The users SYS and SYSTEM will have the same password as the one used for the same users in ORCL.
- The DBA user for the PDB is pdb18\_admin. The pdb18\_admin user will have the same password as the one used for SYS and SYSTEM.
- Set the location of the PDB datafiles in the /u02/app/oracle/oradata/CDB18/PDB18 directory.

In case you need to **create or re-create the CDB18 CDB without its PDB18 PDB**, use the /home/oracle/labs/creation/create\_empty\_CDB18.sh shell script.

```
$ $HOME/labs/creation/create_empty_CDB18.sh
...
$
```

### Tasks

1. Create a directory for the PDB18 datafiles.

```
$ mkdir /u02/app/oracle/oradata/CDB18/PDB18
$
```

2. Run SQL\*Plus and connect to the CDB root with a user with the CREATE PLUGGABLE DATABASE privilege.

```
$ sqlplus / AS SYSDBA

SQL> CREATE PLUGGABLE DATABASE pdb18
      ADMIN USER pdb18_admin IDENTIFIED BY password
      ROLES=(CONNECT)
      CREATE_FILE_DEST='/u02/app/oracle/oradata/CDB18/PDB18';
      2      3      4
Pluggable database created.

SQL>
```

3. Observe the status of the PDB.

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

PDB_ID PDB_NAME STATUS
-----
```

```
2 PDB$SEED NORMAL  
3 PDB18      NEW
```

```
SQL>
```

4. Open the PDB.

```
SQL> ALTER PLUGGABLE DATABASE pdb18 OPEN;
```

```
Pluggable database altered.
```

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

```
PDB_ID PDB_NAME STATUS
```

```
-----  
2 PDB$SEED NORMAL  
3 PDB18      NORMAL
```

```
SQL>
```

5. Connect to the PDB (the net service name has been pre-created for you in the tnsnames.ora) and check that the datafiles are in the expected location.

```
SQL> CONNECT system@PDB18
```

```
Enter password: password
```

```
Connected.
```

```
SQL>
```

```
SQL> COL name FORMAT A78
```

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

```
NAME
```

```
-----  
/u02/app/oracle/oradata/CDB18/PDB18/CDB18/4559EDF875437DA7E053C5  
10ED0A0616/datafile/o1_mf_system_d6wlq27z_.dbf
```

```
/u02/app/oracle/oradata/CDB18/PDB18/CDB18/4559EDF875437DA7E053C5  
10ED0A0616/datafile/o1_mf_sysaux_d6wlq290_.dbf
```

```
/u02/app/oracle/oradata/CDB18/PDB18/CDB18/4559EDF875437DA7E053C5  
10ED0A0616/datafile/o1_mf_undotbs1_d6wlq291_.dbf
```

```
SQL>
```

**Q/ Is there a *USERS* tablespace as in the CDB root?**

**A/ No. You'll have to create one if it is needed. You'll cover this in later practices.**

Q2/ Is there an UNDO tablespace as in the CDB root?

A2/ Yes. By default, an UNDO tablespace is created in each container. Therefore, the UNDO tablespace of the CDB root is not shared by all PDBs. This is the local UNDO mode.

Q3/ Can you connect under the PDBA?

```
SQL> CONNECT pdb18_admin@PDB18
Enter password: password
Connected.
SQL>
```

Q3/ Which roles are active in his session?

```
SQL> SELECT * FROM session_roles;

ROLE
-----
PDB_DBA
CONNECT

SQL>
```

A3/ The role CONNECT defined in the CREATE PLUGGABLE DATABASE statement through the ROLES clause and the default role granted to the ADMIN user defined in the same statement through the ADMIN USER clause.

Q4/ Were the COMMON users replicated in PDB18 during the CREATE PLUGGABLE DATABASE statement?

```
SQL> SELECT username FROM dba_users WHERE COMMON = 'YES';

SELECT username FROM dba_users WHERE COMMON = 'YES'
*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
```

Q4/ Why is the PDB\_DBA not able to view the dictionary view content?

```
SQL> SELECT * FROM session_privs;
```

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

SQL> SELECT * FROM user_tab_privs;

no rows selected

SQL>
```

**A4/ The role PDB\_DBA is granted neither the SELECT ANY DICTIONARY privilege nor the SELECT privilege on the view.**

```
SQL> CONNECT system@PDB18
Enter password: password

Connected.
SQL> SELECT username FROM dba_users WHERE COMMON = 'YES';

USERNAME
-----
SYS
SYSTEM
GSMCATUSER
XS$NULL
...
OLAPSYS
ORDDATA
XDB
WMSYS
ORDSYS

35 rows selected.

SQL>
```

**A4/ If you compare the result with the list of practice 2-1 step 7.b, it is exactly the same list.**

**Q5/ Were the COMMON roles replicated in pdb2 during the CREATE PLUGGABLE DATABASE statement?**

```
SQL> SELECT role, common, con_id FROM cdb_roles ORDER BY 3;

ROLE          COM CON_ID
-----
CONNECT        YES  3
RESOURCE       YES  3
DBA            YES  3
PDB_DBA        YES  3
...
DV_POLICY_OWNER YES  3
DV_REALM_RESOURCE YES 3

86 rows selected.

SQL>
```

**A5/ If you compare the result with the list of practice 2-1 step 8.a, it is not exactly the same result. It is the same list for each container. In practice 2-1 step 8.a, the list displayed all roles created in all containers (172 roles) except the CDB seed. There are currently three containers (CDB root, CDB seed, and PDB18). The CDB root holds 86 roles and PDB18 86 roles. Each new PDB holds the PDB\_DBA role granted to the ADMIN USER defined during the PDB creation.**

6. Check that the service is PDB18.

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

NAME
-----
pdb18

SQL> EXIT
$
```

7. Let's explore the ADR (Automatic Diagnostic Repository) for the CDB and PDBs.

- a. Check the alert\_CDB18.log file of the CDB instance.

```
$ cd $ORACLE_BASE/diag/rdbms/cdb18/CDB18/trace
$ tail -20 alert_CDB18.log
PDB18(3):Adding new file#11 to file$(old file#8).
fopr-1, newblk-12800, oldblk-12800
PDB18(3):Successfully created internal service PDB18 at open
2018-03-30T05:38:24.031806+00:00
*****
```

```

Post plug operations are now complete.
Pluggable database PDB18 with pdb id - 3 is now marked as NEW.
*****
PDB18(3):Database Characterset for PDB18 is AL32UTF8
2018-03-30T05:38:26.506113+00:00
PDB18(3):Resize operation completed for file# 10, old size
358400K, new size 368640K
PDB18(3):Opening pdb with no Resource Manager plan active
Pluggable database PDB18 opened read write
Completed: ALTER PLUGGABLE DATABASE pdb18 OPEN
2018-03-30T05:42:46.376633+00:00
PDB18(3):Resize operation completed for file# 10, old size
368640K, new size 378880K
2018-03-30T05:47:48.865267+00:00
Control autobackup written to DISK device

handle
'/u03/app/oracle/fast_recovery_area/CDB18/autobackup/2018_03_30/
o1_mf_s_972107266_fcvmw44w.bkp'
$
```

Observe that all operations related to PDBs in the CDB are reported in the single `alert_CDB18.log` file of the CDB instance.

- b. Find the CREATE PLUGGABLE DATABASE and ALTER PLUGGABLE DATABASE operations in the `alert.log`.

```

$ grep -i "create pluggable database" alert_CDB18.log
CREATE PLUGGABLE DATABASE PDB$SEED AS CLONE USING
'/u01/app/oracle/product/18.1.0/dbhome_1/assistants/dbca/templates/pdbseed.xml' source_file_name_convert =
('/ade/b/3802259419/oracle/oradata/SEEDDATA/pdbseed/system01.dbf',
', '/u02/app/oracle/oradata/CDB18/pdbseed/system01.dbf',
Completed: CREATE PLUGGABLE DATABASE PDB$SEED AS CLONE USING
'/u01/app/oracle/product/18.1.0/dbhome_1/assistants/dbca/templates/pdbseed.xml' source_file_name_convert =
('/ade/b/3802259419/oracle/oradata/SEEDDATA/pdbseed/system01.dbf',
', '/u02/app/oracle/oradata/CDB18/pdbseed/system01.dbf',
CREATE PLUGGABLE DATABASE pdb18
$
```

```

$ grep -i "alter pluggable database" alert_CDB18.log
...
alter pluggable database PDB$SEED open restricted
Completed: alter pluggable database PDB$SEED open restricted
alter pluggable database all close immediate
Completed: alter pluggable database all close immediate
```

```
alter pluggable database all close immediate
Completed: alter pluggable database all close immediate
alter pluggable database PDB$SEED CLOSE IMMEDIATE
Completed: alter pluggable database PDB$SEED CLOSE IMMEDIATE
alter pluggable database PDB$SEED OPEN RESTRICTED
Completed: alter pluggable database PDB$SEED OPEN RESTRICTED
PDB$SEED(2):alter pluggable database PDB$SEED set
time_zone='+00:00'
PDB$SEED(2):Completed: alter pluggable database PDB$SEED set
time_zone='+00:00'
alter pluggable database PDB$SEED CLOSE IMMEDIATE
Completed: alter pluggable database PDB$SEED CLOSE IMMEDIATE
alter pluggable database PDB$SEED OPEN restricted
Completed: alter pluggable database PDB$SEED OPEN restricted
alter pluggable database all close immediate
Completed: alter pluggable database all close immediate
alter pluggable database all close immediate
Completed: alter pluggable database all close immediate
alter pluggable database pdb$seed close immediate instances=all
Completed: alter pluggable database pdb$seed close immediate
instances=all
alter pluggable database pdb$seed OPEN READ WRITE
Completed: alter pluggable database pdb$seed OPEN READ WRITE
alter pluggable database pdb$seed close immediate instances=all
Completed: alter pluggable database pdb$seed close immediate
instances=all
alter pluggable database pdb$seed OPEN READ ONLY instances=all
Completed: alter pluggable database pdb$seed OPEN READ ONLY
instances=all
alter pluggable database all close immediate
Completed: alter pluggable database all close immediate
alter pluggable database all close immediate
Completed: alter pluggable database all close immediate
ALTER PLUGGABLE DATABASE pdb18 OPEN
Completed: ALTER PLUGGABLE DATABASE pdb18 OPEN
$
```

The ADR has the same directory structure as a non-CDB instance.

## **Practices for Lesson 3: Application PDBs and Application Installation**

## Practices for Lesson 3: Overview

---

### Practices Overview

In these practices, in ORCL CDB, 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.

Finally, you query application data from application PDBs within the same CDB by using container maps.

## Practice 3-1: 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.

In case you need to **re-create the ORCL CDB and its PDB1 PDB**, use the `/home/oracle/labs/admin/recreate_ORCL.sh` shell script.

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

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. Then execute the `$HOME/labs/app/glogin_3.sh` shell script. It sets formatting for all columns selected in queries.

```
$ $HOME/labs/app/glogin_3.sh
$
```

3. 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 = [CDB18] ? 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 password
ROLES=(CONNECT)
CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/toys_root';
2   3   4   5
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$containers;
2   3   4   5
NAME          CON_ID APP_ROO APP_See APP_PDB APP_ROOT_CONID
-----  -----  -----  -----  -----
CDB$ROOT        1 NO      NO      NO
PDB$SEED        2 NO      NO      NO
TOYS_ROOT       3 YES     NO      NO
PDB1            5 NO      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: 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 password 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>
```

*Q/ Why does it fail?*

**A/ 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: password
Connected.
SQL> CREATE PLUGGABLE DATABASE AS SEED ADMIN USER admin
      IDENTIFIED BY password 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$containers;
2   3   4   5
NAME          CON_ID APP_ROO APP_See APP_PDB APP_ROOT_CONID
-----  -----  -----  -----  -----
TOYS_ROOT      3 YES    NO     NO
TOYS_ROOT$SEED 4 NO     YES    YES
                           3
SQL>
```

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

**A3/ The application seed's name is <application\_root>\$SEED.**

- e. 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
-----  -----  -----
3 TOYS_ROOT                         READ WRITE NO
4 TOYS_ROOT$SEED                     READ WRITE NO
SQL>
```

4. 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 password
      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 password
      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: password
```

```
ERROR:  
ORA-01017: invalid username; 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 / AS SYSDBA  
Connected.  
SQL> ALTER SESSION SET CONTAINER=toys_root$seed;  
  
Session 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>
```

```
SQL> CONNECT toys_owner@robots  
Enter password: password  
ERROR:  
ORA-01017: invalid username; 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: 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 password
      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: 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: 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: 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 3-2: Upgrading an Application in an Application Container

### 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

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

```
$ $HOME/labs/app/setup_apps.sh
...
$
```
- 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.
  - 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$SEED           READ ONLY  NO
3 PDB1                READ WRITE NO
4 TOYS_ROOT          MOUNTED
5 TOYS_ROOT$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';

```

```

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: 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        1.0          UPGRADING

```

```

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>

```

*Q/ What do you observe if you connect to the CDB root?*

```

SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHOW PDBS

  CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
  2 PDB$SEED              READ ONLY NO
  3 PDB1                  READ WRITE NO
 11 F399688568_3_1        READ WRITE NO
  4 TOYS_ROOT             READ WRITE NO
  5 TOYS_ROOT$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

SQL>

```

*A1/ There is a new PDB, F399688568\_3\_1 (your application root clone will probably have another number). 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: 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: 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: 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: 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 3-3: Querying Data Across Application PDBs in CDB

### Overview

In this practice, you will use the dynamic container maps. In Oracle Database 12c, a container map enables a session connected to an application root to issue SQL statements that are routed to the appropriate PDB, depending on the value of a predicate used in the SQL statement. Container maps enable the partitioning of data at the application PDB level when the data is not physically partitioned at the table level. In Oracle Database 18c, a container map can be dynamically updated when new PDBs are created or dropped.

### Tasks

- 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/setup_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 common `scott.emp` and `scott.dept` tables.

```
$ $HOME/labs/app/setup_hr_app.sh
...
$
```

- Use the `$HOME/labs/app/script_insert.sql` SQL script to insert rows into the common `scott.emp` and `scott.dept` tables.
  - The script inserts different rows in the `scott.emp` and `scott.dept` tables in each application PDB.

```
$ sqlplus / AS SYSDBA
SQL> @"$HOME/labs/app/script_insert.sql
...
SQL>
```

- Query data from the tables.

```
SQL> CONNECT scott@hr_root
Enter password: password
Connected.
SQL> SELECT * FROM scott.emp;
no rows selected
```

```
SQL> CONNECT scott@sales
Enter password: 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: 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>
```

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

**A/ Use the CONTAINERS clause.**

```
SQL> CONNECT scott@hr_root
Enter password: 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: 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: password
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;

Pluggable database altered.

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

Pluggable database altered.

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

Pluggable database altered.

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

Pluggable database altered.

SQL> CONNECT sys@research AS SYSDBA
```

```
Enter password: password
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;

Pluggable database altered.

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

Pluggable database altered.

SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: 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: 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	Cost	(%CPU)	Time	Pstart	Pstop	Name	TQ	Rows	IN-OUT	Bytes	PQ
Distrib												
8 (100)	0   SELECT STATEMENT	1		00:00:01					1800		54000	
	1   PX COORDINATOR											
8 (100)	2   PX SEND QC (RANDOM)	1		00:00:01			:TQ10000	Q1,00	1800	P->S	54000	
	3   PX PARTITION LIST SINGLE	1		00:00:01	2	2	Q1,00		1800	PCWC		
8 (100)	4   CONTAINERS FULL	1		00:00:01			DEPT		1800	PCWP	54000	
							Q1,00					

```
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	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	Bytes	PQ
Distrib									
0	SELECT STATEMENT							1800	59400
8 (100)		00:00:01							
1	PX COORDINATOR								
2	PX SEND QC (RANDOM)					:TQ10000		1800	59400
8 (100)		00:00:01				Q1,00	P->S	QC (RAND)	
3	PX PARTITION LIST SINGLE							1800	59400
8 (100)		00:00:01		3	3	Q1,00		PCWC	
4	CONTAINERS FULL					EMP		1800	59400
8 (100)		00:00:01				Q1,00	PCWP		

```
SQL> EXIT
$
```

3. Execute the \$HOME/labs/app/Dynamic\_container\_map.sh shell script to re-create the HR\_ROOT application root, install HR\_APP in the application root, and then create the ACCOUNTING, RESEARCH, and SALES application PDBs. The application HR\_APP contains two SCOTT tables, DEPT and EMP. There is a container map table created and set at application root HR\_ROOT level that maps department 10 to ACCOUNTING PDB, department 20 to RESEARCH PDB, and department 30 to SALES PDB.

```
$ $HOME/labs/app/Dynamic_container_map.sh
...
$
```

4. Connect to HR\_ROOT as SYS and verify the content of SCOTT.DEPT and SCOTT.EMP tables.

```
$ sqlplus sys@HR_ROOT AS SYSDBA

Enter password: password

SQL> SELECT deptno, dname, loc FROM scott.dept;

DEPTNO DNAME          LOC
----- -----
      10 ACCOUNTING    NEW YORK
      30 SALES         CHICAGO
      20 RESEARCH      DALLAS
```

```
SQL> SELECT empno, ename, deptno FROM scott.emp;
```

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

14 rows selected.

```
SQL>
```

5. Find the application PDBs associated to HR\_ROOT.

```
SQL> SHOW PDBS
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
6	HR_ROOT	READ WRITE	NO
7	SALES	READ WRITE	NO
8	ACCOUNTING	READ WRITE	NO
12	RESEARCH	READ WRITE	NO

```
SQL>
```

6. Find the container map table and the description of the container map table.

```
SQL> SELECT owner, table_name FROM dba_tables
      WHERE container_map_object='YES';
```

OWNER	TABLE_NAME
SCOTT	MAPTABLE

```
SQL> SELECT dbms_metadata.get_ddl('TABLE', 'MAPTABLE', 'SCOTT')
      FROM dual;
```

```

2
DBMS_METADATA.GET_DDL('TABLE','MAPTABLE','SCOTT')
-----

CREATE TABLE "SCOTT"."MAPTABLE"
(
    "DEPTNO" NUMBER,
    "NAME" VARCHAR2(30)
) PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
STORAGE (
    BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
TABLESPACE "SYSTEM"
PARTITION BY LIST ("DEPTNO")
(PARTITION "ACCOUNTING" VALUES (10) SEGMENT CREATION DEFERRED
PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE (
    BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
TABLESPACE "SYSTEM",
PARTITION "RESEARCH" VALUES (20) SEGMENT CREATION DEFERRED
PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE (
    BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
TABLESPACE "SYSTEM",
PARTITION "SALES" VALUES (30) SEGMENT CREATION DEFERRED
PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE (
    BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
TABLESPACE "SYSTEM" )

SQL>

```

7. Verify that the query for department 20 is routed to the appropriate PDB.

```

SQL> SET LINESIZE 180
SQL> SELECT deptno, dname, loc FROM scott.dept
      WHERE deptno = 20;
2
DEPTNO  DNAME          LOC
-----
```

```
SQL> SELECT * FROM dbms_xplan.display_cursor();
PLAN_TABLE_OUTPUT
-----
SQL_ID  fappp8g8cxr4jm, child number 0
-----
SELECT deptno, dname, loc FROM scott.dept      WHERE deptno = 20

Plan hash value: 1918422367

-----
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU) | Time      | Pstart|Pstop |
| 0  | SELECT STATEMENT   |       |        |        |            |           |          |        |
| 1  | PARTITION LIST SINGLE |       | 700   | 21000 | 2 (100) | 00:00:01 | 2      |
| 2  | CONTAINERS FULL    | DEPT | 700   | 21000 | 2 (100) | 00:00:01 |          |
|
16 rows selected.

SQL>
```

The output of the execution plan is truncated to display the relevant columns only.

8. Create a new PDB for departments 60, 70, 80, 90, and 100.

```
SQL> CREATE PLUGGABLE DATABASE devt
          ADMIN USER admin IDENTIFIED BY password
          CREATE _FILE _DEST='u02/app/oracle/oradata/ORCL/hr_root/devt'
          CONTAINER _MAP UPDATE
                  (ADD PARTITION devt VALUES (60, 70, 80, 90,100));
2      3      4      5
Pluggable database created.

SQL>
```

- ## 9. Open the PDB.

```
SQL> ALTER PLUGGABLE DATABASE devt OPEN;  
  
Pluggable database altered.  
  
SQL>
```

10. Synchronize the application HR\_APP in DEVT.

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

Pluggable database altered.

SQL>
```

11. Insert rows into SCOTT.DEPT and SCOTT.EMP.

```
SQL> INSERT INTO scott.DEPT VALUES (60,'DEVT','FRANCE');

1 row created.

SQL> INSERT INTO scott.DEPT VALUES (70,'DEVT','GERMANY');

1 row created.

SQL> INSERT INTO scott.DEPT VALUES (40,'Admin','Poland');
INSERT INTO scott.DEPT VALUES (40,'Admin','Poland')
*
ERROR at line 1:
ORA-65391: violation of container map partitions

SQL> INSERT INTO scott.EMP VALUES
(7499,'ALLEN','Dev',7698,TO_date('20-2-1981','dd-mm-
yyyy'),1600,300,60);

1 row created.

SQL> INSERT INTO scott.EMP VALUES
(7521,'WARD','Dev',7698,TO_date('22-2-1981','dd-mm-
yyyy'),1250,500,70);

1 row created.

SQL> INSERT INTO scott.EMP VALUES
(7654,'MARTIN','Admin',7698,TO_date('28-9-1981','dd-mm-
yyyy'),1250,1400,40);
INSERT INTO scott.EMP VALUES
(7654,'MARTIN','Admin',7698,TO_date('28-9-1981','dd-mm-
yyyy'),1250,1400,40)
```

```
*  
ERROR at line 1:  
ORA-02291: integrity constraint (SCOTT.FK_DEPTNO) violated -  
parent  
key not found  
  
SQL> COMMIT;  
  
Commit complete.  
  
SQL>
```

*Q/ Why does the third and last insert fail?*

**A/ Department 40 does not refer to any PDB whose value is defined in the partitions of the LIST partitioned map table SCOTT.MAPTABLE.**

12. Verify that departments 60, 70 correspond to values defined in partitions of the LIST partitioned map table SCOTT.MAPTABLE.

```
SQL> CONNECT sys@HR_ROOT AS SYSDBA  
Enter password: password  
Connected.  
SQL> SELECT partition_name, high_value  
      FROM dba_tab_partitions  
     WHERE table_name='MAPTABLE' ORDER BY 1;  
2      3  
PARTITION_NAME HIGH_VALUE  
-----  
ACCOUNTING      10  
DEVT            60, 70, 80, 90, 100  
RESEARCH         20  
SALES           30  
  
SQL>
```

*Q/ Are the partitions automatically created in the map table?*

**A/ Yes. The creation of the PDB DEVT for departments 60, 70, 80, 90, and 100 defined a new partition in the LIST partitioned map table SCOTT.MAPTABLE.**

13. You want to maintain departments 60, 70 in the DEVT PDB but move the rest of the values defined in DEVT in another PDB, ADMINISTRATION.

```
SQL> CREATE PLUGGABLE DATABASE administration ADMIN USER admin  
      IDENTIFIED BY password  
CREATE _FILE_DEST =  
  '/u02/app/oracle/oradata/ORCL/hr_root/administration'  
CONTAINER_MAP UPDATE  
  (SPLIT PARTITION devt  
   INTO (PARTITION devt values (60, 70),  
         PARTITION administration));  
2   3   4   5   6   7   8  
Pluggable database created.  
  
SQL>
```

14. Open the PDB.

```
SQL> ALTER PLUGGABLE DATABASE administration OPEN;  
  
Pluggable database altered.  
  
SQL>
```

15. Synchronize the application HR\_APP in ADMINISTRATION.

```
SQL> CONNECT sys@ADMINISTRATION AS SYSDBA  
Enter password: password  
Connected.  
SQL> ALTER PLUGGABLE DATABASE APPLICATION hr_app SYNC;  
  
Pluggable database altered.  
  
SQL>
```

16. Verify that departments 60, 70 correspond to values defined for DEVT PDB and 80, 90, and 100 for ADMINISTRATION PDB and that a new partition is created in the LIST partitioned map table SCOTT.MAPTABLE.

```
SQL> CONNECT sys@HR_ROOT AS SYSDBA  
Enter password: password  
Connected.  
SQL> SELECT partition_name, high_value  
      FROM dba_tab_partitions  
     WHERE table_name='MAPTABLE' ORDER BY 1;  
2   3  
PARTITION_NAME HIGH_VALUE  
-----
```

```
ACCOUNTING      10
ADMINISTRATION 80, 90, 100
DEVT            60, 70
RESEARCH         20
SALES           30

SQL> EXIT
$
```

17. Execute the \$HOME/labs/app/cleanup\_container\_map.sh shell script. It drops DEVT and ADMINISTRATION PDBs.

```
$ $HOME/labs/app/cleanup_container_map.sh
...
$
```

## **Practices for Lesson 4: PDB Creation**

## Practices for Lesson 4: Overview

---

### Practices Overview

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

- Cloning a regular PDB in hot mode and with automatic refreshing
- Cloning an application PDB
- Unplugging and plugging an application container
- Converting regular PDBs to application PDBs
- Relocating a PDB
- Creating and using a proxy PDB

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

In case you need to **re-create the ORCL CDB and its PDB1 PDB**, use the

/home/oracle/labs/admin/recreate\_ORCL.sh shell script.

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

In case you need to **create or re-create the CDB18 CDB without its PDB18 PDB**, use the

/home/oracle/labs/creation/create\_empty\_CDB18.sh shell script.

```
$ $HOME/labs/creation/create_empty_CDB18.sh  
...  
$
```

## Practice 4-1: Cloning Remote PDBs in Hot Mode

---

### Overview

In this practice, because you have been informed of performance issues on the `pdb_source_for_hotclone` PDB in `ORCL`, you will clone in hot mode the PDB as the `pdb_hotclone` PDB in the `cdb2` test instance 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 refreshable copy, refreshed manually or automatically, which will allow you to take your time to test the performance issue.

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created in `ORCL` and finally re-creates `PDB1`.

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

2. Then execute the `$HOME/labs/PDB/glogin_4.sh` shell script. It sets formatting for all columns selected in queries.

```
$ $HOME/labs/PDB/glogin_4.sh
$
```

3. Execute the `$HOME/labs/PDB/cleanup_apps.sh` script that drops unnecessary PDBs in `ORCL` and then the `$HOME/labs/PDB/setup_hotclone.sh` script that creates the production `pdb_source_for_hotclone` from the CDB seed in `ORCL` and drops unnecessary PDBs in `CDB18`, 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/PDB/cleanup_apps.sh
...
$ $HOME/labs/PDB/setup_hotclone.sh
...
$
```

4. In `CDB18`, 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, named Session `ORCL`.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ $HOME/labs/PDB/glogin_4.sh
```

```

$ 
$ sqlplus system@pdb_source_for_hotclone

SQL> SET sqlprompt "ORCL> "
ORCL> SELECT DISTINCT label FROM source_user.bigtab;

LABEL
-----
DATA FROM source_user.bigtab

ORCL> UPDATE source_user.bigtab SET label='DATA';

10000 rows updated.

ORCL>

```

- b. In the original terminal window, called *Session CDB18*, clone the `pdb_hotclone` from `pdb_source_for_hotclone` while the source PDB is still up and fully functional.

```

$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ mkdir /u02/app/oracle/oradata/CDB18/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 password
      USING 'pdb_source_for_hotclone';
2   3   4
Database link created.

SQL> ALTER SESSION SET
  db_create_file_dest='/u02/app/oracle/oradata/CDB18/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
Pluggable database created.

SQL>
```

5. Open `pdb_hotclone` in READ ONLY mode only.

```
SQL> ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY;

Pluggable database altered.

SQL>
```

6. 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>
```

7. In Session `ORCL`, commit the updated production source data in `ORCL`.

```
ORCL> COMMIT;

Commit complete.

SQL2>
```

8. In Session `CDB18`, refresh the data in the cloned PDB in `CDB18`.

```
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>
```

9. Drop the current refreshable copy PDB and re-create 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
PDB18        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/CDB18/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.**

10. In Session ORCL, update and commit the source data in ORCL.

```
ORCL> UPDATE source_user.bigtab SET label='DATA2';

10000 rows updated.

ORCL> COMMIT;

Commit complete.

ORCL> EXIT
$
```

11. In Session CDB18, wait two minutes and then check 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/ If you do not see the updated value, what is the reason?*

**A/ The refresh is scheduled every two 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 two 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
```

```
$
```

12. Execute the \$HOME/labs/PDB/cleanup\_hotclones.sh script to drop the pdb\_source\_for\_hotclone in ORCL and pdb\_hotclone in CDB18.

```
$ $HOME/labs/PDB/cleanup_hotclones.sh  
...  
$
```

## Practice 4-2: Cloning an Application Container

---

### Overview

In this practice, because you have been informed of performance issues on the `robots` application PDB in `ORCL`, you will clone the `robots` application PDB in the `CDB18` test instance for performance tests. The remote `robots` production application PDB in `ORCL` will still be up and fully functional while the actual clone operation is taking place.

### Tasks

1. Execute the `$HOME/labs/PDB/setup_toys_app.sh` script that creates the production `toys_root` application container and its `robots` application PDB in `ORCL`. In the `robots` application PDB, there are rows in the `toys_owner.sales_data` table.

```
$ $HOME/labs/PDB/setup_toys_app.sh
...
$
```

2. In `CDB18`, clone the production `toys_root` application container including its `robots` application PDB in hot mode.
  - a. In Session `ORCL`, start a transaction in the production `robots` application PDB.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ $HOME/labs/PDB/glogin_4.sh
$
$ sqlplus system@robots

SQL> SET sqlprompt "ORCL> "
ORCL> SELECT * FROM toys_owner.sales_data;
```

YEAR	REGION	QUAR	REVENUE
2012	R_north	Q1	15058
2012	R_north	Q2	12737
2012	R_north	Q3	14342
2012	R_north	Q4	19732
2013	R_north	Q1	14145
2013	R_north	Q2	16359
2013	R_north	Q3	19235
2013	R_north	Q4	18596
2012	R_south	Q1	15547
2012	R_south	Q2	15947
2012	R_south	Q3	12267

```

2012 R_south    Q4      19371
2013 R_south    Q1      11101
2013 R_south    Q2      19306
2013 R_south    Q3      11337
2013 R_south    Q4      16164
2012 R_east     Q1      19287
2012 R_east     Q2      19147
2012 R_east     Q3      10722
2012 R_east     Q4      17645
2013 R_east     Q1      15488
2013 R_east     Q2      15173
2013 R_east     Q3      11118
2013 R_east     Q4      16020
2012 R_west     Q1      10142
2012 R_west     Q2      13028
2012 R_west     Q3      10121
2012 R_west     Q4      14060
2013 R_west     Q1      14957
2013 R_west     Q2      16470
2013 R_west     Q3      15457
2013 R_west     Q4      16884

32 rows selected.

ORCL> UPDATE toys_owner.sales_data SET revenue = revenue*10;

32 rows updated.

ORCL>

```

- b. In Session CDB18, clone in CDB18 the toys\_root\_dev application root from the toys\_root application root of ORCL while the toys\_root is still up and fully functional.

```

$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ mkdir -p /u02/app/oracle/oradata/CDB18/toys_root_dev
$ mkdir /u02/app/oracle/oradata/CDB18/toys_root_dev/robots_dev
$ mkdir /u02/app/oracle/oradata/CDB18/toys_root_dev/dolls_dev
$ sqlplus / AS SYSDBA

SQL> DROP PUBLIC DATABASE LINK link_toys_root;

```

```
DROP PUBLIC DATABASE LINK link_toys_root
*
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE PUBLIC DATABASE LINK
      link_toys_root
      CONNECT TO system IDENTIFIED BY password
      USING 'toys_root';
2   3   4
Database link created.

SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/CDB18/toys_root_dev
';
2
Session altered.

SQL> CREATE PLUGGABLE DATABASE toys_root_dev
      AS APPLICATION CONTAINER
      FROM toys_root@link_toys_root;
2   3
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE toys_root_dev OPEN;
Pluggable database altered.

SQL> CONNECT sys@toys_root_dev AS SYSDBA
Enter password: password
Connected.

SQL> CREATE PUBLIC DATABASE LINK
      link_robots
      CONNECT TO system IDENTIFIED BY password
      USING 'robots';
2   3   4
Database link created.

SQL> CREATE PUBLIC DATABASE LINK
      link_dolls
      CONNECT TO system IDENTIFIED BY password
      USING 'dolls';
2   3   4
```

```
Database link created.

SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/CDB18/toys_root_dev
/robots_dev';
2
Session altered.

SQL> CREATE PLUGGABLE DATABASE robots_dev
      FROM robots@link_robots;
2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE robots_dev OPEN;

Pluggable database altered.

SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/CDB18/toys_root_dev
/dolls_dev';
2
Session altered.

SQL> CREATE PLUGGABLE DATABASE dolls_dev
      FROM robots@link_dolls;
2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE dolls_dev OPEN;

Pluggable database altered.

SQL>
```

3. In Session ORCL, commit the updated production source data in ORCL.

```
ORCL> COMMIT;

Commit complete.

ORCL> SELECT * FROM toys_owner.sales_data;

      YEAR REGION      QUAR      REVENUE
----- ----- ----- -----

```

2012	R_north	Q1	150580
2012	R_north	Q2	127370
2012	R_north	Q3	143420
2012	R_north	Q4	197320
2013	R_north	Q1	141450
2013	R_north	Q2	163590
2013	R_north	Q3	192350
2013	R_north	Q4	185960
2012	R_south	Q1	155470
2012	R_south	Q2	159470
2012	R_south	Q3	122670
2012	R_south	Q4	193710
2013	R_south	Q1	111010
2013	R_south	Q2	193060
2013	R_south	Q3	113370
2013	R_south	Q4	161640
2012	R_east	Q1	192870
2012	R_east	Q2	191470
2012	R_east	Q3	107220
2012	R_east	Q4	176450
2013	R_east	Q1	154880
2013	R_east	Q2	151730
2013	R_east	Q3	111180
2013	R_east	Q4	160200
2012	R_west	Q1	101420
2012	R_west	Q2	130280
2012	R_west	Q3	101210
2012	R_west	Q4	140600
2013	R_west	Q1	149570
2013	R_west	Q2	164700
2013	R_west	Q3	154570
2013	R_west	Q4	168840

32 rows selected.

ORCL> **EXIT**

\$

- In Session CDB18, connect to `robots_dev` and check that the data in `toys_owner.sales_data` is as it was originally before the COMMIT.

SQL> **CONNECT toys\_owner@robots\_dev**

Enter password: *password*

Connected.

```
SQL> SELECT * FROM toys_owner.sales_data;

        YEAR REGION      QUAR     REVENUE
-----  -----
2012 R_north    Q1      15058
2012 R_north    Q2      12737
2012 R_north    Q3      14342
2012 R_north    Q4      19732
2013 R_north    Q1      14145
2013 R_north    Q2      16359
2013 R_north    Q3      19235
2013 R_north    Q4      18596
2012 R_south    Q1      15547
2012 R_south    Q2      15947
2012 R_south    Q3      12267
2012 R_south    Q4      19371
2013 R_south    Q1      11101
2013 R_south    Q2      19306
2013 R_south    Q3      11337
2013 R_south    Q4      16164
2012 R_east     Q1      19287
2012 R_east     Q2      19147
2012 R_east     Q3      10722
2012 R_east     Q4      17645
2013 R_east     Q1      15488
2013 R_east     Q2      15173
2013 R_east     Q3      11118
2013 R_east     Q4      16020
2012 R_west     Q1      10142
2012 R_west     Q2      13028
2012 R_west     Q3      10121
2012 R_west     Q4      14060
2013 R_west     Q1      14957
2013 R_west     Q2      16470
2013 R_west     Q3      15457
2013 R_west     Q4      16884

32 rows selected.

SQL> EXIT
$
```

5. Execute the \$HOME/labs/PDB/cleanup\_apps\_CDB18.sh shell script to drop all PDBs in CDB18 except the regular PDB18.

```
$ $HOME/labs/PDB/cleanup_apps_CDB18.sh  
...  
$
```

## Practice 4-3: Unplugging and Plugging Application Containers

### Overview

In this practice, you will create a new application container by unplugging `toys_root` application container and its two application PDBs, `robots` and `dolls`, from `ORCL` and plugging them all into `CDB18` by using archive files.

### Tasks

- Before starting the practice, execute the `$HOME/labs/PDB/cleanup_apps_CDB18.sh` shell script if you did not execute it at the end of the previous practice.

Then execute the `$HOME/labs/PDB/setup_toys_app.sh` script that sets up the `toys_root` application container in `ORCL`.

```
$ $HOME/labs/PDB/setup_toys_app.sh
...
$
```

- Suppose you use the `CDB18` instance as the development instance and need to move the production `toys_root` application container from `ORCL` to `CDB18`. Unplug `toys_root` by using archive files for each container (the application root, the application seed, and the application PDBs) and plug the whole application container into `CDB18`.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ rm /tmp/*pdb
rm: cannot remove `/tmp/*pdb': No such file or directory
$ 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	PDB1	READ WRITE	NO	

```
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>
```

*Q/ 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.
//
```

*A/ 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 datafiles 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] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ mkdir -p /u02/app/oracle/oradata/CDB18/toys_root/robots
$ mkdir /u02/app/oracle/oradata/CDB18/toys_root/dolls
$ mkdir /u02/app/oracle/oradata/CDB18/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_inViolations;

NAME
-----
MESSAGE
-----
ACTION
-----
TOYS_ROOT
Application Root plugged in as an Non-Application PDB, requires
aproot_to_pdb.sql be run.
Run aproot_to_pdb.sql.

SQL>

```

3. Plug the toys\_root application container into CDB18.

***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/CDB18/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: password
Connected.
SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/CDB18/toys_root/toy
s_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/CDB18/toys_root/rob
ots';
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.
```

```

SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/CDB18/toys_root/dolls';
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>

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

```

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

CODE LABEL
-----
1 Puppet
2 Car

SQL> CONNECT toys_owner@dolls

```

```
Enter password: password
Connected.
SQL> SELECT * FROM sales_data;

no rows selected

SQL> EXIT
$
```

5. Execute the \$HOME/labs/PDB/cleanup\_toys\_app\_CDB18.sh shell script to drop the toys\_root application container in CDB18.

```
$ $HOME/labs/PDB/cleanup_toys_app_CDB18.sh
...
$
```

## Practice 4-4: Converting a Regular PDB to an Application PDB

### Overview

In this practice, you will convert the regular PDB18 of CDB18 as an application PDB in the hr\_root application container in CDB18 so that it can take advantage of all common tables. The conversion uses the clone method. research is cloned from CDB18 as the research application PDB of the hr\_root application container.

### Tasks

1. Execute the \$HOME/labs/PDB/setup\_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/PDB/setup_hr_app.sh
...
$
```

2. Clone PDB18 of CDB18 as the research application PDB in the hr\_root application container.
  - a. Open PDB18 in read-only mode before cloning it.

```
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE pdb18 CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb18 OPEN READ ONLY;

Pluggable database altered.

SQL>
```

- b. Now, clone PDB18 as the research application PDB in the hr\_root application container.

```
SQL> !mkdir /u02/app/oracle/oradata/CDB18/hr_root/research
SQL> ALTER SESSION SET db_create_file_dest =
      '/u02/app/oracle/oradata/CDB18/hr_root/research';
2
Session altered.
```

```
SQL> CREATE PLUGGABLE DATABASE research FROM pdb18;

Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE research OPEN;

Pluggable database altered.

SQL>
```

*Q/ How can you ensure that research is an application PDB associated to hr\_root?*

**A/ 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
PDB18           3    NO      NO      NO
HR_ROOT         4    YES     NO      NO
OPERATIONS      5    NO      NO      YES
RESEARCH         6    NO      NO      NO
4

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: password
Connected.

SQL> ALTER SESSION SET db_create_file_dest =
  '/u02/app/oracle/oradata/CDB18/hr_root/research';
2
Session altered.

SQL> CREATE PLUGGABLE DATABASE research FROM pdb18;

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/ What is the reason 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_appdb.sql be run.
PENDING
Run pdb_to_appdb.sql.

SQL>
```

*A3/ The content of the PDB\_PLUG\_IN\_VIOLATIONS view explains that the main error is that the pdb\_to\_appdb.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> ALTER SESSION SET CONTAINER = research;

Session altered.

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_appadb
Rem
Rem $Header: rdbms/admin/pdb_to_appadb.sql /main/2 2016/01/22
08:51:46 thbabay Exp $
Rem
Rem pdb_to_appadb.sql
Rem
Rem Copyright (c) 2015, 2016, Oracle and/or its affiliates.
Rem All rights reserved.
Rem
Rem      NAME
Rem      pdb_to_appadb.sql - PDB to Federation PDB
Rem
Rem      DESCRIPTION
Rem          Converts PDB (standalone or Application ROOT) to
Application PDB
...
SQL> -- leave the PDB in the same state it was when we started
SQL> BEGIN
 2     execute immediate '&open_sql &restricted_state';
 3 EXCEPTION
 4     WHEN OTHERS THEN
 5     BEGIN
 6         IF (sqlcode <> -900) THEN
 7             RAISE;
 8         END IF;
 9     END;
10 END;
11 /
```

```
PL/SQL procedure successfully completed.
```

```
SQL>
SQL> WHENEVER SQLERROR CONTINUE;
SQL>
SQL>
```

*Q/ Is the shared application table hr\_mgr.mgr\_tab recognized in research?*

```
SQL> SHOW CON_NAME

CON_NAME
-----
RESEARCH

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: 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/CDB18/hr_root
```

```
$
```

## Practice 4-5: Relocating PDBs

In this practice, you will move the test PDB1 from ORCL into the production CDB18 in one step, using the Near-zero Downtime PDB Relocation feature, called PDB relocation.

### Tasks

- Even if PDB1 exists in ORCL, execute the `setup_pdb1.sh` shell script to re-create the PDB with a table containing a lot of rows.

```
$ $HOME/labs/PDB/setup_pdb1.sh
...
$
```

- In Session 1, verify that the source remote CDB is configured to use local UNDO.

```
$ . oraenv
ORACLE_SID = [CDB18] ? 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@PDB1
Enter password: password
Connected.
SQL> SELECT LABEL, COUNT(*) FROM test.bigtab GROUP BY label;

LABEL                      COUNT(*)
-----
DATA FROM test.bigtab          10000

SQL>
```

- Prepare a terminal window, named Session 2, to start a session and a transaction in PDB1 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/PDB/sessions.sh` shell script until `CREATE PLUGGABLE DATABASE pdb_relocated` is ready in the terminal window (3), named Session 3.

4. In Session 1, relocate PDB1 from ORCL into CDB18 as PDB\_RELOCATED.
- In ORCL, in Session 1, create the database link to access CDB18.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> DROP PUBLIC DATABASE LINK link_CDB18;
DROP PUBLIC DATABASE LINK link_CDB18
*
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE PUBLIC DATABASE LINK link_CDB18
      CONNECT TO system IDENTIFIED BY password
      USING 'CDB18';
2      3
Database link created.

SQL>
```

- In Session 3, in CDB18, create the database link to access PDB1 in ORCL.

```
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
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 password
      USING 'ORCL';
2      3
Database link created.

SQL>
```

- Relocate PDB1. Display the status of the new PDB.

```
SQL> !mkdir /u02/app/oracle/oradata/CDB18/pdb_relocated

SQL> ALTER SESSION SET db_create_file_dest =
      '/u02/app/oracle/oradata/CDB18/pdb_relocated';
```

```
2
System altered.

SQL> CREATE PLUGGABLE DATABASE pdb_relocated
      FROM pdb1@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 *SYSPER* 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 data 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. In Session 1, grant the right privilege to SYSTEM.

```
SQL> GRANT sysoper TO system CONTAINER=all;

Grant succeeded.

SQL>
```

- e. In Session 3, now relocate PDB\_RELOCATED from ORCL into CDB18. You can relocate with the `AVAILABILITY MAX` clause, which ensures smooth migration of workload and persistent connection forwarding from ORCL to CDB18.

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 pdb1@link_ORCL RELOCATE;

Pluggable database created.

SQL> SELECT pdb_name, status FROM cdb_pdbs;

PDB_NAME          STATUS
----- -----
PDB_RELOCATED     RELOCATING
PDB$SEED          NORMAL
PDB18              NORMAL

SQL>
```

5. In Session 2, execute `$HOME/labs/PDB/session.sh`. It will last around 5000 seconds.

```
$ $HOME/labs/PDB/session.sh
SQL> Connected.
SQL> 2   3   4   5   6   7   8
```

6. During `session.sh` execution:

- a. In Session 1, you can display the status of the source PDB.

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

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

SQL>
```

- b. In Session 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? (Your number may differ from the example.)*

*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.
- The source PDB is closed and dropped from the source CDB.
  - 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;
2
Pluggable database altered.

SQL>
```

*Observe that this interrupts sessions.sh execution taking place in Session 2.*

```

*
ERROR at line 1:
ORA-01089: immediate shutdown or close in progress - no
operations
are permitted
Process ID: 18163
Session ID: 277 Serial number: 38558

SQL>
$
```

8. Still in Session 3, verify that the application data is relocated in `pdb_relocated` in CDB18.

```

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> SELECT pdb_name, status FROM cdb_pdbs;

PDB_NAME      STATUS
-----
PDB_RELOCATED NORMAL

SQL>
```

9. In Session 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

SQL> EXIT
$
```

**Q/ In which situation would the source PDB not be dropped?**

**A/ 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.*

*Do not execute the following commands or test it in another CDB.*

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> !mkdir /u02/app/oracle/oradata/CDB18/pdb_relocated2

SQL> ALTER SESSION SET db_create_file_dest =
      '/u02/app/oracle/oradata/CDB18/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 Session 3, drop `PDB_RELOCATED` in CDB18.

```
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 by 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 `CDB18`. 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/PDB/setup2_toys_app.sh` shell script to create the `toys_root` application container as requested in points 1) and 2) above.

```
$ $HOME/labs/PDB/setup2_toys_app.sh
...
$
```

2. In `CDB18`, 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 = [CDB18] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ mkdir -p /u02/app/oracle/oradata/CDB18/toys_rr
$ sqlplus / AS SYSDBA

SQL> ALTER SESSION SET
db_create_file_dest='/u02/app/oracle/oradata/CDB18/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 = [CDB18] ? 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: password

SQL> DROP PUBLIC DATABASE LINK link_CDB18;
DROP PUBLIC DATABASE LINK link_CDB18
*
ERROR at line 1:
ORA-02024: database link not found

SQL> CREATE PUBLIC DATABASE LINK link_CDB18
      CONNECT TO system IDENTIFIED BY password
      USING 'CDB18';
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_CDB18;
2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE px_toys_rr OPEN;

Pluggable database altered.

SQL>

```

*Q/ What would have happened if you had created the proxy PDB in CDB root?*

*A/ 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 dba_pdbs;				
2	3	PDB_NAME	CON_ID	IS_ FOREIGN_CDB_DBID FOREIGN_PDB_ID
-----				
DOLLS	10	NO	1464398411	2
PX_TOYS_RR	11	YES	<b>685052628</b>	<b>5</b>
TOYS_ROOT	8	NO	1464398411	2
ROBOTS	9	NO	1464398411	2

```

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@CDB18
Connected.
SQL> SELECT dbid FROM v$database;

          DBID
-----
685052628

SQL> SELECT pdb_name, con_id FROM cdb_pdbs;

PDB_NAME      CON_ID
-----
PDB$SEED        2
PDB18           3
TOYS_RR         5

SQL>

```

**The FOREIGN\_CDB\_DBID in the first query in ORCL corresponds to the DBID in the second query in CDB18. The FOREIGN\_PDB\_ID in the first query in ORCL corresponds to the CON\_ID in the third query in CDB18.**

- 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: password
Connected.
SQL> SELECT * FROM toys_owner.sales_data;

      YEAR REGION     QUAR    REVENUE
-----
2000  US          Q4        1
2000  FRANCE      Q3        2
2000  GERMANY     Q2        4
2000  UK          Q1        3

SQL> SELECT * FROM toys_owner.codes;

      CODE LABEL      CON_ID
-----
1  Puppet          5
2  Car             5

SQL> CONNECT system@px_toys_rr

```

```

Enter password: password
Connected.
SQL> SELECT * FROM toys_owner.sales_data;

      YEAR REGION     QUAR    REVENUE
-----  -----
  2000  US          Q4        1
  2000  FRANCE      Q3        2
  2000  GERMANY    Q2        4
  2000  UK          Q1        3

SQL> SELECT * FROM toys_owner.codes;

      CODE LABEL      CON_ID
-----  -----
    1  Puppet        5
    2  Car           5

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/PDB/setup\_doodles.sh shell script to create the doodles application PDB in toys\_rr, as requested in point 4) earlier.

```

$ $HOME/labs/PDB/setup_doodles.sh
...
$
```

4. Execute the \$HOME/labs/PDB/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/PDB/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: password

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
 WHERE con$name NOT IN ('TOYS_ROOT','TOYS_RR')
 GROUP BY a.year, a.region, con$name, cdb$name
 ORDER BY con$name, a.year, a.region;
```

2	3	4	5
REVENUE	YEAR	REGION	CON\$NAME
2	2000	FRANCE	DOLLS
4		GERMANY	DOLLS
3		UK	DOLLS
1		US	DOLLS

-----  
\*\*\*\*\*  
10 Yearly Rev

60672	2012	east	DOLLS	ORCL
50880		north	DOLLS	ORCL
66009		south	DOLLS	ORCL
62363		west	DOLLS	ORCL

-----  
\*\*\*\*\*  
239924 Yearly Rev

59705	2013	east	DOLLS	ORCL
63172		north	DOLLS	ORCL
52477		south	DOLLS	ORCL
51195		west	DOLLS	ORCL

-----  
\*\*\*\*\*  
226549 Yearly Rev

2	2000	FRANCE	DOODLES	CDB18
4		GERMANY	DOODLES	CDB18
3		UK	DOODLES	CDB18
1		US	DOODLES	CDB18

-----  
\*\*\*\*\*  
10 Yearly Rev

64920	2012	east	DOODLES	CDB18
68265		north	DOODLES	CDB18
60212		south	DOODLES	CDB18
56744		west	DOODLES	CDB18

-----  
\*\*\*\*\*

250141 Yearly Rev

56481	2013	east	DOODLES	CDB18
53129		north	DOODLES	CDB18
60081		south	DOODLES	CDB18
55328		west	DOODLES	CDB18

----- \*\*\*\*\*

225019 Yearly Rev

2	2000	FRANCE	ROBOTS	ORCL
4		GERMANY	ROBOTS	ORCL
3		UK	ROBOTS	ORCL
1		US	ROBOTS	ORCL

----- \*\*\*\*\*

10 Yearly Rev

53279	2012	R_east	ROBOTS	ORCL
59010		R_north	ROBOTS	ORCL
67382		R_south	ROBOTS	ORCL
68094		R_west	ROBOTS	ORCL
60667		east	ROBOTS	ORCL
64025		north	ROBOTS	ORCL
73187		south	ROBOTS	ORCL
55809		west	ROBOTS	ORCL

----- \*\*\*\*\*

501453 Yearly Rev

59904	2013	R_east	ROBOTS	ORCL
69602		R_north	ROBOTS	ORCL
71851		R_south	ROBOTS	ORCL
44336		R_west	ROBOTS	ORCL
56107		east	ROBOTS	ORCL
54777		north	ROBOTS	ORCL
57356		south	ROBOTS	ORCL
63599		west	ROBOTS	ORCL

----- \*\*\*\*\*

477532 Yearly Rev

44 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 CDB18 and ORCL is retrieved.**

**Note: If the DOODLES data is not displayed, retry a few seconds later.**

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 CDB18.
  - a. Execute the `$HOME/labs/PDB/relocate_dolls.sh` shell script. The script relocates dolls from ORCL to CDB18.

```
$ $HOME/labs/PDB/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: password
```

```
SQL> SHOW PDBS
```

CON_ID	CON_NAME	OPEN	MODE	RESTRICTED
8	TOYS_ROOT	READ	WRITE	NO
9	ROBOTS	READ	WRITE	NO
11	PX_TOYS_RR	READ	WRITE	NO

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

```
Enter password: password
```

```
Connected.
```

```
SQL> SHOW PDBS
```

CON_ID	CON_NAME	OPEN	MODE	RESTRICTED
5	TOYS_RR	READ	WRITE	NO
4	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: 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
 WHERE con$name NOT IN ('TOYS_ROOT', 'TOYS_RR')
 GROUP BY a.year, a.region, con$name, cdb$name
 ORDER BY con$name, a.year, a.region;

```

2	3	4	5	
REVENUE		YEAR	REGION	CON\$NAME
2		2000	FRANCE	DOLLS CDB18
4			GERMANY	DOLLS CDB18
3			UK	DOLLS CDB18
1			US	DOLLS CDB18
----- ***** -----				
10 Yearly Rev				
60672		2012	east	DOLLS CDB18
50880			north	DOLLS CDB18
66009			south	DOLLS CDB18
62363			west	DOLLS CDB18
----- ***** -----				
239924 Yearly Rev				
59705		2013	east	DOLLS CDB18
63172			north	DOLLS CDB18
52477			south	DOLLS CDB18
51195			west	DOLLS CDB18
----- ***** -----				
226549 Yearly Rev				
2		2000	FRANCE	DOODLES CDB18
4			GERMANY	DOODLES CDB18
3			UK	DOODLES CDB18
1			US	DOODLES CDB18
----- ***** -----				
10 Yearly Rev				
64920		2012	east	DOODLES CDB18
68265			north	DOODLES CDB18
60212			south	DOODLES CDB18
56744			west	DOODLES CDB18

-----	*****			
250141 Yearly Rev				
56481	2013	east	DOODLES	CDB18
53129		north	DOODLES	CDB18
60081		south	DOODLES	CDB18
55328		west	DOODLES	CDB18
-----	*****			
225019 Yearly Rev				
2	2000	FRANCE	ROBOTS	ORCL
4		GERMANY	ROBOTS	ORCL
3		UK	ROBOTS	ORCL
1		US	ROBOTS	ORCL
-----	*****			
10 Yearly Rev				
53279	2012	R_east	ROBOTS	ORCL
59010		R_north	ROBOTS	ORCL
67382		R_south	ROBOTS	ORCL
68094		R_west	ROBOTS	ORCL
60667		east	ROBOTS	ORCL
64025		north	ROBOTS	ORCL
73187		south	ROBOTS	ORCL
55809		west	ROBOTS	ORCL
-----	*****			
501453 Yearly Rev				
59904	2013	R_east	ROBOTS	ORCL
69602		R_north	ROBOTS	ORCL
71851		R_south	ROBOTS	ORCL
44336		R_west	ROBOTS	ORCL
56107		east	ROBOTS	ORCL
54777		north	ROBOTS	ORCL
57356		south	ROBOTS	ORCL
63599		west	ROBOTS	ORCL
-----	*****			
477532 Yearly Rev				
44 rows selected.				
SQL> <b>EXIT</b>				

## 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

- Drop `toys_rr` referenced by the `px_toys_rr` proxy PDB.

```
$ . oraenv
ORACLE_SID = [CDB18] ? CDB18
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 PDB18        READ ONLY NO
5 TOYS _RR     READ WRITE NO
4 DOODLES       READ WRITE NO
6 DOLLS        READ WRITE NO

SQL> ALTER PLUGGABLE DATABASE toys_rr CLOSE;
Pluggable database altered.

SQL>
```

- 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;
```

NAME	CON_ID	APP_ROOT	APP_Seed	APP_PDB	APP_ROOT_CONID
CDB\$ROOT	1	NO	NO	NO	
PDB\$SEED	2	NO	NO	NO	
PDB18	3	NO	NO	NO	

TOYS_RR	5	YES	NO	NO	
DOODLES	4	NO	NO	YES	5
DOLLS	6	NO	NO	YES	5

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
```

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 = [CDB18] ? ORCL
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
8	TOYS_ROOT	READ WRITE	NO

```

    9 ROBOTS                      READ WRITE NO
   11 PX_TOYS_RR                  READ WRITE NO
SQL> ALTER PLUGGABLE DATABASE toys_root CLOSE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE robots INCLUDING DATAFILES;

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE toys_root INCLUDING DATAFILES;

DROP PLUGGABLE DATABASE toys_root INCLUDING DATAFILES
*
ERROR at line 1:
ORA-65265: PDB cannot be dropped or unplugged.

SQL>

```

*Q/ What is the reason for this behavior?*

```

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
TOYS_ROOT      8  YES     NO      NO
PX_TOYS_RR     11 NO      NO      YES
                           8

SQL>

```

*A/ The PX\_TOYS\_RR is associated to TOYS\_ROOT.*

```

SQL> DROP PLUGGABLE DATABASE px_toys_rr INCLUDING DATAFILES;

Pluggable database dropped.

```

```
SQL> DROP PLUGGABLE DATABASE toys_root INCLUDING DATAFILES;  
Pluggable database dropped.  
  
SQL> EXIT  
$
```

4. Execute the \$HOME/labs/PDB/cleanup\_apps.sh shell script to finish the PDBs cleanup.

```
$ $HOME/labs/PDB/cleanup_apps.sh  
...  
$
```

## **Practices for Lesson 5: CDB and PDB Management**

## Practices for Lesson 5: Overview

---

### Practices Overview

In these practices, in ORCL CDB, you will start up and shut down CDBs, open and close PDBs, rename PDBs, change the different modes and settings of PDBs, evaluate the impact of parameter value changes, and finally avoid service name conflicts.

## Practice 5-1: Starting up and Shutting down a CDB

---

### Overview

In this practice, you shut down ORCL and start up ORCL.

In case you need to **re-create the ORCL CDB and its PDB1 PDB**, use the /home/oracle/labs/admin/recreate\_ORCL.sh shell script.

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

### Tasks

1. Execute the /home/oracle/labs/admin/cleanup\_PDBs.sh shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. Then execute the \$HOME/labs/mgt/glogin\_5.sh and \$HOME/labs/mgt/setup\_toys\_app.sh shell scripts. The first one sets formatting for all columns selected in queries, and the second one creates the toys\_root application container with two application PDBs, robots and dolls.

```
$ $HOME/labs/mgt/glogin_5.sh
$
```

```
$ $HOME/labs/mgt/setup_toys_app.sh
...
$
```

3. Connect to ORCL to shut it down.

  - a. Connect to the CDB as a user with SYSDBA privilege.

```
$ . oraenv
ORACLE_SID = [CDB18] ? ORCL
$ sqlplus / AS SYSDBA

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

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

SQL>
```

- b. Verify that the PDBs are open.

```
SQL> SHOW PDBS
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	PDB1	READ WRITE	NO
4	TOYS_ROOT	READ WRITE	NO
5	ROBOTS	READ WRITE	NO
6	DOLLS	READ WRITE	NO

```
SQL
```

- c. Shut down the CDB.

```
SQL> SHUTDOWN IMMEDIATE
```

Database closed.

Database dismounted.

ORACLE instance shut down.

```
SQL> EXIT
```

```
$
```

*Q/ Are the PDBs closed?*

*A/ Yes.*

- d. Explore the background processes.

```
$ pgrep -lf ORCL
$
```

4. Connect to ORCL and start it up.

```
$ sqlplus / AS SYSDBA
```

Connected to an idle instance.

```
SQL> STARTUP
```

ORACLE instance started.

```
Total System Global Area 4697620480 bytes
Fixed Size                  2923760 bytes
Variable Size                989856528 bytes
Database Buffers            3690987520 bytes
Redo Buffers                 13852672 bytes
Database mounted.
Database opened.
```

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

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

SQL>
```

5. Explore the background processes. The list you get on your server may vary from the list below.

```
SQL> ! pgrep -lf ORCL
17912 ora_pmon_ORCL
17914 ora_clmn_ORCL
17916 ora_psp0_ORCL
17918 ora_vktm_ORCL
17922 ora_gen0_ORCL
17924 ora_mman_ORCL
17928 ora_gen1_ORCL
17932 ora_diag_ORCL
17934 ora_ofsd_ORCL
17938 ora_dbrm_ORCL
17940 ora_vkrm_ORCL
17942 ora_svcb_ORCL
17944 ora_pman_ORCL
17946 ora_dia0_ORCL
17948 ora_dbw0_ORCL
17950 ora_lgwr_ORCL
17952 ora_ckpt_ORCL
17954 ora_lg00_ORCL
17956 ora_smon_ORCL
17958 ora_lg01_ORCL
17960 ora_smco_ORCL
17962 ora_reco_ORCL
17964 ora_w000_ORCL
17966 ora_lreg_ORCL
17968 ora_w001_ORCL
17970 ora_pxmn_ORCL
17974 ora_mmon_ORCL
17976 ora_mmnl_ORCL
17978 ora_d000_ORCL
17980 ora_s000_ORCL
17982 ora_tmon_ORCL
```

```

17992 oracleORCL
(DESCRIPTION=(LOCAL=YES) (ADDRESS=(PROTOCOL=beq)) )
17994 ora_tt00_ORCL
17996 ora_arc0_ORCL
17998 ora_tt01_ORCL
18000 ora_arc1_ORCL
18002 ora_arc2_ORCL
18004 ora_arc3_ORCL
18006 ora_tt02_ORCL
18008 ora_aqpc_ORCL
18012 ora_p000_ORCL
18014 ora_p001_ORCL
18016 ora_p002_ORCL
18018 ora_p003_ORCL
18020 ora_p004_ORCL
18022 ora_p005_ORCL
18024 ora_p006_ORCL
18026 ora_p007_ORCL
18028 ora_cjq0_ORCL
18106 ora_s001_ORCL
18182 ora_qm02_ORCL
18186 ora_q002_ORCL
18188 ora_q003_ORCL
18252 ora_m000_ORCL
SQL>

```

6. Explore the PDBs. They are all in MOUNTED state by default.

```

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

CON_ID NAME          OPEN_MODE
----- -----
 2  PDB$SEED        READ ONLY
 3  PDB1            MOUNTED
 4  TOYS_ROOT       MOUNTED
 5  ROBOTS          MOUNTED
 6  DOLLS           MOUNTED

SQL>

```

7. Open all PDBs.

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

```
Pluggable database altered.
```

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

CON_ID	NAME	OPEN_MODE
2	PDB\$SEED	READ ONLY
3	PDB1	READ WRITE
4	TOYS_ROOT	READ WRITE
5	ROBOTS	READ WRITE
6	DOLLS	READ WRITE

```
SQL>
```

8. Connect to any of the PDBs in your ORCL, except PDB\$SEED.

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

```
Enter password: password
```

```
Connected.
```

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

CON_ID	NAME	OPEN_MODE
5	ROBOTS	READ WRITE

```
SQL>
```

9. Display the context of the PDB you are connected to.

```
SQL> SHOW CON_NAME
```

```
CON_NAME
```

```
-----
```

```
ROBOTS
```

```
SQL> EXIT
```

```
$
```

## Practice 5-2: Opening and Closing PDBs

### Overview

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

### Tasks

- Before starting the practice, execute the `$HOME/labs/mgt/setup_apps.sh`, shell script. The script re-creates the `toys_root` and `hr_root` application containers and finally closes the `hr_root` application container.

```
$ $HOME/labs/mgt/setup_apps.sh
...
$
```

- 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> SHOW PDBS

CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
  2 PDB$SEED           READ ONLY NO
  3 PDB1               READ WRITE NO
  4 TOYS_ROOT          READ WRITE NO
  5 ROBOTS             READ WRITE NO
  6 DOLLS              READ WRITE NO
 10 HR_ROOT            MOUNTED
 11 OPERATIONS         MOUNTED
 14 RESEARCH           MOUNTED

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>
```

*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 PDB1             READ WRITE  NO
  4 TOYS_ROOT        READ WRITE  NO
  5 ROBOTS           READ WRITE  NO
  6 DOLLS            READ WRITE  NO
 10 HR_ROOT          READ WRITE  NO
 11 OPERATIONS      READ WRITE  NO
 14 RESEARCH         MOUNTED

SQL>
```

- 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        10 YES     NO      NO

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 = 10;
2   3   4   5   6
NAME      CON_ID APP_ROOT APP_Seed APP_PDB APP_ROOT_CONID
-----
OPERATIONS      11 NO      NO      YES      10
RESEARCH        14 NO      NO      YES      10

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 PDB1                READ WRITE NO
4 TOYS_ROOT          READ WRITE NO
5 ROBOTS              READ WRITE NO
6 DOLLS               READ WRITE NO
10 HR_ROOT            READ WRITE NO
11 OPERATIONS         READ WRITE NO
14 RESEARCH           READ WRITE NO
SQL>
```

*Q/ 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 PDB1                READ WRITE NO
4 TOYS_ROOT          READ WRITE NO
5 ROBOTS              READ WRITE NO
6 DOLLS               READ WRITE NO
```

```

10 HR_ROOT          MOUNTED
11 OPERATIONS       MOUNTED
14 RESEARCH         MOUNTED
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 PDB1         READ WRITE NO
  4 TOYS_ROOT    READ WRITE NO
  5 ROBOTS       READ WRITE NO
  6 DOLLS        READ WRITE NO
 10 HR_ROOT      READ WRITE NO
 11 OPERATIONS   MOUNTED
 14 RESEARCH     MOUNTED
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 / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE ALL CLOSE;

Pluggable database altered.

```

```
SQL> ALTER PLUGGABLE DATABASE ALL EXCEPT research OPEN;
```

Pluggable database altered.

```
SQL> SHOW PDBS
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
2	PDB\$SEED	READ ONLY	NO
3	PDB1	READ WRITE	NO
4	TOYS_ROOT	READ WRITE	NO
5	ROBOTS	READ WRITE	NO
6	DOLLS	READ WRITE	NO
10	HR_ROOT	READ WRITE	NO
11	OPERATIONS	READ WRITE	NO
14	RESEARCH	MOUNTED	

```
SQL> ALTER PLUGGABLE DATABASE ALL SAVE STATE;
```

Pluggable database altered.

```
SQL> SELECT con_name, state FROM CDB_PDB_SAVED_STATES;
```

CON_NAME	STATE
PDB1	OPEN
OPERATIONS	OPEN
ROBOTS	OPEN
TOYS_ROOT	OPEN
DOLLS	OPEN
HR_ROOT	OPEN

6 rows selected.

```
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
2	PDB\$SEED	READ ONLY	NO
3	PDB1	READ WRITE	NO
4	TOYS_ROOT	READ WRITE	NO
5	ROBOTS	READ WRITE	NO
6	DOLLS	READ WRITE	NO
10	HR_ROOT	READ WRITE	NO
11	OPERATIONS	READ WRITE	YES
14	RESEARCH	MOUNTED	

```
SQL> EXIT
$
```

5. Clean up application containers by executing the \$HOME/labs/mgt/cleanup\_apps.sh script.

```
$ $HOME/labs/mgt/cleanup_apps.sh
...
$
```

## Practice 5-3: Renaming a PDB

### Overview

In this practice, you will change the open mode of PDBs for specific operations.

### Tasks

Rename the global database name for `pdb1` to `PDB1_ORCL` in `ORCL`. For this purpose, you must open the PDB in `RESTRICTED` mode.

1. Execute the `$HOME/labs/admin/create_PDB1.sh` shell script to re-create `PDB1` in `ORCL`.

```
$ $HOME/labs/admin/create_PDB1.sh
...
$
```

2. Connect to `PDB1`.

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

SQL> SHOW PDBS

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

3. Change the global database name for `PDB1` to `PDB1_ORCL`.

```
SQL> ALTER PLUGGABLE DATABASE RENAME GLOBAL_NAME TO pdb1_ORCL;
alter pluggable database RENAME GLOBAL_NAME TO pdb1_ORCL
*
ERROR at line 1:
ORA-65046: operation not allowed from outside a pluggable
database

SQL> CONNECT sys@pdb1 AS SYSDBA
Enter password: password
Connected.
SQL> ALTER PLUGGABLE DATABASE RENAME GLOBAL_NAME TO pdb1_ORCL;
ALTER PLUGGABLE DATABASE RENAME GLOBAL_NAME TO pdb1_ORCL
*
ERROR at line 1:
```

```
ORA-65045: pluggable database not in a restricted mode
SQL>
```

4. Close PDB1.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE;
Pluggable database altered.
SQL>
```

5. Open PDB1 in restricted mode.

```
SQL> ALTER PLUGGABLE DATABASE OPEN RESTRICTED;
Pluggable database altered.
SQL>
```

```
SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;
CON_ID NAME          OPEN_MODE  RES
----- -----
 3  PDB1           READ WRITE YES
SQL>
```

6. Change the global database name for PDB1 to PDB1\_ORCL.

```
SQL> ALTER PLUGGABLE DATABASE RENAME GLOBAL_NAME TO pdb1_ORCL;
Pluggable database altered.
SQL>
```

```
SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;
CON_ID NAME          OPEN_MODE  RES
----- -----
 3  PDB1_ORCL       READ WRITE YES
SQL>
```

7. Open PDB1\_ORCL.

```
SQL> ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE;
Pluggable database altered.
```

```
SQL> ALTER PLUGGABLE DATABASE OPEN;  
  
Pluggable database altered.  
  
SQL>
```

8. Check PDB1 is in READ WRITE mode.

```
SQL> SELECT con_id, name, open_mode, restricted FROM v$pdbs;  
  
CON_ID NAME          OPEN_MODE  RES  
----- ----  
      3 PDB1_ORCL      READ WRITE NO  
  
SQL>
```

## Practice 5-4: Setting Parameter Values for PDBs

### Overview

In this practice, you will discover the impact of instance parameter changes on PDBs.

### Tasks

- Not all instance parameters are modifiable at the PDB level. A modifiable one, `OPTIMIZER_USE_SQL_PLAN_BASELINES`, has been chosen for the example so as to show how instance parameters behave at PDB and CDB level. Connect to `cdb2`.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT ispdb_modifiable FROM v$parameter
      WHERE name = 'optimizer_use_sql_plan_baselines';
2
ISPDB
-----
TRUE

SQL>
```

- Check the current value of instance parameter `OPTIMIZER_USE_SQL_PLAN_BASELINES`.

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

SQL>
```

- Connect to `PDB1_ORCL` in `ORCL` and check the current value of the same instance parameter `OPTIMIZER_USE_SQL_PLAN_BASELINES`.

```
SQL> CONNECT sys@PDB1_ORCL AS SYSDBA
Enter password: password
Connected.
SQL> SHOW PARAMETER optimizer_use_sql_plan_baselines
NAME                           TYPE        VALUE
-----
optimizer_use_sql_plan_baselines    boolean     TRUE

SQL>
```

4. Change the instance parameter value to FALSE in PDB1\_ORCL.

```
SQL> ALTER SYSTEM SET optimizer_use_sql_plan_baselines=FALSE
SCOPE=BOTH;

System altered.

SQL>
SQL> SHOW PARAMETER optimizer_use_sql_plan_baselines
NAME                      TYPE        VALUE
-----
optimizer_use_sql_plan_baselines    boolean      FALSE
SQL>
```

5. Create another PDB and check the instance parameter value in this new PDB of the same CDB.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> !mkdir /u02/app/oracle/oradata/ORCL/test

SQL> CREATE PLUGGABLE DATABASE test ADMIN USER admin
          IDENTIFIED BY password ROLES=(CONNECT)
          CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/test';
2   3   4
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE test OPEN;

Pluggable database altered.

SQL> CONNECT sys@test AS SYSDBA
Enter password: password
Connected.
SQL> SHOW PARAMETER optimizer_use_sql_plan_baselines
NAME                      TYPE        VALUE
-----
optimizer_use_sql_plan_baselines    boolean      TRUE
SQL>
```

6. Close and open PDB1\_ORCL.

```
SQL> CONNECT sys@PDB1_ORCL AS SYSDBA
Enter password: password Connected.
SQL> ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE;
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> SHOW PARAMETER optimizer_use_sql_plan_baselines
NAME                      TYPE        VALUE
-----
optimizer_use_sql_plan_baselines    boolean      FALSE

SQL>
```

7. Check the instance parameter value after CDB shutdown/startup in both the CDB root and PDBs.

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

SQL>
SQL> STARTUP
ORACLE instance started.

Total System Global Area  4697620480 bytes
Fixed Size                  2923760 bytes
Variable Size                989856528 bytes
Database Buffers            3690987520 bytes
Redo Buffers                 13852672 bytes
Database mounted.
Database opened.

SQL> SELECT con_id, value FROM v$system_parameter
      WHERE name ='optimizer_use_sql_plan_baselines';
2
CON_ID  VALUE
-----
          0  TRUE

SQL>
```

*Q/ Which value does the value for the CON\_ID 0 represent?*

**A/ This is the default value for all containers including the CDB root, which did not set a different value.**

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> SELECT con_id, value FROM v$system_parameter
  WHERE name ='optimizer_use_sql_plan_baselines';
2
  CON_ID VALUE
-----
    0  TRUE
    3  FALSE

SQL> EXIT
$
```

*Q2/ Why are there only two rows whereas there are three containers (the CDB root, PDB1\_ORCL, and TEST)?*

**A2/ The first value displays the default value for all containers and the second row displays the con\_id and value of the PDB, which sets a different value (PDB1\_ORCL).**

## Practice 5-5: Renaming PDB Services

---

### Overview

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

### Tasks

1. For testing purposes, clone TEST into CDB18 from the source ORCL and open the new PDB.
  - a. If TEST does not exist in ORCL, execute the \$HOME/labs/mgt/setup\_test.sh shell script to create the PDB.

```
$ $HOME/labs/mgt/setup_test.sh
...
$
```

- b. Because the SERVICE\_NAME\_CONVERT feature works only on managed services and not the default service, create a new service for test in the source ORCL.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus sys@test AS SYSDBA
Enter password: password
Connected.
SQL> EXEC DBMS_SERVICE.delete_SERVICE('test_node1')
BEGIN DBMS_SERVICE.delete_SERVICE('test_node1'); END;
*
ERROR at line 1:
ORA-44304: service test_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('test_node1','test_node1')

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_SERVICE.START_SERVICE('test_node1')

PL/SQL procedure successfully completed.

SQL> SELECT name, network_name FROM dba_services;
```

```

NAME          NETWORK_NAME
-----
TEST          TEST
test_node1    test_node1

SQL> ALTER PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN READ ONLY;

Pluggable database altered.

SQL> EXIT
$
```

- c. Clone TEST into CDB18 from the source ORCL and open the cloned PDB. Restart CDB18 (you shut it down in a previous practice).

```

$ mkdir /u02/app/oracle/oradata/CDB18/test
$
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
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 1426062800 bytes
Fixed Size                  8895952 bytes
Variable Size                486539264 bytes
Database Buffers             922746880 bytes
Redo Buffers                 7880704 bytes
Database mounted.
Database opened.

SQL> DROP DATABASE LINK link_orcl;
DROP DATABASE LINK link_orcl
*
ERROR at line 1:
ORA-02024: database link not found
```

```
SQL> CREATE DATABASE LINK link_orcl
      CONNECT TO system IDENTIFIED BY password
      USING 'ORCL';
2     3
Database link created.

SQL> ALTER SESSION SET
      db_create_file_dest='/u02/app/oracle/oradata/CDB18/test';
2
Session altered.

SQL> CREATE PLUGGABLE DATABASE test FROM test@link_orcl;

Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE test 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 CDB18?**

```
$ lsnrctl status
...
Service "test" has 2 instance(s).
  Instance "CDB18", status READY, has 1 handler(s) for this
service...
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
...
$ EXIT
SQL>
```

A/ Rename the test service to another name during the PDB creation. First drop the PDB to re-create it.

```
SQL> ALTER PLUGGABLE DATABASE test CLOSE;

Pluggable database altered.

SQL> DROP PLUGGABLE DATABASE test INCLUDING DATAFILES;

Pluggable database dropped.

SQL> CREATE PLUGGABLE DATABASE test FROM test@link_orcl
      SERVICE_NAME_CONVERT = ('test_node1', 'test_node2');

2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE test OPEN
      SERVICES = ('test_node2');

2
Pluggable database altered.

SQL> HOST
$ lsnrctl status
...
Service "test" has 2 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this
service...
  Instance "CDB18", status READY, has 1 handler(s) for this
service...
Service "test_node2" has 1 instance(s).
  Instance "CDB18", status READY, has 1 handler(s) for this
service...
...
$ exit
SQL> CONNECT sys@test_node2 AS SYSDBA
Enter password: password
Connected.
SQL> SHOW CON_NAME
```

```
CON_NAME  
-----  
TEST  
SQL> EXIT  
$
```

2. Drop test from ORCL by using the HOME/labs/mgt/cleanup\_test.sh script.

```
$ $HOME/labs/mgt/cleanup_test.sh  
...  
$
```

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.

## **Practices for Lesson 6: Storage**

## Practices for Lesson 6: Overview

---

### Practices Overview

In this practice, you will manage the tablespaces in the CDB root and in the PDBs.

In case you need to **re-create the ORCL CDB and its PDB1 PDB**, use the /home/oracle/labs/admin/recreate\_ORCL.sh shell script.

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

## Practice 6-1: Managing Permanent and Temporary Tablespaces

---

### Overview

In this practice, you will manage the permanent and temporary tablespaces in the CDB root and in the PDBs.

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. Then execute the `$HOME/labs/storage/glogin_6.sh` and `$HOME/labs/admin/create_PDB1.sh` shell scripts. The first script sets formatting for all columns selected in queries, and the second script re-creates pdb1 in ORCL in case it has been dropped.

```
$ $HOME/labs/storage/glogin_6.sh
...
$ $HOME/labs/admin/create_PDB1.sh
...
$
```

3. View permanent and temporary tablespaces properties in ORCL.

```
$ . oraenv
ORACLE_SID = [CDB18] ? 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 LIKE 'DEFAULT_%TABLE%';
2      3
PROPERTY_NAME          PROPERTY_VALUE
-----
DEFAULT_PERMANENT_TABLESPACE USERS
DEFAULT_TEMP_TABLESPACE    TEMP

SQL> SELECT tablespace_name, CON_ID from CDB_TABLESPACES;

TABLESPACE_NAME CON_ID
-----
```

```
SYSTEM          1
SYSAUX         1
UNDOTBS1       1
TEMP           1
USERS          1
SYSTEM          5
SYSAUX         5
UNDOTBS1       5
TEMP           5
```

9 rows selected.

```
SQL> SELECT tablespace_name, CON_ID from CDB_TABLESPACES
      WHERE TABLESPACE_NAME LIKE 'TEMP%';
2
TABLESPACE_NAME CON_ID
-----
TEMP           5
TEMP           1

SQL>
```

4. Create a permanent tablespace CDATA in the CDB root.

```
SQL> CREATE TABLESPACE CDATA
      DATAFILE '/u02/app/oracle/oradata/ORCL/cdata_01.dbf'
      SIZE 10M;
2     3
Tablespace created.

SQL> SELECT tablespace_name, CON_ID from CDB_TABLESPACES
      WHERE TABLESPACE_NAME = 'CDATA';
2
TABLESPACE_NAME CON_ID
-----
CDATA           1

SQL>
```

5. Make the CDATA tablespace the default tablespace in the root container.

```
SQL> ALTER DATABASE DEFAULT TABLESPACE CDATA;
Database altered.

SQL> SELECT property_name, property_value
```

```

    FROM  database_properties
 WHERE  property_name LIKE 'DEFAULT_%TABLE%';
2      3
PROPERTY_NAME          PROPERTY_VALUE
-----
DEFAULT_PERMANENT_TABLESPACE CDATA
DEFAULT_TEMP_TABLESPACE     TEMP

SQL>

```

6. Create a permanent tablespace, LDATA, in PDB1.

```

SQL> CONNECT system@PDB1
Enter password: password
Connected.
SQL> CREATE TABLESPACE ldata DATAFILE
      '/u02/app/oracle/oradata/ORCL/pdb1/ldata_01.dbf'
      SIZE 10M;
2      3
Tablespace created.

SQL>

```

7. Make the LDATA tablespace the default tablespace in the PDB2 container.

```

SQL> ALTER PLUGGABLE DATABASE DEFAULT TABLESPACE ldata;

Pluggable database altered.

SQL> SELECT property_name, property_value
      FROM  database_properties
 WHERE  property_name LIKE 'DEFAULT_%TABLE%';
2      3
PROPERTY_NAME          PROPERTY_VALUE
-----
DEFAULT_PERMANENT_TABLESPACE LDATA
DEFAULT_TEMP_TABLESPACE     TEMP

SQL>

```

8. Create a temporary tablespace in the CDB root.

```

SQL> CONNECT system
Enter password: password
Connected.
SQL> CREATE TEMPORARY TABLESPACE TEMP_ROOT
      TEMPFILE '/u02/app/oracle/oradata/ORCL/temproot_01.dbf'

```

```
SIZE 500M;  
2      3  
Tablespace created.  
  
SQL>
```

9. Make TEMP\_ROOT the default temporary tablespace in the CDB root.

```
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE TEMP_ROOT;  
  
Database altered.  
  
SQL> SELECT property_name, property_value  
      FROM database_properties  
     WHERE property_name LIKE 'DEFAULT_%TABLE%';  
2      3  
PROPERTY_NAME          PROPERTY_VALUE  
-----  
DEFAULT_PERMANENT_TABLESPACE CDATA  
DEFAULT_TEMP_TABLESPACE      TEMP_ROOT  
  
SQL>
```

10. Create a temporary tablespace TEMP\_PDB1 in PDB1.

```
SQL> CONNECT system@PDB1  
Enter password: password  
Connected.  
SQL> CREATE TEMPORARY TABLESPACE temp_pdb1  
      TEMPFILE  
      '/u02/app/oracle/oradata/ORCL/pdb1/temp_pdb1_01.dbf'  
      SIZE 100M;  
2      3  
Tablespace created.  
  
SQL>
```

11. Make TEMP\_PDB1 the default temporary tablespace in PDB1.

```
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp_pdb1;  
  
Database altered.  
  
SQL> SELECT property_name, property_value  
      FROM database_properties  
     WHERE property_name LIKE 'DEFAULT_%TABLE%';  
2      3
```

PROPERTY_NAME	PROPERTY_VALUE
<hr/>	
DEFAULT_TEMP_TABLESPACE	<b>TEMP_PDB1</b>
DEFAULT_PERMANENT_TABLESPACE	LDATA
SQL>	

Note that you could also use ALTER PLUGGABLE DATABASE command.

12. Create a temporary tablespace MYTEMP in PDB1 .

```
SQL> CREATE TEMPORARY TABLESPACE my_temp TEMPFILE
      '/u02/app/oracle/oradata/ORCL/pdb1/mytemp_01.dbf'
      SIZE 10M;
2      3
Tablespace created.

SQL>
```

13. Display default tablespaces of another PDB in ORCL. If there is not any other PDB in the CDB, create a new one using the \$HOME/labs/storage/setup\_newpdb.sql SQL script.

```
SQL> CONNECT system@PDB1
Enter password: password

Connected.
SQL> @$HOME/labs/storage/setup_newpdb.sql
...
PROPERTY_NAME          PROPERTY_VALUE
-----
DEFAULT_PERMANENT_TABLESPACE  SYSTEM
DEFAULT_TEMP_TABLESPACE       TEMP

SQL>
```

14. Manage default permanent and temporary tablespaces of users.

- a. Create a common user C##U.

```
SQL> CONNECT system
Enter password: password
Connected.
SQL> CREATE USER c##u IDENTIFIED BY password;

User created.

SQL>
```

- b. View the default tablespace and temporary tablespace assignment for user CU in all containers.

```
SQL> SELECT username, default_tablespace,
      temporary_tablespace, con_id
    FROM cdb_users
   WHERE username = 'C##U';
2   3   4
USERNAME      DEFAULT_TABLESPACE TEMPORARY_TABLESPACE CON_ID
-----
C##U          CDATA             TEMP_ROOT           1
C##U          SYSTEM            TEMP               3
C##U          LDATA             TEMP_PDB1         5
SQL>
```

- c. Create a local user LU in PDB1.

```
SQL> CONNECT system@PDB1
Enter password: password
Connected.
SQL> CREATE USER lu IDENTIFIED BY password;

User created.

SQL>
```

- d. View the default tablespace and temporary tablespace assignment for user LU.

```
SQL> SELECT username, default_tablespace, temporary_tablespace
      FROM dba_users
     WHERE username = 'LU';
2   3
USERNAME      DEFAULT_TABLESPACE TEMPORARY_TABLESPACE
-----
LU           LDATA             TEMP_PDB1

SQL>
```

- e. Change the temporary tablespace assignment for user LU to MY\_TEMP in PDB2.

```
SQL> ALTER USER lu TEMPORARY TABLESPACE my_temp;

User altered.

SQL>
```

- f. View the default temporary tablespace assignment for user LU.

```
SQL> SELECT username, default_tablespace, temporary_tablespace
      FROM dba_users
     WHERE username = 'LU';
2      3
USERNAME      DEFAULT_TABLESPACE TEMPORARY_TABLESPACE
-----
LU            LDATA                  MY_TEMP
SQL>
```

## Practice 6-2: Managing UNDO Tablespaces

### Overview

In this practice, you manage UNDO tablespaces in PDBs.

1. Display the UNDO tablespaces used in the CDB.

```
SQL> CONNECT system
Enter password: password
Connected.

SQL> SELECT file#, ts.name, ts.ts#, ts.con_id
      FROM v$logfile d, v$tablespace ts
     WHERE d.ts#=ts.ts#
       AND d.con_id=ts.con_id
       AND ts.name like 'UNDOTBS%';

2      3      4      5
FILE# NAME          TS# CON_ID
----- -----
        4 UNDOTBS1          2      1
        8 UNDOTBS1          2      2
      159 UNDOTBS1          2      5
      164 UNDOTBS1          2      3

SQL>
```

*Q/ According to the list of UNDO tablespaces, what can you conclude about the UNDO mode used?*

**A/ Since there is an UNDO tablespace in each container, the UNDO mode used is the local UNDO mode.**

*Q2/ Why should you use the UNDO local mode?*

**A2/ The UNDO local mode is useful for hot cloning, PDB relocation, and PDB proxying.**

2. Verify that the UNDO mode is local.

```
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> EXIT  
$
```

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.

## **Practices for Lesson 7: Security**

## Practices for Lesson 7: Overview

---

### Overview

In this practice, you will manage the major aspects of data security.

- Common and local users, privileges, and roles
- Access to common and local objects
- Static and dynamic PDB lockdown profiles
- Auditing
- Protection by Database Vault
- TDE encryption with PDB keystores
- Plugging encrypted plugged PDB

## Practice 7-1: Managing Common and Local Users, Privileges, and Roles

---

### Overview

In this practice, you will manage the common and local users in CDB and PDBs, regular and application PDBs, and the common and local privileges and roles granted in CDB and PDBs, regular and application PDBs.

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. Then execute the `$HOME/labs/sec/glogin_7.sh` and `$HOME/labs/admin/create_PDB1.sh` shell scripts. The first script sets formatting for all columns selected in queries and the second script re-creates PDB1 in ORCL.

```
$ $HOME/labs/sec/glogin_7.sh
...
$ $HOME/labs/admin/create_PDB1.sh
...
$
```

3. View all common and local users in ORCL.

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

SQL> SELECT username, common, con_id FROM cdb_users;

USERNAME          COMMON CON_ID
-----  -----  -----
SYS              YES      1
SYSTEM           YES      1
OUTLN            YES      1
LBACSYS          YES      1
XS$NULL          YES      1
SYS$UMF          YES      1
OJVMSYS          YES      1
SI_INFORMTN_SCHEMA YES      1
```

GGSYS	YES	1
ORDDATA	YES	1
OLAPSYS	YES	1
ANONYMOUS	YES	1
ORDSYS	YES	1
DVF	YES	1
DBSNMP	YES	1
...		
C##U	YES	5
OJVMSYS	YES	5
PDB_ADMIN	NO	5
AUDSYS	YES	5
DIP	YES	5
LBACSYS	YES	5
SYSKM	YES	5
OUTLN	YES	5
ORACLE_OCM	YES	5
SYS\$UMF	YES	5
SYSDG	YES	5
DBSNMP	YES	5
APPQOSSYS	YES	5
DBSFWUSER	YES	5
GGSYS	YES	5
ANONYMOUS	YES	5
CTXSYS	YES	5
DVF	YES	5
SI_INFORMTN_SCHEMA	YES	5
DVSYS	YES	5
GSMADMIN_INTERNAL	YES	5
ORDPLUGINS	YES	5
MDSYS	YES	5
OLAPSYS	YES	5
ORDDATA	YES	5
XDB	YES	5
WMSYS	YES	5
ORDSYS	YES	5

73 rows selected.

SQL>

```
SQL> SELECT username, common, con_id FROM cdb_users
      WHERE username = 'SYSTEM';
```

```
2
USERNAME          COMMON CON_ID
-----
SYSTEM            YES      1
SYSTEM            YES      5

SQL> SELECT distinct username FROM cdb_users
      WHERE common='YES';

2
USERNAME
-----
SI_INFORMTN_SCHEMA
OUTLN
ORDDATA
ANONYMOUS
GSMCATUSER
SYSDG
ORDSYS
SYSBACKUP
GSMADMIN_INTERNAL
C##U
SYSRAC
XDB
DBSFWUSER
ORDPLUGINS
DV$SYS
APPQOSSYS
GSMUSER
MDDATA
GG$SYS
OLAPSYS
LBACSYS
XS$NULL
SYS
ORACLE_OCM
SYSKM
REMOTE_SCHEDULER_AGENT
SYS$UMF
DVF
DBSNMP
DIP
MD$SYS
```

```
SYSTEM
WMSYS
CTXSYS
AUDSYS
OJVMSYS

36 rows selected.

SQL> SELECT username, con_id FROM cdb_users
  WHERE common='NO';
2
USERNAME          CON_ID
-----
PDB_ADMIN           5

SQL>
```

4. Create a common user C##\_USER.

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

SQL>
```

5. View the new common user C##\_USER.

```
SQL> SELECT distinct username, con_id FROM cdb_users
  WHERE username='C##_USER';
2
USERNAME          CON_ID
-----
C##_USER            1
C##_USER            5

SQL>
```

**Notice that the common user exists in each container.**

6. Grant CREATE SESSION as a common privilege.

```
SQL> GRANT CREATE SESSION TO c##_user CONTAINER=ALL;

Grant succeeded.

SQL>
```

7. Connect to the CDB root and PDB1 as c##\_user.

```
SQL> CONNECT c##_user@PDB1
Enter password: password
Connected.
SQL> CONNECT c##_user@ORCL
Enter password: password
Connected.
SQL>
```

8. Create a local user LOCAL\_USER in the CDB root.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> CREATE USER local_user IDENTIFIED BY password
      CONTAINER=CURRENT;
create user local_user identified by password
*
ERROR at line 1:
ORA-65049: creation of local user or role is not allowed in this
container.

SQL>
```

**Notice that no local user is authorized in the CDB root.**

9. Create a common user and grant CREATE SESSION as a local privilege.

- a. Create the user and grant the privilege.

```
SQL> CREATE USER c##_user2 IDENTIFIED BY password
      CONTAINER=ALL;
2
User created.

SQL> GRANT CREATE SESSION TO c##_user2;

Grant succeeded.
```

```
SQL> CONNECT c##_user2@PDB1
ERROR:
ORA-01045: user C##_USER2 lacks CREATE SESSION privilege; logon
denied

SQL> CONNECT c##_user2@ORCL
```

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

```
SQL>
```

**Note that even though the user is a common user, the privilege is granted locally in the CDB root. That is why the common user can connect to the CDB root only where he is granted the CREATE SESSION privilege.**

- Drop the common user.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> DROP USER c##_user2;

User dropped.

SQL>
```

- Create a local user, LOCAL\_USER\_PDB1, in the regular PDB1.

- View all users of PDB1.

```
SQL> CONNECT sys@PDB1 AS SYSDBA
Enter password: password
Connected.
SQL> SELECT username, common, con_id FROM cdb_users
      ORDER BY 1;
2
USERNAME          COMMON CON_ID
-----
ANONYMOUS         YES    5
APPQOSSYS        YES    5
AUDSYS           YES    5
C##U              YES    5
C##_USER          YES    5
CTXSYS            YES    5
DBSFWUSER         YES    5
DBSNMP             YES   5
DIP                YES   5
DVF                YES   5
DVSYS              YES   5
GGSYS              YES   5
GSMADMIN_INTERNAL YES   5
GSMCATUSER         YES   5
GSMUSER             YES   5
LBACSYS            YES   5
MDDATA              YES   5
MDSYS              YES   5
OJVMSYS            YES   5
OLAPSYS            YES   5
ORACLE_OCM         YES   5
```

ORDDATA	YES	5
ORDPLUGINS	YES	5
ORDSYS	YES	5
OUTLN	YES	5
PDB_ADMIN	NO	5
REMOTE_SCHEDULER_AGENT	YES	5
SI_INFORMTN_SCHEMA	YES	5
SYS	YES	5
SYS\$UMF	YES	5
SYSBACKUP	YES	5
SYSDG	YES	5
SYSKM	YES	5
SYSRAC	YES	5
SYSTEM	YES	5
WMSYS	YES	5
XDB	YES	5
XS\$NULL	YES	5

38 rows selected.

SQL>

**Notice that you view all common and local users of the current PDB.**

SQL> `SELECT username, common FROM dba_users;`

**Notice that you view the same list.**

- b. Attempt to create the C##\_USER2\_PDB1 common user in the regular PDB1.

```
SQL> CREATE USER c##_user2_pdb1 IDENTIFIED BY password
      CONTAINER=ALL;
create user c##_user2_pdb1 identified by password CONTAINER=ALL
*
ERROR at line 1:
ORA-65050: Common DDLs only allowed in CDB$ROOT

SQL>
```

**Notice that no common user can be created except from the CDB root.**

- c. Create the local user LOCAL\_USER\_PDB1 in the regular PDB1.

```
SQL> CREATE USER local_user_pdb1 IDENTIFIED BY password
      CONTAINER=CURRENT;
2
User created.
```

SQL> SELECT username, common, con_id FROM cdb_users ORDER BY 1;		
2	USERNAME	COMMON CON_ID
	ANONYMOUS	YES 5
	APPQOSSYS	YES 5
	AUDSYS	YES 5
	C##U	YES 5
	C##_USER	YES 5
	CTXSYS	YES 5
	DBSFWUSER	YES 5
	DBSNMP	YES 5
	DIP	YES 5
	DVF	YES 5
	DVSYS	YES 5
	GGSYS	YES 5
	GSMADMIN_INTERNAL	YES 5
	GSMCATUSER	YES 5
	GSMUSER	YES 5
	LBACSYS	YES 5
	<b>LOCAL_USER_PDB1</b>	<b>NO 5</b>
	MDDATA	YES 5
	MDSYS	YES 5
	OJVMSYS	YES 5
	OLAPSYS	YES 5
	ORACLE_OCM	YES 5
	ORDDATA	YES 5
	ORDPLUGINS	YES 5
	ORDSYS	YES 5
	OUTLN	YES 5
	PDB_ADMIN	NO 5
	REMOTE_SCHEDULER_AGENT	YES 5
	SI_INFORMTN_SCHEMA	YES 5
	SYS	YES 5
	SYS\$UMF	YES 5
	SYSBACKUP	YES 5
	SYSDG	YES 5
	SYSKM	YES 5
	SYSRAC	YES 5
	SYSTEM	YES 5
	WMSYS	YES 5
	XDB	YES 5

```
X$NULL          YES      5

39 rows selected.

SQL> GRANT create session TO local_user_pdb1;

Grant succeeded.

SQL>
```

- d. Connect to the regular PDB1 as LOCAL\_USER\_PDB1.

```
SQL> CONNECT local_user_pdb1@PDB1
Enter password: password
Connected.

SQL>
```

- e. Connect to the CDB root as LOCAL\_USER\_PDB1.

```
SQL> CONNECT local_user_PDB1@ORCL
ERROR:
ORA-01017: invalid username/password; logon denied

Warning: You are no longer connected to ORACLE.

SQL>
```

***Notice that it fails because LOCAL\_USER\_PDB1 does not exist in the CDB root.***

- f. List all common and local users from a regular PDB.

```
SQL> CONNECT sys@PDB1 AS SYSDBA
Enter password: password
Connected.

SQL> SELECT username, common, con_id FROM cdb_users
      ORDER BY 1;
      2
USERNAME          COMMON CON_ID
-----  -----  -----
ANONYMOUS        YES      5
APPQOSSYS       YES      5
AUDSYS          YES      5
C##U            YES      5
C##_USER        YES      5
CTXSYS          YES      5
DBSFWUSER       YES      5
DBSNMP          YES      5
DIP              YES      5
DVF              YES      5
```

DVSYST	YES	5
GGSYS	YES	5
GSMADMIN_INTERNAL	YES	5
GSMCATUSER	YES	5
GSMUSER	YES	5
LBACSYS	YES	5
LOCAL_USER_PDB1	NO	5
MDDATA	YES	5
MDSYS	YES	5
OJVMSYS	YES	5
OLAPSYS	YES	5
ORACLE_OCM	YES	5
ORDDATA	YES	5
ORDPLUGINS	YES	5
ORDSYS	YES	5
OUTLN	YES	5
PDB_ADMIN	NO	5
REMOTE_SCHEDULER_AGENT	YES	5
SI_INFORMTN_SCHEMA	YES	5
SYS	YES	5
SYS\$UMF	YES	5
SYSBACKUP	YES	5
SYSDG	YES	5
SYSKM	YES	5
SYSRAC	YES	5
SYSTEM	YES	5
WMSYS	YES	5
XDB	YES	5
XS\$NULL	YES	5

39 rows selected.

SQL> **EXIT**

\$

11. You learned that the same concept of commonality exists for users at the application container level. You will now manage the common and local users in application roots and application PDBs.

- a. First, execute the `$HOME/labs/sec/setup_toys_app.sh` shell scripts. The script creates the `toys_root` application root and the `robots` and `dolls` application PDBs in ORCL.

```
$ $HOME/labs/sec/setup_toys_app.sh  
...  
$
```

- b. 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 if it exists, and the two application PDBs, `robots` and `dolls`. 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.

```
$ sqlplus sys@toys_root AS SYSDBA  
Enter password: password  
Connected.  
SQL> SELECT username, common, p.pdb_name, p.con_id  
      FROM cdb_users u, cdb_pdbs p  
     WHERE u.con_id = p.con_id  
       AND username IN ('C##_USER', 'TOYS_OWNER')  
      ORDER BY 3;
```

2	3	4	5	COMMON	PDB_NAME	CON_ID
USERNAME						
TOYS_OWNER				YES	DOLLS	6
TOYS_OWNER				YES	ROBOTS	5
TOYS_OWNER				YES	TOYS_ROOT	4
C##_USER				YES	DOLLS	6
C##_USER				YES	ROBOTS	5
C##_USER				YES	TOYS_ROOT	4
SQL>						

*Q/ Can the toys\_owner user be a common user at the CDB level?*

**A/ 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, pdb_name
      FROM cdb_users u, cdb_pdbs p
     WHERE u.con_id = p.con_id
       AND username = 'TOYS_OWNER'
      ORDER BY 4;
2      3      4      5
USERNAME                  COMMON INH PDB_NAME
-----
TOYS_OWNER                 YES    YES DOLLS
TOYS_OWNER                 YES    YES ROBOTS
TOYS_OWNER                 YES    NO   TOYS_ROOT
SQL>

```

**Q3/ Why does the `toys_owner` application common 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.**

- c. Create an application 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 password
      CONTAINER = ALL;
2

User created.

SQL> SELECT username, common, inherited, pdb_name
      FROM cdb_users u, cdb_pdbs p
     WHERE u.con_id = p.con_id
       AND username like 'TOYS_TEST%';
2   3   4
USERNAME          COMMON INH PDB_NAME
-----
TOYS_TEST          YES     NO    TOYS_ROOT

SQL>
```

- d. 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 the users created commonly in an application root outside an application and within an application.
- 1) 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 password
      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, pdb_name
      FROM cdb_users u, cdb_pdbs p
      WHERE u.con_id = p.con_id
      AND username like 'TOYS_TEST%';
2   3   4
USERNAME           COMMON INH PDB_NAME
-----
TOYS_TEST2          YES    NO  TOYS_ROOT
TOYS_TEST           YES    NO  TOYS_ROOT

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.*

- 2) 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.
- 3) Connect as toys\_test2 to toys\_root, then robots, and finally to dolls.

```

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

SQL> CONNECT toys_test2@dolls
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.**

- 4) Synchronize the application PDBs.

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

Pluggable database altered.

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

Pluggable database altered.

SQL>
```

- 5) Connect as toys\_test2 and then as toys\_test to robots and to dolls.

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

```
ERROR:  
ORA-01045: user TOYS_TEST2 lacks CREATE SESSION privilege; logon  
denied  
  
Warning: You are no longer connected to ORACLE.  
  
SQL> CONNECT toys_test@robots  
Enter password: password  
ERROR:  
ORA-01017: invalid username/password; logon denied  
  
Warning: You are no longer connected to ORACLE.  
SQL> CONNECT toys_test@dolls  
Enter password: password  
ERROR:  
ORA-01017: invalid username/password; logon denied  
  
Warning: You are no longer connected to ORACLE.  
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: password  
Connected.  
SQL> SELECT username, common, inherited, pdb_name  
      FROM cdb_users u, cdb_pdbs p  
     WHERE u.con_id = p.con_id  
       AND username like 'TOYS_TEST%' ORDER BY 1;  
2      3      4  
USERNAME          COMMON INH PDB_NAME  
-----  
TOYS_TEST         YES    NO   TOYS_ROOT  
TOYS_TEST2        YES    YES  ROBOTS  
TOYS_TEST2        YES    YES  DOLLS  
TOYS_TEST2        YES    NO   TOYS_ROOT  
  
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=3*), and not by replication from the CDB root. The *YES* value for the same users in *robots* and *do11s* means that the same common user *toys\_test2* was created in the application PDBs by replication from the *toys\_app* application root.

- e. Create a local user in the robots application PDB.

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

User created.

SQL> GRANT CREATE SESSION TO l_user;

Grant succeeded.

SQL> SELECT username, common, inherited, pdb_name
  FROM cdb_users u, cdb_pdbs p
 WHERE u.con_id = p.con_id
   AND common = 'NO';
 2      3      4
USERNAME                  COMMON INH PDB_NAME
-----
ADMIN                      NO     NO    ROBOTS
L_USER                     NO     NO    ROBOTS

SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: password
Connected.
SQL> SELECT username, common, inherited, pdb_name
  FROM cdb_users u, cdb_pdbs p
 WHERE u.con_id = p.con_id
   AND common = 'NO';
 2      3      4
USERNAME                  COMMON INH PDB_NAME
```

```

-----
ADMIN          NO    NO  TOYS_ROOT
ADMIN          NO    NO  ROBOTS
L_USER         NO    NO  ROBOTS
ADMIN          NO    NO  DOLLS

SQL>

```

**Observe that L\_USER exists only in the robots application PDB.**

12. You will now manage roles created as common or local and granted as common and or local in CDB and regular PDBs.

- a. List all predefined roles in CDB.

```

SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT role, common, con_id FROM cdb_roles ORDER BY role;

ROLE           COMMON CON_ID
-----
ADM_PARALLEL_EXECUTE_TASK   YES    1
ADM_PARALLEL_EXECUTE_TASK   YES    3
ADM_PARALLEL_EXECUTE_TASK   YES    6
ADM_PARALLEL_EXECUTE_TASK   YES    4
ADM_PARALLEL_EXECUTE_TASK   YES    5
APPLICATION_TRACE_VIEWER   YES    5
APPLICATION_TRACE_VIEWER   YES    1
APPLICATION_TRACE_VIEWER   YES    3
APPLICATION_TRACE_VIEWER   YES    6
APPLICATION_TRACE_VIEWER   YES    4
...
DBA             YES    1
DBA             YES    3
DBA             YES    6
DBA             YES    5
DBA             YES    4
...
XS_SESSION_ADMIN    YES    4
XS_SESSION_ADMIN    YES    3
XS_SESSION_ADMIN    YES    1
XS_SESSION_ADMIN    YES    6
XS_SESSION_ADMIN    YES    5

430 rows selected.

SQL>

```

You can view all common and local roles of the CDB root and PDBs.

- List all predefined roles in the CDB root.

```
SQL> SELECT role, common FROM dba_roles ORDER BY role;

ROLE                      COMMON
-----
ADM_PARALLEL_EXECUTE_TASK YES
APPLICATION_TRACE_VIEWER  YES
AQ_ADMINISTRATOR_ROLE     YES
AQ_USER_ROLE               YES
AUDIT_ADMIN                YES
AUDIT_VIEWER               YES
AUTENTICATEDUSER          YES
CAPTURE_ADMIN              YES
CDB_DBA                    YES
CONNECT                   YES
...
XS_CONNECT                 YES
XS_NAMESPACE_ADMIN          YES
XS_SESSION_ADMIN            YES

86 rows selected.

SQL>
```

**Notice that all roles of the CDB root are common; there cannot be any local roles in the CDB root.**

- Create a common C##\_ROLE in the CDB root.

```
SQL> CREATE ROLE c##_role CONTAINER = ALL;

Role created.

SQL>
```

- Create a local LOCAL\_ROLE in the CDB root.

```
SQL> CREATE ROLE local_role CONTAINER = CURRENT;
CREATE ROLE local_role CONTAINER = CURRENT
*
ERROR at line 1:
ORA-65049: creation of local user or role is not allowed in this
container.

SQL>
```

You get an error message because no local role is authorized in the CDB root.

- e. List all predefined roles in PDB PDB1.

ROLE	COMMON	CON_ID
CONNECT	YES	5
RESOURCE	YES	5
DBA	YES	5
PDB_DBA	YES	5
AUDIT_ADMIN	YES	5
AUDIT_VIEWER	YES	5
SELECT_CATALOG_ROLE	YES	5
EXECUTE_CATALOG_ROLE	YES	5
CAPTURE_ADMIN	YES	5
EXP_FULL_DATABASE	YES	5
IMP_FULL_DATABASE	YES	5
CDB_DBA	YES	5
APPLICATION_TRACE_VIEWER	YES	5
LOGSTDBY_ADMINISTRATOR	YES	5
DBFS_ROLE	YES	5
GSMUSER_ROLE	YES	5
AQ_ADMINISTRATOR_ROLE	YES	5
AQ_USER_ROLE	YES	5
DATAPUMP_EXP_FULL_DATABASE	YES	5
DATAPUMP_IMP_FULL_DATABASE	YES	5
ADM_PARALLEL_EXECUTE_TASK	YES	5
PROVISIONER	YES	5
XS_SESSION_ADMIN	YES	5
XS_NAMESPACE_ADMIN	YES	5
XS_CACHE_ADMIN	YES	5
XS_CONNECT	YES	5
GATHER_SYSTEM_STATISTICS	YES	5
GSM_POOLADMIN_ROLE	YES	5
OPTIMIZER_PROCESSING_RATE	YES	5
DBMS_MDX_INTERNAL	YES	5
RECOVERY_CATALOG_OWNER	YES	5
RECOVERY_CATALOG_OWNER_VPD	YES	5
RECOVERY_CATALOG_USER	YES	5

EM_EXPRESS_BASIC	YES	5
EM_EXPRESS_ALL	YES	5
SYSUMF_ROLE	YES	5
SCHEDULER_ADMIN	YES	5
HS_ADMIN_SELECT_ROLE	YES	5
HS_ADMIN_EXECUTE_ROLE	YES	5
HS_ADMIN_ROLE	YES	5
GLOBAL_AQ_USER_ROLE	YES	5
OEM_ADVISOR	YES	5
OEM_MONITOR	YES	5
GSMADMIN_ROLE	YES	5
EJBCLIENT	YES	5
GDS_CATALOG_SELECT	YES	5
GGSYS_ROLE	YES	5
XDBADMIN	YES	5
XDB_SET_INVOKER	YES	5
AUTHENTICATEDUSER	YES	5
XDB_WEBSERVICES	YES	5
XDB_WEBSERVICES_WITH_PUBLIC	YES	5
XDB_WEBSERVICES_OVER_HTTP	YES	5
SODA_APP	YES	5
DATAPATCH_ROLE	YES	5
WM_ADMIN_ROLE	YES	5
JAVAUSERPRIV	YES	5
JAVOIDPRIV	YES	5
JAVASYSPRIV	YES	5
JAVADEBUGPRIV	YES	5
JMXSERVER	YES	5
DBJAVASCRIPT	YES	5
JAVA_ADMIN	YES	5
CTXAPP	YES	5
ORDADMIN	YES	5
OLAP_XS_ADMIN	YES	5
OLAP_DBA	YES	5
OLAP_USER	YES	5
RDFCTX_ADMIN	YES	5
LBAC_DBA	YES	5
DV_SECANALYST	YES	5
DV_MONITOR	YES	5
DV_ADMIN	YES	5
DV_OWNER	YES	5
DV_ACCTMGR	YES	5

DV_PUBLIC	YES	5
DV_PATCH_ADMIN	YES	5
DV_STREAMS_ADMIN	YES	5
DV_GOLDENGATE_ADMIN	YES	5
DV_XSTREAM_ADMIN	YES	5
DV_GOLDENGATE_REDO_ACCESS	YES	5
DV_AUDIT_CLEANUP	YES	5
DV_DATAPUMP_NETWORK_LINK	YES	5
DV_POLICY_OWNER	YES	5
DV_REALM_RESOURCE	YES	5
DV_REALM_OWNER	YES	5
C##_ROLE	YES	5

87 rows selected.

SQL>

You can view all common and local roles of the PDB only.

SQL> **SELECT role, common FROM dba\_roles ORDER BY role;**

ROLE	COMMON
ADM_PARALLEL_EXECUTE_TASK	YES
APPLICATION_TRACE_VIEWER	YES
AQ_ADMINISTRATOR_ROLE	YES
AQ_USER_ROLE	YES
AUDIT_ADMIN	YES
AUDIT_VIEWER	YES
AUTHENTICATEDUSER	YES
C##_ROLE	YES
CAPTURE_ADMIN	YES
CDB_DBA	YES
CONNECT	YES
CTXAPP	YES
DATAPATCH_ROLE	YES
DATAPUMP_EXP_FULL_DATABASE	YES
DATAPUMP_IMP_FULL_DATABASE	YES
DBA	YES
DBFS_ROLE	YES
DBJAVASCRIPT	YES
DBMS_MDX_INTERNAL	YES
DV_ACCTMGR	YES
DV_ADMIN	YES

DV_AUDIT_CLEANUP	YES
DV_DATAPUMP_NETWORK_LINK	YES
DV_GOLDENGATE_ADMIN	YES
DV_GOLDENGATE_REDO_ACCESS	YES
DV_MONITOR	YES
DV_OWNER	YES
DV_PATCH_ADMIN	YES
DV_POLICY_OWNER	YES
DV_PUBLIC	YES
DV_REALM_OWNER	YES
DV_REALM_RESOURCE	YES
DV_SECANALYST	YES
DV_STREAMS_ADMIN	YES
DV_XSTREAM_ADMIN	YES
EJBCLIENT	YES
EM_EXPRESS_ALL	YES
EM_EXPRESS_BASIC	YES
EXECUTE_CATALOG_ROLE	YES
EXP_FULL_DATABASE	YES
GATHER_SYSTEM_STATISTICS	YES
GDS_CATALOG_SELECT	YES
GGSYS_ROLE	YES
GLOBAL_AQ_USER_ROLE	YES
GSMADMIN_ROLE	YES
GSMUSER_ROLE	YES
GSM_POOLADMIN_ROLE	YES
HS_ADMIN_EXECUTE_ROLE	YES
HS_ADMIN_ROLE	YES
HS_ADMIN_SELECT_ROLE	YES
IMP_FULL_DATABASE	YES
JAVADEBUGPRIV	YES
JAVAIDPRIV	YES
JAVASYSPRIV	YES
JAVAUSERPRIV	YES
JAVA_ADMIN	YES
JMXSERVER	YES
LBAC_DBA	YES
LOGSTDBY_ADMINISTRATOR	YES
OEM_ADVISOR	YES
OEM_MONITOR	YES
OLAP_DBA	YES
OLAP_USER	YES

```

OLAP_XS_ADMIN           YES
OPTIMIZER_PROCESSING_RATE YES
ORDADMIN                YES
PDB_DBA                  YES
PROVISIONER               YES
RDFCTX_ADMIN              YES
RECOVERY_CATALOG_OWNER    YES
RECOVERY_CATALOG_OWNER_VPD YES
RECOVERY_CATALOG_USER     YES
RESOURCE                 YES
SCHEDULER_ADMIN            YES
SELECT_CATALOG_ROLE       YES
SODA_APP                  YES
SYSUMF_ROLE                YES
WM_ADMIN_ROLE               YES
XDBADMIN                  YES
XDB_SET_INVOKER             YES
XDB_WEBSERVICES             YES
XDB_WEBSERVICES_OVER_HTTP   YES
XDB_WEBSERVICES_WITH_PUBLIC YES
XS_CACHE_ADMIN               YES
XS_CONNECT                  YES
XS_NAMESPACE_ADMIN            YES
XS_SESSION_ADMIN               YES

87 rows selected.

SQL>

```

You view the same list.

- f. Create a common role in PDB1.

```

SQL> CREATE ROLE c##_role_pdb1 CONTAINER=ALL;
create role c##_role_pdb1 container=ALL
*
ERROR at line 1:
ORA-65050: Common DDLs only allowed in root

SQL>

```

You get an error message because no common role can be created from a PDB.

- g. Create a local role in PDB1.

```
SQL> CREATE ROLE local_role_pdb1 CONTAINER=CURRENT;
```

Role created.

```
SQL> SELECT role, common FROM dba_roles ORDER BY role;
```

ROLE	COMMON
ADM_PARALLEL_EXECUTE_TASK	YES
APPLICATION_TRACE_VIEWER	YES
AQ_ADMINISTRATOR_ROLE	YES
AQ_USER_ROLE	YES
AUDIT_ADMIN	YES
AUDIT_VIEWER	YES
AUTHENTICATEDUSER	YES
C##_ROLE	YES
CAPTURE_ADMIN	YES
CDB_DBA	YES
CONNECT	YES
CTXAPP	YES
DATAPATCH_ROLE	YES
DATAPUMP_EXP_FULL_DATABASE	YES
DATAPUMP_IMP_FULL_DATABASE	YES
DBA	YES
DBFS_ROLE	YES
DBJAVASCRIPT	YES
DBMS_MDX_INTERNAL	YES
DV_ACCTMGR	YES
DV_ADMIN	YES
DV_AUDIT_CLEANUP	YES
DV_DATAPUMP_NETWORK_LINK	YES
DV_GOLDENGATE_ADMIN	YES
DV_GOLDENGATE_REDO_ACCESS	YES
DV_MONITOR	YES
DV_OWNER	YES
DV_PATCH_ADMIN	YES
DV_POLICY_OWNER	YES
DV_PUBLIC	YES
DV_REALM_OWNER	YES
DV_REALM_RESOURCE	YES
DV_SECANALYST	YES

DV_STREAMS_ADMIN	YES
DV_XSTREAM_ADMIN	YES
EJBCLIENT	YES
EM_EXPRESS_ALL	YES
EM_EXPRESS_BASIC	YES
EXECUTE_CATALOG_ROLE	YES
EXP_FULL_DATABASE	YES
GATHER_SYSTEM_STATISTICS	YES
GDS_CATALOG_SELECT	YES
GGSYS_ROLE	YES
GLOBAL_AQ_USER_ROLE	YES
GSMADMIN_ROLE	YES
GSMUSER_ROLE	YES
GSM_POOLADMIN_ROLE	YES
HS_ADMIN_EXECUTE_ROLE	YES
HS_ADMIN_ROLE	YES
HS_ADMIN_SELECT_ROLE	YES
IMP_FULL_DATABASE	YES
JAVADEBUGPRIV	YES
JAVAIDPRIV	YES
JAVASYSPRIV	YES
JAVAUSERPRIV	YES
JAVA_ADMIN	YES
JMXSERVER	YES
LBAC_DBA	YES
<b>LOCAL_ROLE_PDB1</b>	<b>NO</b>
LOGSTDBY_ADMINISTRATOR	YES
OEM_ADVISOR	YES
OEM_MONITOR	YES
OLAP_DBA	YES
OLAP_USER	YES
OLAP_XS_ADMIN	YES
OPTIMIZER_PROCESSING_RATE	YES
ORDADMIN	YES
PDB_DBA	YES
PROVISIONER	YES
RDFCTX_ADMIN	YES
RECOVERY_CATALOG_OWNER	YES
RECOVERY_CATALOG_OWNER_VPD	YES
RECOVERY_CATALOG_USER	YES
RESOURCE	YES
SCHEDULER_ADMIN	YES

```

SELECT_CATALOG_ROLE           YES
SODA_APP                      YES
SYSUMF_ROLE                    YES
WM_ADMIN_ROLE                  YES
XDBADMIN                       YES
XDB_SET_INVOKER                YES
XDB_WEBSERVICES                 YES
XDB_WEBSERVICES_OVER_HTTP      YES
XDB_WEBSERVICES_WITH_PUBLIC    YES
XS_CACHE_ADMIN                  YES
XS_CONNECT                      YES
XS_NAMESPACE_ADMIN               YES
XS_SESSION_ADMIN                 YES

88 rows selected.

SQL>

```

13. Grant common or local roles as common or local.

- Grant a common role to a common user from the CDB root.

```

SQL> CONNECT / AS SYSDBA
Connected.
SQL> GRANT c##_role TO c##_user;

Grant succeeded.

SQL> SELECT grantee, granted_role, common, con_id
      FROM cdb_role_privs WHERE grantee = 'C##_USER';
2
GRANTEE          GRANTED_ROLE COMMON CON_ID
-----
C##_USER         C##_ROLE      NO      1

SQL>

```

**Note that the common role is granted locally to the common user. The granted role is applicable only in the CDB root.**

```

SQL> CONNECT c##_user
Enter password: password
Connected.
SQL> SELECT * FROM session_roles;
ROLE
-----

```

```
C##_ROLE
SQL>
```

```
SQL> CONNECT c##_user@PDB1
Enter password: password
Connected.
SQL> SELECT * FROM session_roles;

no rows selected

SQL>
```

- b. Now grant the common role to a common user from the CDB root as common, to be applicable in all containers.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> GRANT c##_role TO c##_user CONTAINER=ALL;

Grant succeeded.

SQL> SELECT grantee, granted_role, common, con_id
      FROM cdb_role_privs WHERE grantee = 'C##_USER';
2
GRANTEE          GRANTED_ROLE        COMMON CON_ID
-----          -----
C##_USER          C##_ROLE           YES    3
C##_USER          C##_ROLE           YES    5
C##_USER          C##_ROLE           YES    6
C##_USER          C##_ROLE           NO     1
C##_USER          C##_ROLE           YES    1
C##_USER          C##_ROLE           YES    4

6 rows selected.

SQL>
```

```
SQL> CONNECT c##_user
Enter password: password
Connected.
SQL> SELECT * FROM session_roles;

ROLE
-----
```

```
C##_ROLE  
SQL>
```

```
SQL> CONNECT c##_user@PDB1  
Enter password: password  
Connected.  
SQL> SELECT * FROM session_roles;  
  
ROLE  
-----  
C##_ROLE  
  
SQL>
```

- c. Revoke the common role from the common user so that the role cannot be used in any container.

```
SQL> CONNECT / AS SYSDBA  
Connected.  
SQL> REVOKE c##_role FROM c##_user CONTAINER=ALL;  
  
Revoke succeeded.  
  
SQL> CONNECT c##_user  
Enter password: password  
Connected.  
SQL> SELECT * FROM session_roles;  
ROLE  
-----  
C##_ROLE  
  
SQL>
```

```
SQL> CONNECT c##_user@PDB1  
Enter password: password  
Connected.  
SQL> SELECT * FROM session_roles;  
  
no rows selected  
  
SQL>
```

- d. Grant a common role to a local user from the CDB root.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> GRANT c##_role TO local_user_pdb1;
GRANT c##_role TO local_user_pdb1
*
ERROR at line 1:
ORA-01917: user or role 'LOCAL_USER_PDB1' does not exist

SQL>
```

**Note that the user is unknown in the CDB root. It is a local user in PDB1.**

- e. Grant a common role to a local user from PDB1.

```
SQL> CONNECT system@PDB1
Enter password: password
Connected.
SQL> GRANT c##_role TO local_user_pdb1;

Grant succeeded.

SQL> SELECT grantee, granted_role, common, con_id
      FROM cdb_role_privs WHERE grantee='LOCAL_USER_PDB1';
2
GRANTEE          GRANTED_ROLE        COMMON CON_ID
-----
LOCAL_USER_PDB1    C##_ROLE           NO            5

SQL>
```

**Note that the user is granted a common role locally (common column = NO) applicable only in the PDB PDB1 .**

- f. Test the connection as the local user.

```
SQL> CONNECT local_user_pdb1@PDB1
Enter password: password
Connected.
SQL> SELECT * FROM session_roles;

ROLE
-----
C##_ROLE

SQL>
```

- g. Grant a common role to a local user from PDB1 applicable in all containers.

```
SQL> CONNECT system@PDB1
Enter password: password
Connected.
SQL> GRANT c##_role TO local_user_pdb1 CONTAINER=ALL;
GRANT c##_role TO local_user_pdb1 CONTAINER=ALL
*
ERROR at line 1:
ORA-65030: cannot grant a privilege commonly to a local user or
role

SQL>
```

**Notice that a common role cannot be granted globally from a PDB.**

- h. Grant a local role to a local user from PDB1.

```
SQL> GRANT local_role_pdb1 TO local_user_pdb1;

Grant succeeded.

SQL> SELECT grantee, granted_role, common, con_id
      FROM cdb_role_privs WHERE grantee='LOCAL_USER_PDB1';
2
GRANTEE          GRANTED_ROLE        COMMON CON_ID
-----
LOCAL_USER_PDB1    LOCAL_ROLE_PDB1    NO      5
LOCAL_USER_PDB1    C##_ROLE          NO      5

SQL>
```

- i. Test the connection as the local user.

```
SQL> CONNECT local_user_pdb1@PDB1
Enter password: password
Connected.
SQL> SELECT * FROM session_roles;

ROLE
-----
C##_ROLE
LOCAL_ROLE_PDB1

SQL>
```

14. You will manage privileges granted as common or local in CDB and regular PDBs.
- Check whether privileges are created as common or local.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> DESC sys.system_privilege_map
Name          Null?    Type
-----
PRIVILEGE      NOT NULL NUMBER
NAME           NOT NULL VARCHAR2(40)
PROPERTY        NOT NULL NUMBER

SQL> DESC sys.table_privilege_map
Name          Null?    Type
-----
PRIVILEGE      NOT NULL NUMBER
NAME           NOT NULL VARCHAR2(40)

SQL>
```

**Notice that there is no COMMON column. Privileges are created neither as common nor as local, but they can be granted as common or local.**

- Check how the CREATE SESSION system privilege was granted to C##\_USER and LOCAL\_USER\_PDB1 users.

```
SQL> CONNECT system
Enter password: password
Connected.
SQL> SELECT grantee, privilege, common, con_id
      FROM cdb_sys_privs
     WHERE grantee IN ('C##_USER', 'LOCAL_USER_PDB1');
2   3
GRANTEE          PRIVILEGE      COMMON CON_ID
-----
C##_USER         CREATE SESSION YES    3
LOCAL_USER_PDB1 CREATE SESSION NO     5
C##_USER         CREATE SESSION YES    5
C##_USER         CREATE SESSION YES    1
C##_USER         CREATE SESSION YES    4
C##_USER         CREATE SESSION YES    6

6 rows selected.

SQL>
```

```

SQL> CONNECT system@PDB1
Enter password: password
Connected.
SQL> SELECT grantee, privilege, common
  FROM  dba_sys_privs
 WHERE  grantee in ('C##_USER', 'LOCAL_USER_PDB1');
 2      3
GRANTEE          PRIVILEGE        COMMON
-----
LOCAL_USER_PDB1    CREATE SESSION    NO
C##_USER          CREATE SESSION    YES

SQL>

```

- c. Grant the system privileges CREATE TABLE and UNLIMITED TABLESPACE to common user C##\_USER to be applicable in any container. This will be a common privilege.

```

SQL> CONNECT system
Enter password: password
Connected.
SQL> GRANT CREATE TABLE, UNLIMITED TABLESPACE TO c##_user
CONTAINER=ALL;
 2
Grant succeeded.

SQL> SELECT grantee, privilege, common, con_id
  FROM  cdb_sys_privs
 WHERE  grantee = 'C##_USER'
 ORDER BY con_id;
 2      3      4
GRANTEE          PRIVILEGE        COMMON CON_ID
-----
C##_USER          CREATE TABLE     YES      1
C##_USER          UNLIMITED TABLESPACE YES      1
C##_USER          CREATE SESSION    YES      1
C##_USER          CREATE TABLE     YES      3
C##_USER          CREATE SESSION    YES      3
C##_USER          UNLIMITED TABLESPACE YES      3
C##_USER          CREATE TABLE     YES      4
C##_USER          UNLIMITED TABLESPACE YES      4
C##_USER          CREATE SESSION    YES      4
C##_USER          CREATE TABLE     YES      5

```

```

C##_USER          UNLIMITED TABLESPACE YES      5
C##_USER          CREATE SESSION      YES      5
C##_USER          CREATE TABLE       YES      6
C##_USER          UNLIMITED TABLESPACE YES      6
C##_USER          CREATE SESSION      YES      6

```

15 rows selected.

SQL>

- d. Grant the system privilege CREATE SEQUENCE to common user C##\_USER to be applicable in the CDB root only. This will be a local privilege.

```
SQL> GRANT CREATE SEQUENCE TO c##_user CONTAINER=CURRENT;
```

Grant succeeded.

```

SQL> SELECT grantee, privilege, common, con_id
      FROM cdb_sys_privs
     WHERE grantee = 'C##_USER'
   ORDER BY con_id;

```

GRANTEE	PRIVILEGE	COMMON	CON_ID
C##_USER	CREATE SEQUENCE	NO	1
C##_USER	CREATE SESSION	YES	1
C##_USER	UNLIMITED TABLESPACE	YES	1
C##_USER	CREATE TABLE	YES	1
C##_USER	CREATE SESSION	YES	3
C##_USER	CREATE TABLE	YES	3
C##_USER	UNLIMITED TABLESPACE	YES	3
C##_USER	UNLIMITED TABLESPACE	YES	4
C##_USER	CREATE SESSION	YES	4
C##_USER	CREATE TABLE	YES	4
C##_USER	CREATE SESSION	YES	5
C##_USER	UNLIMITED TABLESPACE	YES	5
C##_USER	CREATE TABLE	YES	5
C##_USER	CREATE SESSION	YES	6
C##_USER	UNLIMITED TABLESPACE	YES	6
C##_USER	CREATE TABLE	YES	6

16 rows selected.

SQL>

- e. Grant the system privilege CREATE SYNONYM to common user C##\_USER to be applicable in PDB1 only. This will be a local privilege.

```
SQL> CONNECT system@PDB1
Enter password: password
Connected.
SQL> GRANT CREATE SYNONYM TO c##_user CONTAINER=CURRENT;

Grant succeeded.

SQL> SELECT grantee, privilege, common, con_id
  FROM  cdb_sys_privs
 WHERE  grantee = 'C##_USER'
 ORDER BY con_id;
 2      3      4
GRANTEE          PRIVILEGE          COMMON CON_ID
-----
C##_USER          CREATE SYNONYM        NO      5
C##_USER          CREATE SESSION       YES     5
C##_USER          UNLIMITED TABLESPACE YES     5
C##_USER          CREATE TABLE        YES     5

SQL>
```

- f. Grant the system privilege CREATE VIEW to common user C##\_USER to be applicable in root only but connected in PDB1.

```
SQL> GRANT CREATE VIEW TO c##_user CONTAINER=ALL;
GRANT CREATE VIEW TO c##_user CONTAINER=ALL
*
ERROR at line 1:
ORA-65040: operation not allowed from within a pluggable
database

SQL>
```

**Note that you cannot grant a common privilege from a PDB.**

- g. Grant the system privilege CREATE ANY TABLE to local user LOCAL\_USER\_PDB1 to be applicable in any container.

```
SQL> CONNECT system
Enter password: password
Connected.
SQL> GRANT CREATE ANY TABLE TO local_user_pdb1 CONTAINER=ALL;
GRANT CREATE ANY TABLE TO local_user_pdb1 CONTAINER=ALL
*
ERROR at line 1:
```

```
ORA-01917: user or role 'LOCAL_USER_PDB1' does not exist
SQL>
```

**Notice that the user is unknown in the CDB root. It is a local user in PDB1.**

- h. Grant the system privilege UNLIMITED TABLESPACE to local user LOCAL\_USER\_PDB1 to be applicable in PDB1 only. This will be a local privilege.

```
SQL> CONNECT system@PDB1
Enter password: password
Connected.
SQL> GRANT UNLIMITED TABLESPACE TO local_user_pdb1;

Grant succeeded.

SQL> SELECT grantee, privilege, common, con_id
   FROM cdb_sys_privs
  WHERE grantee = 'LOCAL_USER_PDB1';
2      3
GRANTEE          PRIVILEGE      COMMON CON_ID
-----          -----
LOCAL_USER_PDB1    CREATE SESSION    NO        5
LOCAL_USER_PDB1    UNLIMITED TABLESPACE NO        5

SQL>
```

- i. Grant the system privilege DROP ANY VIEW to local user LOCAL\_USER\_PDB1 to be applicable in root only but connected in PDB1.

```
SQL> GRANT DROP ANY VIEW TO local_user_pdb1 CONTAINER=ALL;
GRANT DROP ANY VIEW TO local_user_pdb1 CONTAINER=ALL
*
ERROR at line 1:
ORA-65030: cannot grant a privilege commonly to a local user or
role

SQL>
```

**Notice that you cannot grant a local privilege that will be applicable in another container.**

15. 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 application common 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 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 the `CDB_USERS` view, the `CDB_ROLES` view also has a new `INHERITED` column.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: password
Connected.

SQL> SELECT role, common, inherited, con_id FROM cdb_roles
      WHERE role IN ('DBA' , 'CONNECT') ORDER BY 4, 1;
2
ROLE                  COMMON INH CON_ID
-----
CONNECT                YES   YES    3
DBA                     YES   YES    3
CONNECT                YES   YES    4
DBA                     YES   YES    4
CONNECT                YES   YES    6
DBA                     YES   YES    6

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 and the two application PDBs, `robots` and `dolls`. 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   3
ROLE                  COMMON INH CON_ID
-----  -----  -----
DBA                   YES    YES     3
DBA                   YES    YES     4
DBA                   YES    YES     6
EMP_ROLE              YES    NO      3
MGR_ROLE              YES    NO      3

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.**

c. Synchronize the application PDBs.

```

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

Pluggable database altered.

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

Pluggable database altered.

SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: 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    3
EMP_ROLE                   YES   YES   4
EMP_ROLE                   YES   YES   6
MGR_ROLE                   YES   NO    3
MGR_ROLE                   YES   YES   4
MGR_ROLE                   YES   YES   6

6 rows selected.

SQL>
```

*Q/ 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?*

**A/ Yes, if the application seed existed it would be synchronized first.**

- d. Create the new doodles application PDB.

```
SQL> !mkdir /u02/app/oracle/oradata/ORCL/toys_root/doodles
SQL>
SQL> ALTER SESSION SET db_create_file_dest =
      '/u02/app/oracle/oradata/ORCL/toys_root/doodles';
2
Session altered.

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

SQL> ALTER PLUGGABLE DATABASE doodles OPEN;
Pluggable database altered.

SQL>
```

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

```
SQL> CONNECT system@doodles
Enter password: password
Connected.

SQL> SELECT username, common, inherited, con_id
      FROM cdb_users
      WHERE common='YES' AND username like 'C#%'
      ORDER BY 2, 3;
2      3      4
USERNAME                  COMMON INH CON_ID
-----  -----  -----
C##U                      YES     YES    10
C##_USER                   YES     YES    10

SQL>
```

*Q/ Which type of users are displayed? Common users at the CDB level or application root level?*

**A/ Common users at the CDB level: C## users can only be created in the CDB root.**

```
SQL> SELECT role, common, inherited, con_id FROM cdb_roles
      WHERE role IN ('MGR_ROLE', 'EMP_ROLE') ORDER BY 1,4;
```

2

```
no rows selected
SQL>
```

**Q/ Why are the application common roles not displayed?**

**A/ robots and dolls were synchronized whereas doodles has not been synchronized yet.**

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

Pluggable database altered.

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>
```

16. Application common privileges granted within application roots follow the same rules as the application common users and application common roles requiring synchronization in application PDBs. Manage system administrative privileges such as SYSDBA in application root and application PDBs. Within `toys_root`, grant the SYSDBA privilege to `toys_owner`.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: password
Connected.
SQL> SELECT username, sysdba, con_id FROM v$pwfile_users;

USERNAME          SYSDBA CON_ID
-----
SYS              TRUE    0
SYSDG            FALSE   0
SYSBACKUP        FALSE   0
SYSKM            FALSE   0
SYSTEM           FALSE   0
```

```

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

SQL> GRANT SYSDBA TO toys_owner CONTAINER = ALL;

Grant succeeded.

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

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

```

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

```

SQL> SELECT username, sysdba, common, con_id
      FROM v$pwfile_users;
2
USERNAME          SYSDB COMMON CON_ID
-----
SYS              TRUE  YES    0
SYSDG             FALSE YES    0
SYSBACKUP         FALSE YES    0
SYSKM             FALSE YES    0
SYSTEM            FALSE YES    0
TOYS_OWNER        TRUE  YES    3

6 rows selected.

SQL> CONNECT toys_owner@robots AS SYSDBA
Enter password: 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 application 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: password
Connected.
SQL> SELECT username, sysdba, common, con_id
      FROM v$pwfile_users;
2
USERNAME          SYSDB COMMON CON_ID
-----
SYS              TRUE  YES   0
SYSDG            FALSE YES   0
SYSBACKUP        FALSE YES   0
SYSKM            FALSE YES   0
SYSTEM           FALSE 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
SYSDG            FALSE YES   0
SYSBACKUP        FALSE YES   0
SYSKM            FALSE YES   0
SYSTEM           FALSE YES   0
TOYS_OWNER       TRUE  YES   4
6 rows selected.
```

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

Pluggable database altered.

SQL> CONNECT toys_owner@dolls AS SYSDBA
Enter password: password
Connected.
SQL> SELECT username, sysdba, common, con_id
      FROM v$pwfile_users;
2
-----  
USERNAME          SYSDB COMMON CON_ID
-----  
SYS              TRUE  YES    0
SYSDG             FALSE YES    0
SYSBACKUP         FALSE YES    0
SYSKM             FALSE YES    0
SYSTEM            FALSE YES    0
TOYS_OWNER        TRUE  YES    6
6 rows selected.

SQL>
```

## Practice 7-2: Managing Common and Local Objects in Application Containers

### Overview

In this practice, you will manage local and common objects in application PDBs.

### Tasks

1. 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> CONNECT sys@toys_root AS SYSDBA
Enter password: password
Connected.
SQL> SELECT owner, object_name, sharing, application
   FROM dba_objects
  WHERE owner = 'TOYS_OWNER';
  2      3
OWNER          OBJECT_NAM SHARING
----- -----
TOYS_OWNER      SALES_DATA METADATA LINK Y
TOYS_OWNER      CODES       DATA     LINK      Y
SQL>
```

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

*A/ 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$';
  2      3
OWNER          OBJECT_NAM SHARING
----- -----
SYS            TAB$      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 '1.3' TO '1.4';
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 '1.4';
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: password
Connected.
SQL> DESC toys_owner.tab1
ERROR:
ORA-04043: object toys_owner.tab1 does not exist

SQL>
```

**Q/ Why is the table not visible?**

**A/ 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: password  
Connected.  
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;  
  
Pluggable database altered.  
  
SQL> CONNECT toys_test2@robots  
Enter password: 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: 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. Data inserted, updated, or deleted during application install or upgrade is visible to all application PDBs. Nevertheless, data inserted, updated, or deleted locally in application PDBs is accessible by their respective application PDBs only.*

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

SQL> CONNECT sys@robots AS SYSDBA
Enter password: password
Connected.

SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.
```

```
SQL> CONNECT sys@dolls AS SYSDBA
Enter password: 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: 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: 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: password  
Connected.  
SQL> CREATE USER user1 IDENTIFIED BY password  
      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: password  
Connected.  
SQL> CREATE USER user1 IDENTIFIED BY password  
      QUOTA unlimited ON system;  
2  
User created.
```

```
SQL> CREATE USER l_user IDENTIFIED BY password;
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: 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: 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> ROLLBACK;  
  
Rollback complete.  
  
SQL> EXIT  
$
```

*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.*

## Practice 7-3: Enabling Common Users to View Information About PDB Objects

### Overview

In this practice, you will manage the CONTAINER\_DATA attributes of common users to enable common users to view information about PDB objects in specific PDBs.

### Tasks

1. Execute the \$HOME/labs/sec/glogin\_7b.sh shell script. The script sets formatting for all columns selected in queries.

```
$ $HOME/labs/sec/glogin_7b.sh
$
```

Find information about the default (user-level) and object-specific CONTAINER\_DATA attributes that are explicitly set to a value other than DEFAULT in the DBA\_CONTAINER\_DATA data dictionary view.

```
$ sqlplus / AS SYSDBA

SQL> SELECT USERNAME, DEFAULT_ATTR as "Def_Att", OWNER,
       OBJECT_NAME, ALL_CONTAINERS "All_Cont",
       CONTAINER_NAME as "C_NAME", CON_ID as "CID"
  FROM CDB_CONTAINER_DATA
 WHERE username NOT IN ('GSMADMIN_INTERNAL', 'APPQOSSYS',
                        'DBSNMP', 'SYSRAC', 'SYSDG', 'DBSFWUSER')
   ORDER BY OBJECT_NAME;
2   3   4   5   6   7
USERNAME      Def_Att OWNER OBJECT_NAM All_Cont C_NAME      CID
-----  -----  -----  -----  -----  -----  -----
SYS          Y           Y           Y           3
SYSBACKUP    Y           Y           Y           3
SYSBACKUP    Y           Y           Y           1
SYS          Y           Y           Y           1
SYSTEM        Y           Y           Y           1

SQL>
```

2. Create the common user c##jfv and grant c##jfv the system privileges CREATE SESSION and SET CONTAINER.

```
SQL> CREATE USER c##jfv IDENTIFIED BY password CONTAINER=ALL;

User created.

SQL> GRANT CREATE SESSION, SET CONTAINER TO c##jfv
```

```
CONTAINER=ALL;

Grant succeeded.

SQL>
```

3. Then grant c##jfsv the object privileges SELECT on V\_\$SESSION view.

```
SQL> GRANT SELECT ON sys.v_$session TO c##jfsv CONTAINER=ALL;

Grant succeeded.

SQL>
```

4. Create a second session connected to PDB1 as user SYS and stay connected.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus sys@PDB1 AS SYSDBA
Enter password: password

SQL>
```

5. In the first session, you should see one row for PDB1.

```
SQL> SELECT username, con_id FROM v_$session
      WHERE username IS NOT NULL AND username <> 'DBSNMP';
      2
USERNAME    CON_ID
-----
SYS          0
SYS          5
SYS          1

SQL>
```

6. Still in the first session, you connect as the common user c##jfsv. The common user does not see any information in V\_\$SESSION related to PDB1.

```
SQL> CONNECT c##jfsv
Enter password: password
Connected.
SQL> SELECT username, con_id FROM sys.v_$session
      WHERE username IS NOT NULL AND username <> 'DBSNMP';
      2
USERNAME    CON_ID
-----
SYS          0
```

```
C##JFV          1
SQL>
```

7. Enable the common user c##jfv to see information in V\_\$SESSION related to PDB1.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER USER c##jfv
      SET CONTAINER_DATA = (CDB$ROOT, PDB1)
      FOR V_$SESSION
      CONTAINER=CURRENT;
2   3   4
User altered.

SQL>
```

8. Connect as the common user c##jfv to view information in V\$SESSION related to PDB1.

```
SQL> CONNECT c##jfv
Enter password: password
Connected.
SQL> SELECT username, con_id FROM sys.v$session
      WHERE username IS NOT NULL AND username <> 'DBSNMP';
2
USERNAME    CON_ID
-----
SYS          0
SYS          5
C##JFV       1

SQL>
```

9. View the CONTAINER\_DATA attribute set for the common user C##JFV on object V\$SESSION in PDB1.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT USERNAME, DEFAULT_ATTR as "Def_Attr", OWNER,
      OBJECT_NAME, ALL_CONTAINERS "All_Cont",
      CONTAINER_NAME as "C_NAME", CON_ID as "CID"
      FROM CDB_CONTAINER_DATA
      WHERE username NOT IN ('GSMADMIN_INTERNAL', 'APPQOSSYS',
                             'DBSNMP', 'SYSRAC', 'SYSDG', 'DBSFWUSER')
      ORDER BY OBJECT_NAME;
2   3   4   5   6   7
USERNAME  Def_Attr OWNER OBJECT_NAME All_Cont C_NAME     CID
```

C##JFV	N	SYS	V\$_SESSION	N	PDB1	1
C##JFV	N	SYS	V\$_SESSION	N	CDB\$ROOT	1
SYSTEM	Y			Y		1
SYS	Y			Y		1
SYSBACKUP	Y			Y		1
SYS	Y			Y		3
SYSBACKUP	Y			Y		3
7 rows selected.						
SQL> <b>EXIT</b>						
\$						

10. Exit from all SQL\*Plus sessions.

## Practice 7-4: 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: password

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    1.5      NORMAL toys_root          3 TOYS_ROOT
TOYS_APP    1.5      NORMAL robots            4 ROBOTS
TOYS_APP    1.2      NORMAL doodles          10 DOODLES
TOYS_APP    1.5      NORMAL dolls             6 DOLLS

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
3	TOYS_ROOT	READ WRITE NO	
4	ROBOTS	MOUNTED	
6	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	CON_ID
1	Puppet	Y	3
2	Car	Y	3
3	Game	N	3

```

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

SQL>
```

### 3. Open the application PDBs.

```

SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
Pluggable database altered.

SQL>
```

*Q/ Which operation is 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: 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: 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. If this NEWPDB has not been dropped through the \$HOME/admin/cleanup\_PDB.sh shell script, drop it.

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

```
ALTER PLUGGABLE DATABASE newpdb CLOSE
*
ERROR at line 1:
ORA-65011: Pluggable database NEWPDB does not exist.

SQL> DROP PLUGGABLE DATABASE newpdb INCLUDING DATAFILES;

Pluggable database dropped.

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: 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 password;
2
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE newpdb OPEN;

Pluggable database altered.

SQL> CONNECT sys@newpdb AS SYSDBA
Enter password: password
Connected.
SQL> SELECT code, label, c3 FROM toys_owner.codes;
SELECT code, label, c3 FROM toys_owner.codes
*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
```

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

**A/ Synchronization of the application PDBs is required.**

```
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>
```

- c. Drop the new PDB.

```
SQL> ALTER PLUGGABLE DATABASE newpdb CLOSE;
```

Pluggable database altered.

```
SQL> CONNECT / AS SYSDBA
```

Connected.

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

Pluggable database dropped.

```
SQL>
```

- 5. What happens if the data-linked table is dropped?

- a. Drop the table after upgrading the application.

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

Enter password: *password*

Connected.

```
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: *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	1.6	NORMAL	4	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	4	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: password
Connected.
SQL> ALTER PLUGGABLE DATABASE APPLICATION toys_app SYNC;

Pluggable database altered.

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

Pluggable database altered.

SQL>

```

- b. Check that all toys\_root application PDBs are in sync with the application root.

```

SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: 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      5 ROBOTS
TOYS_APP      1.7      NORMAL      4 TOYS_ROOT
TOYS_APP      1.7      NORMAL     10 DOODLES
TOYS_APP      1.7      NORMAL      6 DOLLS

SQL> EXIT
$
```

## Practice 7-5: Managing PDB Lockdown Profiles

### Overview

In this practice, you will observe how lockdown profiles in the CDB root, in application roots, and application PDBs behave regarding inheritance. You will create and alter lockdown profiles and set the profiles at different PDB levels in a CDB. You will also discover the behavior of static and dynamic lockdown profiles created from base lockdown profiles.

### Tasks

1. Before starting the practice, execute the `$HOME/labs/sec/HR_ROOT.sh` shell script. It creates the `HR_ROOT` application root, installs `HR_APP` in the application root, and creates the `OPERATIONS` and `SALES` application PDBs.

```
$ $HOME/labs/sec/HR_ROOT.sh  
...  
$
```

2. Create lockdown profiles at different levels in the CDB.

- a. Create two lockdown profiles in the CDB root: `CDB_prof1` and `CDB_prof2`.

```
$ sqlplus / AS SYSDBA  
  
SQL> CREATE LOCKDOWN PROFILE CDB_prof1;  
  
Lockdown Profile created.  
  
SQL> ALTER LOCKDOWN PROFILE CDB_prof1  
      DISABLE STATEMENT = ('alter system');  
      2  
Lockdown Profile altered.  
  
SQL> ALTER LOCKDOWN PROFILE CDB_prof1  
      ENABLE STATEMENT = ('alter system')  
      CLAUSE = ('set');  
      2      3  
Lockdown Profile altered.  
  
SQL>
```

```
SQL> CREATE LOCKDOWN PROFILE CDB_prof2;  
  
Lockdown Profile created.  
  
SQL> ALTER LOCKDOWN PROFILE CDB_prof2
```

```

        DISABLE STATEMENT = ('alter pluggable database');

2
Lockdown Profile altered.

SQL>

```

- b. Verify the existence of lockdown profiles.

```

SQL> SELECT profile_name, rule, clause, status
      FROM cdb_lockdown_profiles;

2
-----+-----+-----+-----+
PROFILE_NAME      RULE          CLAUSE        STATUS
-----+-----+-----+-----+
CDB_PROF1         ALTER SYSTEM
CDB_PROF1         ALTER SYSTEM      SET           ENABLE
CDB_PROF2         ALTER PLUGGABLE DATABASE
PRIVATE_DBaaS
PUBLIC_DBaaS
SAAS

6 rows selected.

SQL>

```

- c. Log in to the HR\_ROOT application root as HR\_LOCK\_MGR to create the app\_root\_prof lockdown profile in HR\_ROOT.

```

SQL> CONNECT hr_lock_mgr@hr_root
Enter password: password
Connected.
SQL> CREATE LOCKDOWN PROFILE app_root_prof;

Lockdown Profile created.

SQL> ALTER LOCKDOWN PROFILE app_root_prof
        DISABLE STATEMENT = ('alter system');

2
Lockdown Profile altered.

SQL> ALTER LOCKDOWN PROFILE app_root_prof
        ENABLE STATEMENT = ('alter system')
        CLAUSE = ('flush shared_pool');

2      3
Lockdown Profile altered.

SQL>

```

- d. Verify the existence of the lockdown profile in the application root.

```
SQL> SELECT profile_name, rule, clause, status
      FROM cdb_lockdown_profiles;
2
PROFILE_NAME      RULE          CLAUSE          STATUS
-----
APP_ROOT_PROF    ALTER SYSTEM
APP_ROOT_PROF    ALTER SYSTEM    FLUSH SHARED_POOL    ENABLE

SQL>
```

- e. Create the `sales_prof` lockdown profile for the `SALES` application PDB in the application root.

```
SQL> CREATE LOCKDOWN PROFILE sales_prof;

Lockdown Profile created.

SQL> ALTER LOCKDOWN PROFILE sales_prof
      DISABLE STATEMENT = ('alter pluggable database');

2
Lockdown Profile altered.

SQL> SELECT profile_name, rule, clause, status
      FROM cdb_lockdown_profiles;
2
PROFILE_NAME      RULE          CLAUSE          STATUS
-----
APP_ROOT_PROF    ALTER SYSTEM    DISABLE
APP_ROOT_PROF    ALTER SYSTEM    FLUSH SHARED_POOL    ENABLE
SALES_PROF       ALTER PLUGGABLE DATABASE    DISABLE

SQL>
```

3. Observe profile lockdown inheritance when the CDB root lockdown profile is set.

- a. Set the `CDB_prof1` lockdown profile in the CDB root.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHOW PARAMETER pdb_lockdown
```

NAME	TYPE	VALUE
pdb_lockdown	string	
<b>SQL&gt; ALTER SYSTEM SET pdb_lockdown = CDB_PROF1 SCOPE = BOTH;</b>		
System altered.		
<b>SQL&gt; SHOW PARAMETER pdb_lockdown</b>		
NAME	TYPE	VALUE
pdb_lockdown	string	CDB_PROF1
<b>SQL&gt;</b>		

- b. Observe how the PDB1 regular PDB inherits the restrictions of the CDB root lockdown profile.

```
SQL> CONNECT sys@PDB1 AS SYSDBA
Enter password: password
Connected.
SQL> SHOW PARAMETER pdb_lockdown

NAME          TYPE        VALUE
-----
pdb_lockdown string      CDB_PROF1
SQL> ALTER SYSTEM SET ddl_lock_timeout=30 SCOPE = BOTH;

System altered.

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

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

SQL>
```

- c. Observe how the `HR_ROOT` application root inherits the restrictions of the CDB root lockdown profile.

```
SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: password
Connected.
SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE        VALUE
-----
pdb_lockdown              string      CDB_PROF1
SQL> ALTER SYSTEM SET ddl_lock_timeout=30 SCOPE=BOTH;

System altered.

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

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

SQL>
```

- d. Observe how the `SALES` application PDB inherits the restrictions of the CDB root lockdown profile.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: password
Connected.
SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE        VALUE
-----
pdb_lockdown              string      CDB_PROF1
SQL> ALTER SYSTEM SET ddl_lock_timeout=30 SCOPE=BOTH;

System altered.
```

```

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

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

SQL>

```

4. Observe profile lockdown inheritance when the application root lockdown profile is set in the application root.
  - a. Set the `app_root_prof` lockdown profile in the application root.

```

SQL> CONNECT hr_lock_mgr@hr_root
Enter password: password
Connected.
SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE          VALUE
-----
pdb_lockdown              string        CDB_PROF1
SQL> ALTER SYSTEM SET pdb_lockdown = app_root_prof
      SCOPE=BOTH;
2
System altered.

SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE          VALUE
-----
pdb_lockdown              string        APP_ROOT_PROF
SQL>

```

- b. Observe how the `HR_ROOT` application root inherits the restrictions of the CDB root lockdown profile.

```
SQL> CONNECT system@hr_root
Enter password: password
Connected.
SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE        VALUE
-----
pdb_lockdown              string      APP_ROOT_PROF
SQL> ALTER SYSTEM SET ddl_lock_timeout=30 SCOPE=BOTH;

System altered.

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

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

SQL>
```

Observe that `app_root_prof` affects all application PDBs in the application container, but the application root still inherits the rules of its nearest ancestor, `CDB_PROF1`.

- c. Observe how the `SALES` application PDB inherits the restrictions of the application root lockdown profile.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM SET ddl_lock_timeout=30 SCOPE=BOTH;
ALTER SYSTEM SET ddl_lock_timeout=30 scope=both
*
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 CHECKPOINT;
ALTER SYSTEM CHECKPOINT
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

The SALES application PDB inherits the restrictions of the application root lockdown profile in addition to those set in its nearest ancestor, CDB\_PROF1.

5. Observe profile lockdown inheritance when the application root lockdown profile is set in the application PDB.
  - a. Set the sales\_prof lockdown profile in the SALES application PDB.

```
SQL> CONNECT hr_lock_mgr@sales
Enter password: password
Connected.
SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE          VALUE
-----
pdb_lockdown              string        APP_ROOT_PROF
SQL> ALTER SYSTEM SET pdb_lockdown = '' SCOPE = BOTH;
ALTER SYSTEM SET pdb_lockdown = '' SCOPE = both
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

- 1) Temporarily unset app\_root\_prof so as to be able to set the sales\_prof lockdown profile in the SALES application PDB. Then reset app\_root\_prof.

```
SQL> CONNECT hr_lock_mgr@hr_root
Enter password: password
Connected.
SQL> ALTER SYSTEM SET pdb_lockdown = '' SCOPE = BOTH;

System altered.

SQL>
```

- 2) Set the sales\_prof lockdown profile in the SALES application PDB.

```
SQL> CONNECT hr_lock_mgr@sales
Enter password: password
Connected.
SQL> ALTER SYSTEM SET pdb_lockdown = sales_prof SCOPE = BOTH;

System altered.

SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE          VALUE
-----                    -----        -----
pdb_lockdown              string        SALES_PROF
SQL>
```

- 3) Reset app\_root\_prof in the application root.

```
SQL> CONNECT hr_lock_mgr@hr_root
Enter password: password
Connected.
SQL> ALTER SYSTEM SET pdb_lockdown = app_root_prof
      SCOPE = BOTH;
2
System altered.

SQL>
```

- b. Observe how the SALES application PDB still inherits the restrictions of the application root lockdown profile and the CDB root lockdown profile.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: password
Connected.
SQL> ALTER PLUGGABLE DATABASE sales CLOSE;
ALTER PLUGGABLE DATABASE sales CLOSE
*
ERROR at line 1:
```

```

ORA-01031: insufficient privileges

SQL> ALTER SYSTEM SET ddl_lock_timeout=30 SCOPE = BOTH;

System altered.

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

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

SQL>

```

Because a lockdown profile is set in the application PDB, the application PDB inherits the restrictions of the application PDB lockdown profile and those of the CDB root lockdown profile and not from the application root lockdown profile.

## 6. Drop lockdown profiles.

- Drop the `CDB_prof1` lockdown profile in the CDB root.

```

SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET pdb_lockdown = '' SCOPE = BOTH;

System altered.

SQL> DROP LOCKDOWN PROFILE CDB_prof1;

Lockdown Profile dropped.

SQL> DROP LOCKDOWN PROFILE CDB_prof2;

Lockdown Profile dropped.

SQL>

```

- b. Drop the `sales_prof` lockdown profile in the `SALES` application PDB.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM SET pdb_lockdown = '' SCOPE = BOTH;

System altered.

SQL> CONNECT hr_lock_mgr@hr_root
Enter password: password
Connected.
SQL> DROP LOCKDOWN PROFILE sales_prof;

Lockdown Profile dropped.

SQL>
```

- c. Update the existing `app_root_prof` to remove `ALTER SYSTEM SET` restriction.

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

SQL>
```

7. In the second part of this practice, you will create static and dynamic lockdown profiles. You create and set, respectively, a static lockdown profile in the `SALES` application PDB from a basic application root lockdown profile and a dynamic lockdown profile in the `OPERATIONS` application PDB from the same basic application root lockdown profile.

- a. Log in to the application root to create the static application root lockdown profile `static_sales_prof` from the `app_root_prof` lockdown profile for the `SALES` application PDB.

```
SQL> CREATE LOCKDOWN PROFILE static_sales_prof
      FROM app_root_prof;
2
Lockdown Profile created.

SQL> ALTER LOCKDOWN PROFILE static_sales_prof
      DISABLE STATEMENT = ('alter pluggable database');
2
Lockdown Profile altered.
```

```

SQL> SELECT profile_name, rule, clause, status
  FROM cdb_lockdown_profiles;
2
PROFILE_NAME          RULE                      CLAUSE
-----
STATUS
-----
APP_ROOT_PROF         ALTER SYSTEM
DISABLE

APP_ROOT_PROF         ALTER SYSTEM          FLUSH SHARED_POOL
ENABLE

APP_ROOT_PROF         ALTER SYSTEM          SET
ENABLE

STATIC_SALES_PROF    ALTER PLUGGABLE DATABASE
DISABLE

STATIC_SALES_PROF    ALTER SYSTEM
DISABLE

STATIC_SALES_PROF    ALTER SYSTEM          FLUSH SHARED_POOL
ENABLE

STATIC_SALES_PROF    ALTER SYSTEM          SET
ENABLE

7 rows selected.

SQL>
```

*Q/ What do you observe about static\_sales\_prof in the CDB\_LOCKDOWN\_PROFILES view?*

**A/ The rules from the basic app\_root\_prof lockdown profile are all copied into the static\_sales\_prof lockdown profile.**

- b. Log in to the SALES application PDB to set the static static\_sales\_prof lockdown profile.

```
SQL> CONNECT hr_lock_mgr@sales
```

```
Enter password: password
```

```

Connected.

SQL> ALTER SYSTEM SET pdb_lockdown = '' SCOPE = BOTH;

System altered.

SQL> ALTER SYSTEM SET pdb_lockdown = static_sales_prof
      SCOPE = both;
2
System altered.

SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE        VALUE
-----
pdb_lockdown              string     STATIC_SALES_PROF
SQL>
```

- c. Before creating the dynamic lockdown profile in the OPERATIONS application PDB, log in to the application PDB to verify that you can still create a partitioned table because the restriction rule of the dynamic lockdown profile will be the partitioning feature.

```

SQL> CONNECT hr_lock_mgr@operations
Enter password: password
Connected.

SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE        VALUE
-----
pdb_lockdown              string     APP_ROOT_PROF
SQL>
SQL> CREATE TABLE sales ( SALESMAN_ID NUMBER(5),
                           SALESMAN_NAME VARCHAR2(30), SALES_STATE VARCHAR2(20))
      PARTITION BY LIST (SALES_STATE) AUTOMATIC
      (PARTITION P_CAL VALUES ('CALIFORNIA'));
2   3   4
Table created.

SQL> DROP TABLE sales;

Table dropped.

SQL>
```

- d. Log in to the application root to create the dynamic application root lockdown profile dynamic\_op\_prof from app\_root\_prof for the OPERATIONS application PDB.

```

SQL> CONNECT hr_lock_mgr@hr_root
Enter password: password
```

```
Connected.  
SQL> CREATE LOCKDOWN PROFILE dynamic_op_prof  
      INCLUDING app_root_prof;  
2  
Lockdown Profile created.  
  
SQL> ALTER LOCKDOWN PROFILE dynamic_op_prof  
      DISABLE OPTION = ('PARTITIONING');  
2  
Lockdown Profile altered.  
  
SQL> SELECT profile_name, rule, clause, status  
      FROM cdb_lockdown_profiles;  
2  
PROFILE_NAME          RULE                      CLAUSE  
-----  
STATUS  
-----  
APP_ROOT_PROF         ALTER SYSTEM  
DISABLE  
  
APP_ROOT_PROF         ALTER SYSTEM              FLUSH SHARED_POOL  
ENABLE  
  
APP_ROOT_PROF         ALTER SYSTEM              SET  
ENABLE  
  
DYNAMIC_OP_PROF      PARTITIONING  
DISABLE  
  
STATIC_SALES_PROF    ALTER PLUGGABLE DATABASE  
DISABLE  
  
STATIC_SALES_PROF    ALTER SYSTEM  
DISABLE  
  
STATIC_SALES_PROF    ALTER SYSTEM              FLUSH SHARED_POOL  
ENABLE  
  
STATIC_SALES_PROF    ALTER SYSTEM              SET  
ENABLE
```

```
8 rows selected.

SQL>
```

*Q/ What do you observe about dynamic\_op\_prof in the CDB\_LOCKDOWN\_PROFILES view?*

**A/ The rules from the basic app\_root\_prof lockdown profile are not copied into the dynamic\_op\_prof lockdown profile. You will check whether they are in effect.**

- e. Log in to the OPERATIONS application PDB to set the dynamic dynamic\_op\_prof lockdown profile.

```
SQL> CONNECT hr_lock_mgr@operations
Enter password: password
Connected.
SQL> ALTER SYSTEM SET pdb_lockdown = '' SCOPE = BOTH;

System altered.

SQL> ALTER SYSTEM SET pdb_lockdown = dynamic_op_prof
      SCOPE = BOTH;
2
System altered.

SQL> SHOW PARAMETER pdb_lockdown

NAME                      TYPE            VALUE
-----
pdb_lockdown              string          DYNAMIC_OP_PROF
SQL>
```

8. Test the behavior of static and dynamic lockdown profiles set in the SALES and OPERATIONS application PDBs, respectively.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM SET ddl_lock_timeout=30 SCOPE = BOTH;

System altered.

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 PLUGGABLE DATABASE CLOSE;
ALTER PLUGGABLE DATABASE CLOSE
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> CREATE TABLE sales
  ( SALESMAN_ID NUMBER(5) , SALESMAN_NAME VARCHAR2(30) ,
    SALES_STATE VARCHAR2(20) )
  PARTITION BY LIST (SALES_STATE) AUTOMATIC
  (PARTITION P_CAL VALUES ('CALIFORNIA'));
2      3      4      5
Table created.

SQL>

```

The first, second, and third commands respect rules inherited from `app_root_prof`. The fourth command respects rules inherited from `static_sales_prof`.

There is no restriction in `static_sales_prof` for the fifth command to execute.

9. Test in the `OPERATIONS` application PDB if the restriction rules of the dynamic `dynamic_op_prof` lockdown profile using the values from the existing basic `app_root_prof` lockdown profile apply.

```

SQL> CONNECT sys@operations AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM SET ddl_lock_timeout=30 SCOPE = BOTH;

System altered.

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 PLUGGABLE DATABASE CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE OPEN;

Pluggable database altered.

SQL> CREATE TABLE sales
      ( SALESMAN_ID NUMBER(5), SALESMAN_NAME VARCHAR2(30),
        SALES_STATE VARCHAR2(20))
      PARTITION BY LIST (SALES_STATE) AUTOMATIC
      (PARTITION P_CAL VALUES ('CALIFORNIA');
2      3      4      5
CREATE TABLE sales
*
ERROR at line 1:
ORA-00439: feature not enabled: Partitioning

SQL>

```

The first, second, and third commands respect rules inherited from `app_root_prof`.

There is no restriction in `dynamic_op_prof` for the fourth command to execute.

The fifth command respects rules inherited from `dynamic_op_prof`.

10. Test how any subsequent changes to the existing basic `app_root_prof` lockdown profile impact the derived `static_sales_prof` and `dynamic_op_prof` lockdown profiles. Add a rule to `app_root_prof`.

```

SQL> CONNECT hr_lock_mgr@hr_root
Enter password: password
Connected.
SQL> ALTER LOCKDOWN PROFILE app_root_prof
      ENABLE STATEMENT = ('alter system')
      CLAUSE = ('checkpoint');

```

```
2      3
Lockdown Profile altered.

SQL> SELECT profile_name, rule, clause, status
  FROM    cdb_lockdown_profiles;
2
PROFILE_NAME          RULE                      CLAUSE
-----
STATUS
-----
APP_ROOT_PROF         ALTER SYSTEM
DISABLE

APP_ROOT_PROF         ALTER SYSTEM          FLUSH SHARED_POOL
ENABLE

APP_ROOT_PROF         ALTER SYSTEM          SET
ENABLE

APP_ROOT_PROF         ALTER SYSTEM          CHECKPOINT
ENABLE

DYNAMIC_OP_PROF      PARTITIONING
DISABLE

STATIC_SALES_PROF    ALTER PLUGGABLE DATABASE
DISABLE

STATIC_SALES_PROF    ALTER SYSTEM
DISABLE

STATIC_SALES_PROF    ALTER SYSTEM          FLUSH SHARED_POOL
ENABLE

STATIC_SALES_PROF    ALTER SYSTEM          SET
ENABLE

9 rows selected.

SQL>
```

11. Check whether the rule added to the existing basic `app_root_prof` lockdown profile does not impact the derived `static_sales_prof`.

```
SQL> CONNECT sys@sales AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM CHECKPOINT;
ALTER SYSTEM CHECKPOINT
*
ERROR at line 1:
ORA-01031: insufficient privileges

SQL>
```

12. Check whether the rule added to the existing basic `app_root_prof` lockdown profile impacts the derived `dynamic_op_prof`.

```
SQL> CONNECT sys@operations AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM CHECKPOINT;

System altered.

SQL> CREATE TABLE sales
  (SALESMAN_ID NUMBER(5), SALESMAN_NAME VARCHAR2(30),
   SALES_STATE VARCHAR2(20))
  PARTITION BY LIST (SALES_STATE) AUTOMATIC
  (PARTITION P_CAL VALUES ('CALIFORNIA'));
2      3      4      5
CREATE TABLE sales
*
ERROR at line 1:
ORA-00439: feature not enabled: Partitioning

SQL> EXIT
$
```

13. Execute the `$HOME/labs/sec/cleanup_profiles.sh` shell script. It drops the `HR_ROOT` application root PDB and its associated application PDBs, `SALES` and `OPERATIONS`, and therefore, all lockdown profiles created in these PDBs.

```
$ $HOME/labs/sec/cleanup_profiles.sh
...
$
```

## Practice 7-6: 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> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> EXIT
$
```

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

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

2. Shut down all other instances in Oracle Database 18c.

```
$ pgrep -lf smon
13929 ora_smon_ORCL
25696 ora_smon_CDB18
31539 ora_smon_cdb12
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
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
$
```

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

```
$ $HOME/labs/sec/enable_unified_auditing.sh
*****
Stopping listener
*****  

LSNRCTL for Linux: Version 18.0.0.0.0 - Production on 30-APR-
2018 08:44:41  

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

Connecting to (ADDRESS=(PROTOCOL=tcp) (HOST=) (PORT=1521))
The command completed successfully
*****
Linking Oracle with Unified Auditing on
*****
/usr/bin/ar cr
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/libknlopt.a
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/kzaiang.o
chmod 755 /u01/app/oracle/product/18.1.0/dbhome_1/bin  

- Linking Oracle
rm -f /u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/oracle
/u01/app/oracle/product/18.1.0/dbhome_1/bin/orald -o
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/oracle -m64 -z
noexecstack -Wl,--disable-new-dtags -
L/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/ -
L/u01/app/oracle/product/18.1.0/dbhome_1/lib/ -
L/u01/app/oracle/product/18.1.0/dbhome_1/lib/stubs/ -Wl,-E
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/opimai.o
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/ssoraed.o
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/ttcsoi.o -Wl,-
-whole-archive -lperfsvr18 -Wl,--no-whole-archive
/u01/app/oracle/product/18.1.0/dbhome_1/lib/nautab.o
/u01/app/oracle/product/18.1.0/dbhome_1/lib/naeet.o
/u01/app/oracle/product/18.1.0/dbhome_1/lib/naect.o
/u01/app/oracle/product/18.1.0/dbhome_1/lib/naedhs.o
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/config.o -
ldmext -lserver18 -lodm18 -lofs -lcell18 -lnnet18 -lskgxp18 -
lsnls18 -lnls18 -lcore18 -lsnls18 -lnls18 -lcore18 -lsnls18 -
lnls18 -lxml18 -lcore18 -lunls18 -lsnls18 -lnls18 -lcore18 -
lnls18 -lclient18 -lvsnst18 -lcommon18 -lgeneric18 -lknlopt -
loraolap18 -lskjcx18 -lslax18 -lpls18 -lrt -lplp18 -ldmext -
lserver18 -lclient18 -lvsnst18 -lcommon18 -lgeneric18 `if [ -f
/u01/app/oracle/product/18.1.0/dbhome_1/lib/libavserver18.a ] ;
```

```

then echo "-lavserver18" ; else echo "-lavstub18"; fi` `if [ -f
/u01/app/oracle/product/18.1.0/dbhome_1/lib/libavclient18.a ] ;
then echo "-lavclient18" ; fi` -lknlopt -lslax18 -lpls18 -lrt -
lpplp18 -ljavavm18 -lserver18 -lwsg `cat
/u01/app/oracle/product/18.1.0/dbhome_1/lib/ldflags` -
lncrypt18 -lnsgr18 -lnzjs18 -ln18 -lnl18 -lngsmshd18 -lnro18
`cat /u01/app/oracle/product/18.1.0/dbhome_1/lib/ldflags` -
lncrypt18 -lnsgr18 -lnzjs18 -ln18 -lnl18 -lngsmshd18 -lnnzst18 -
lzt18 -lztkg18 -lmm -lsnls18 -lnls18 -lcore18 -lsnls18 -lnls18
-lcore18 -lsnls18 -lnls18 -lxml18 -lcore18 -lunls18 -lsnls18 -
lnls18 -lcore18 -lnls18 -lztkg18 `cat
/u01/app/oracle/product/18.1.0/dbhome_1/lib/ldflags` -
lncrypt18 -lnsgr18 -lnzjs18 -ln18 -lnl18 -lngsmshd18 -lnro18
`cat /u01/app/oracle/product/18.1.0/dbhome_1/lib/ldflags` -
lncrypt18 -lnsgr18 -lnzjs18 -ln18 -lnl18 -lngsmshd18 -lnnzst18 -
lzt18 -lztkg18 -lsnls18 -lnls18 -lcore18 -lsnls18 -lnls18 -
lcore18 -lsnls18 -lnls18 -lxml18 -lcore18 -lunls18 -lsnls18 -
lnls18 -lcore18 -lnls18 `if /usr/bin/ar tv
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/libknlopt.a |
grep "kxmnsd.o" > /dev/null 2>&1 ; then echo " " ; else echo "-"
lordsdo18 -lserver18"; fi` -
L/u01/app/oracle/product/18.1.0/dbhome_1/ctx/lib/ -lctxc18 -
lctx18 -lzx18 -lgx18 -lctx18 -lzx18 -lgx18 -lordimt -lclscest18
-loevm -lclsra18 -ldbcfg18 -lhasgen18 -lskgxn2 -lnnzst18 -lzt18
-lxml18 -lgeneric18 -locr18 -locrb18 -locrutil18 -lhasgen18 -
lskgxn2 -lnnzst18 -lzt18 -lxml18 -lgeneric18 -lgeneric18 -
lorazip -loraz -llzopro5 -lorabz2 -lipp_z -lipp_bz2 -
lippdcemerged -lippsemerged -lippdcmerged -lippmerged -
lippcore -lippcpemerged -lippcpmerged -lsnls18 -lnls18 -
lcore18 -lsnls18 -lnls18 -lcore18 -lsnls18 -lnls18 -lxml18 -
lcore18 -lunls18 -lsnls18 -lnls18 -lcore18 -lnls18 -lsnls18 -
lunls18 -lsnls18 -lnls18 -lcore18 -lsnls18 -lnls18 -lcore18 -
lsnls18 -lnls18 -lxml18 -lcore18 -lunls18 -lsnls18 -lnls18 -
lcore18 -lnls18 -lasmcnt18 -lcommon18 -lcore18 -ledtn18 -laio
-lons -lfthread18 `cat
/u01/app/oracle/product/18.1.0/dbhome_1/lib/sysliblist` -Wl,-
rpath,/u01/app/oracle/product/18.1.0/dbhome_1/lib -lm `cat
/u01/app/oracle/product/18.1.0/dbhome_1/lib/sysliblist` -ldl -lm
-L/u01/app/oracle/product/18.1.0/dbhome_1/lib `test -x
/usr/bin/hugeedit -a -r /usr/lib64/libhugetlbfs.so && test -r
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/shugetlbfs.o
&& echo -Wl,-zcommon-page-size=2097152 -Wl,-zmax-page-
size=2097152 -lhugetlbfs` 
rm -f /u01/app/oracle/product/18.1.0/dbhome_1/bin/oracle
mv /u01/app/oracle/product/18.1.0/dbhome_1/rdbms/lib/oracle
/u01/app/oracle/product/18.1.0/dbhome_1/bin/oracle
chmod 6751 /u01/app/oracle/product/18.1.0/dbhome_1/bin/oracle
...
$
```

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

```
$ . oraenv
ORACLE_SID = [CDB18] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

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

SQL>
```

5. 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/sec/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 host01.  
**Note:** Edit the script to make the application version number match your own version number.

```
SQL> @$HOME/labs/sec/script_grant.sql
...
SQL>
```

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

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: 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>

```

*Q/ Is the user\_toys\_pol audit policy common in the toys\_root application container?*

*A/ 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: 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(*)
-----
36

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

COUNT(*)
-----
36

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: 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(*)
-----
36
```

```

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

      COUNT (*)
-----
    72784

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

      COUNT (*)
-----
     36

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: 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: 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: 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
Enter password: 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: password
Connected.
SQL> NOAUDIT POLICY user_toys_pol;

Noaudit succeeded.

SQL> DROP AUDIT POLICY user_toys_pol;

Audit Policy dropped.

SQL>
```

6. You will audit the SELECT object privilege for any user connected in application PDBs in the toys\_root application container.
- 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/sec/script\_grant2.sql SQL script.  
**Note:** Edit the script to make the application version number match your own version number.

```
SQL> @$HOME/labs/sec/script_grant2.sql
...
SQL>
```

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

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: 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: 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 (*)
-----
36

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

COUNT (*)
-----
36

SQL>
```

```
SQL> CONNECT toys_owner@dolls
Enter password: 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 (*)
-----
36

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

COUNT (*)
-----
36

SQL>
```

- d. Check whether SELECT on the toys\_owner.sales\_data table is audited.

```
SQL> CONNECT system@toys_root
Enter password: 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: 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
Enter password: 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
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: password  
Connected.  
SQL> NOAUDIT POLICY user_toys_pol2;  
  
Noaudit succeeded.  
  
SQL> DROP AUDIT POLICY user_toys_pol2;  
  
Audit Policy dropped.  
  
SQL> EXIT  
$
```

## Practice 7-7: Protecting Application Common Objects with Database Vault Common Realms

### 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/sec/setup_DV_CDB.sh` shell script. For facilitating the application versions management, the script re-creates the `toys_app` application in the `toys_root` application container with a new application version.

```
$ $HOME/labs/sec/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: password
Connected.

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 password
      CONTAINER = all;
2
User created.

SQL> CREATE USER accts_admin IDENTIFIED BY password
      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: 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: 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: 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>

```

*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, 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: 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: 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: 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: 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: 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: password
Connected.
SQL> exec DVSYS.DBMS_MACADM.ENABLE_DV
PL/SQL procedure successfully completed.

SQL> CONNECT sys@dolls AS SYSDBA
Enter password: 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/sec/script\_populate.sql SQL script to insert rows in the toys\_owner.sales\_data table in robots. Then use \$HOME/labs/sec/script\_populate2.sql to insert rows in the toys\_owner.sales\_data table in dolls.

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

PL/SQL procedure successfully completed.

SQL> CONNECT toys_owner@dolls
Enter password: password
Connected.
SQL> @$HOME/labs/sec/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: 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 949
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 user being granted a 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> COL owner FORMAT A10
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: password
Connected.
SQL>

```

*Q/ 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 <code>toys_owner@toys_root</code>	CONNECT <code>test@toys_root</code>	SELECT * FROM
OK	OK	<code>toys_owner.sales_data</code>
OK	OK	<code>toys_owner.codes</code>

**A/ 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;
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	toys_owner.sales_data
ORA-01031	ORA-01031	toys_owner.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;
2
no rows selected

SQL>
```

5. Still in the Sec\_admin window, re-create the common realm in real mode.

```

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 DV$SYS.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 dv$sys.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	<code>toys_owner.sales_data</code>
ORA-01031	ORA-01031	<code>toys_owner.codes</code>

*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/sec/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/sec/disable_DV.sh  
...  
$ sqlplus / AS SYSDBA  
  
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: password
```

```
Connected.  
SQL> SELECT * FROM dba_dv_status;  
  
NAME          STATUS  
-----  
DV_CONFIGURE_STATUS  TRUE  
DV_ENABLE_STATUS    FALSE  
  
SQL> EXIT  
$
```

## Practice 7-8: Managing PDB Keystores

---

### Overview

In this practice, you will learn about the keystore isolated and united modes, and the reason for introducing these modes, and set up isolated mode PDB keystores.

### Tasks

1. You will use the Oracle by Example “Managing PDB Keystores” to see how to proceed. On VM2, launch a browser and click the bookmark from the Bookmarks Toolbar of the browser. The URL is the following:

[file:///home/oracle/labs/OBE/managing\\_pdb\\_keystores/managing\\_pdb\\_keystores.html](file:///home/oracle/labs/OBE/managing_pdb_keystores/managing_pdb_keystores.html)

## Practice 7-9: Unplugging and Plugging Encrypted PDBs

### 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/sec/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 PDB1.

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

2. Then verify that the master encryption key for PDB1 exists.

```
$ sqlplus sys@PDB1 AS SYSDBA
Enter password: password
Connected.
SQL> SELECT key_id, activating_pdbname, p.name
      FROM v$encryption_keys e, v$pdb$ p
     WHERE e.con_id = p.con_id;
      2   3
KEY_ID
-----
ACTIVATING_PDBNAME CON_ID
-----
AcTMJC/PjU9dvw+Br174qoUAAAAAAAAAAAAAAAAAAAAAAA
PDB1

SQL>
```

3. Create a table in PDB1 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 PDB1.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE pdb1 CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb1
      UNPLUG INTO '/tmp/PDB1.xml' ENCRYPT USING pass_plug;

Pluggable database altered.

SQL>
```

The *pass\_plug* is a temporary transport password that you define to protect the implicitly exported master keys.

```
SQL> DROP PLUGGABLE DATABASE pdb1 KEEP DATAFILES;

Pluggable database dropped.

SQL> EXIT
$
```

5. Plug the unplugged PDB into CDB18. If CDB18 was not restarted since practice 7-6 step 2, when it was shut down, start it up.

```
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> startup
ORACLE instance started.

Total System Global Area 1426062800 bytes
Fixed Size                  8895952 bytes
Variable Size                486539264 bytes
Database Buffers             922746880 bytes
Redo Buffers                 7880704 bytes
```

```

Database mounted.
Database opened.
SQL> CREATE PLUGGABLE DATABASE pdb_encrypt
      USING '/tmp/PDB1.xml' NOCOPY
      KEYSTORE IDENTIFIED BY password
      DECRYPT USING pass_plug;
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 is required to store the master keys.*

*Q2/ Can the pass\_plug and password be different?*

*A2/ Yes. The pass\_plug is the temporary transport password that you defined to protect the exported master keys at unplug time, which must be reused during the plug operation. The password is the password to access the CDB root keystore.*

```

$ $HOME/labs/sec/enable_TDE_in_CDB18.sh
...
$
$ sqlplus / AS SYSDBA

SQL> CREATE PLUGGABLE DATABASE pdb_encrypt
      USING '/tmp/PDB1.xml' NOCOPY
      KEYSTORE IDENTIFIED BY password
      DECRYPT USING pass_plug;
2     3     4
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb_encrypt OPEN;

Pluggable database created.

SQL> SELECT key_id, activating_pdbname, p.name
      FROM v$encryption_keys e, v$pdbs p
```

```
WHERE e.con_id = p.con_id;
      3
KEY_ID
-----
ACTIVATING_PDBNAME NAME
-----
AV3w5LaRjk/Av7IZ1qjnWcAAAAAAAAAAAAAAA
PDB_ENCRYPT          PDB_ENCRYPT
SQL>
```

6. Verify that the encrypted data is readable.

```
SQL> CONNECT system@pdb_encrypt
Enter password: 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 PDB1. The keystore for the CDB was opened before the PDB was plugged in.*

```
SQL> CONNECT sys@pdb_encrypt AS SYSDBA
Enter password: password
Connected.
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
```

```
IDENTIFIED BY password CONTAINER = current;
2
keystore altered.

SQL> SELECT wrl_parameter, status, wallet_type
   FROM v$encryption_wallet;
2
WRL_PARAMETER                      STATUS      WALLET_T
-----
OPEN                  PASSWORD

SQL> CONNECT system@pdb_encrypt
Enter password: password
Connected.
SQL> SELECT * FROM l_user.test;

C
-----
1

SQL> EXIT
$
```

7. Drop PDB\_ENCRYPT using the \$HOME/labs/sec/cleanup\_TDE\_ENCRYPT.sh script.

```
$ $HOME/labs/sec/cleanup_TDE_ENCRYPT.sh
...
$
```

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.

## **Practices for Lesson 8: Backup and Duplicate**

## Practices for Lesson 8: Overview

---

### Practices Overview

In the following practices, you will perform backup and duplicate operations on the CDB and PDBs.

- RMAN ORCL backup
- RMAN regular PDB1 backup
- RMAN application toys\_root backup
- PDB duplicate into an active CDB
- On-premise CDB duplicate for Cloud migrate/on

## Practice 8-1: RMAN Whole CDB Backup

---

### Overview

In this practice, you will perform a whole CDB backup of ORCL.

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. Then execute the `$HOME/labs/BD/login_8.sh`, `$HOME/labs/BD/setup_toys_app.sh`, and `$HOME/labs/BD/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/BD/glogin_8.sh
$ $HOME/labs/BD/setup_toys_app.sh
...
$ $HOME/labs/BD/backup.sh
...
$
```

3. Run RMAN to connect to ORCL with a user with SYSDBA or SYSBACKUP privilege.

```
$ . oraenv
ORACLE_SID = [CDB18] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ export NLS_DATE_FORMAT='DD-MM-YYYY HH:MI:SS'
$ rman target /

connected to target database: ORCL (DBID=1461390531)
RMAN>
```

4. As usual, back up all datafiles of the database (the CDB root and all PDBs), control files and SPFILE, and archive log files, after setting the `db_recovery_file_dest_size` to 18 GB.

```
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO disk;
using target database control file instead of recovery catalog
new RMAN configuration parameters:
CONFIGURE DEFAULT DEVICE TYPE TO DISK;
new RMAN configuration parameters are successfully stored
```

```
RMAN> ALTER SYSTEM SET db_recovery_file_dest_size=18G
SCOPE=BOTH;

Statement processed

RMAN> BACKUP DATABASE PLUS ARCHIVELOG;

Starting backup at 03-05-2018 10:51:09
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=39 device type=DISK
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=916 RECID=915
STAMP=975190562
input archived log thread=1 sequence=917 RECID=916
STAMP=975190704
input archived log thread=1 sequence=918 RECID=917
STAMP=975190788
input archived log thread=1 sequence=919 RECID=918
STAMP=975190889
input archived log thread=1 sequence=920 RECID=919
STAMP=975192255
input archived log thread=1 sequence=921 RECID=920
STAMP=975192257
input archived log thread=1 sequence=922 RECID=921
STAMP=975192541
input archived log thread=1 sequence=923 RECID=922
STAMP=975192669
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:51:10
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:51:17
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05
_03/o1_mf_annnn_TAG20180503T225110_fgq4lyh0_.bkp
tag=TAG20180503T225110 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 03-05-2018 10:51:17

Starting backup at 03-05-2018 10:51:17
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00003
name=/u02/app/oracle/oradata/ORCL/sysaux01.dbf
```

```
input datafile file number=00001
name=/u02/app/oracle/oradata/ORCL/system01.dbf
input datafile file number=00004
name=/u02/app/oracle/oradata/ORCL/undotbs01.dbf
input datafile file number=00116
name=/u02/app/oracle/oradata/ORCL/cdata_01.dbf
input datafile file number=00007
name=/u02/app/oracle/oradata/ORCL/users01.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:51:17
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:54:43
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05
_03/o1_mf_nnndf_TAG20180503T225117_fgg4m6os_.bkp
tag=TAG20180503T225117 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:03:26
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00119
name=/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B55C50D8DAE36F
5E0532133960ABBA7/datafile/o1_mf_sysaux_fgg3gw04_.dbf
input datafile file number=00118
name=/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B55C50D8DAE36F
5E0532133960ABBA7/datafile/o1_mf_system_fgg3gw01_.dbf
input datafile file number=00120
name=/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B55C50D8DAE36F
5E0532133960ABBA7/datafile/o1_mf_undotbs1_fgg3gw05_.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:54:43
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:54:58
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B55C50D8DAE36F5E
0532133960ABBA7/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
117_fgg4sn00_.bkp tag=TAG20180503T225117 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00122
name=/u02/app/oracle/oradata/ORCL/toys_root/robots/ORCL/6B55C70C
6B513743E0532133960AA917/datafile/o1_mf_sysaux_fgg3hro5_.dbf
input datafile file number=00121
name=/u02/app/oracle/oradata/ORCL/toys_root/robots/ORCL/6B55C70C
6B513743E0532133960AA917/datafile/o1_mf_system_fgg3hro4_.dbf
input datafile file number=00123
name=/u02/app/oracle/oradata/ORCL/toys_root/robots/ORCL/6B55C70C
6B513743E0532133960AA917/datafile/o1_mf_undotbs1_fgg3hro6_.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:54:59
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:55:14
```

```
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B55C70C6B513743E
0532133960AA917/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
117_fq4t3bg_.bkp tag=TAG20180503T225117 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00125
name=/u02/app/oracle/oradata/ORCL/toys_root/dolls/ORCL/6B55C70C6
B543743E0532133960AA917/datafile/o1_mf_sysaux_fq3jfl4_.dbf
input datafile file number=00124
name=/u02/app/oracle/oradata/ORCL/toys_root/dolls/ORCL/6B55C70C6
B543743E0532133960AA917/datafile/o1_mf_system_fq3jfl0_.dbf
input datafile file number=00126
name=/u02/app/oracle/oradata/ORCL/toys_root/dolls/ORCL/6B55C70C6
B543743E0532133960AA917/datafile/o1_mf_undotbs1_fq3jfl4_.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:55:14
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:55:29
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B55C70C6B543743E
0532133960AA917/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
117_fq4tlkq_.bkp tag=TAG20180503T225117 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00006
name=/u02/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf
input datafile file number=00005
name=/u02/app/oracle/oradata/ORCL/pdbseed/system01.dbf
input datafile file number=00008
name=/u02/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:55:29
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:55:45
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/675B8BA1AD892F49E
0532133960AAB76/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
117_fq4v1st_.bkp tag=TAG20180503T225117 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:16
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00128
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B55EB88BE0F3BB8E053
2133960AB60C/datafile/o1_mf_sysaux_fq42yy3_.dbf
input datafile file number=00127
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B55EB88BE0F3BB8E053
2133960AB60C/datafile/o1_mf_system_fq42yxz_.dbf
```

```
input datafile file number=00129
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B55EB88BE0F3BB8E053
2133960AB60C/datafile/o1_mf_undotbs1_fgg42yy4_.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:55:45
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:56:00
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B55EB88BE0F3BB8E
0532133960AB60C/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
117_fgg4vko4_.bkp tag=TAG20180503T225117 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 03-05-2018 10:56:00

Starting backup at 03-05-2018 10:56:00
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=924 RECID=923
STAMP=975192961
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:56:01
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:56:03
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05_
03/o1_mf_annnn_TAG20180503T225601_fgg4w1vx_.bkp
tag=TAG20180503T225601 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:02
Finished backup at 03-05-2018 10:56:03

Starting Control File and SPFILE Autobackup at 03-05-2018
10:56:03
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
5_03/o1_mf_s_975192963_fgg4w4vd_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 03-05-2018
10:56:06

RMAN>
```

## Practice 8-2: RMAN PDB Backup

---

### Overview

In this practice, you will perform the regular PDB backup of PDB1 and then the application PDB root backup of TOYS\_ROOT.

### Tasks

1. Perform the PDB1 backup. A new RMAN command allows you to back up all datafiles of the PDB.

```
RMAN> BACKUP PLUGGABLE DATABASE pdb1;

Starting backup at 03-05-2018 10:56:54
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00128
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B55EB88BE0F3BB8E053
2133960AB60C/datafile/o1_mf_sysaux_fgg42yy3_.dbf
input datafile file number=00127
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B55EB88BE0F3BB8E053
2133960AB60C/datafile/o1_mf_system_fgg42yxz_.dbf
input datafile file number=00129
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B55EB88BE0F3BB8E053
2133960AB60C/datafile/o1_mf_undotbs1_fgg42yy4_.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:56:55
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:57:11
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B55EB88BE0F3BB8E
0532133960AB60C/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
655_fgg4xqhg_.bkp tag=TAG20180503T225655 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:16
Finished backup at 03-05-2018 10:57:11

Starting Control File and SPFILE Autobackup at 03-05-2018
10:57:11
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
5_03/o1_mf_s_975193031_fgg4y885_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 03-05-2018
10:57:14

RMAN>
```

2. Perform an application PDB backup of the `toys_root` application container.

```
RMAN> BACKUP PLUGGABLE DATABASE toys_root;

Starting backup at 03-05-2018 10:57:34
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00119
name=/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B55C50D8DAE36F
5E0532133960ABBA7/datafile/o1_mf_sysaux_fgq3gw04_.dbf
input datafile file number=00118
name=/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B55C50D8DAE36F
5E0532133960ABBA7/datafile/o1_mf_system_fgq3gw01_.dbf
input datafile file number=00120
name=/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B55C50D8DAE36F
5E0532133960ABBA7/datafile/o1_mf_undotbs1_fgq3gw05_.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:57:35
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:57:50
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B55C50D8DAE36F5E
0532133960ABBA7/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
734_fgq4yzg5_.bkp tag=TAG20180503T225734 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 03-05-2018 10:57:50

Starting Control File and SPFILE Autobackup at 03-05-2018
10:57:50
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
5_03/o1_mf_s_975193070_fgq4zhnf_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 03-05-2018
10:57:53

RMAN>
```

*Q/ Does the backup include the application PDBs associated to the application root?*

**A/ No. Therefore, you should back up the application PDBs too.**

```
RMAN> BACKUP PLUGGABLE DATABASE robots;

Starting backup at 03-05-2018 10:57:56
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
```

```

input datafile file number=00122
name=/u02/app/oracle/oradata/ORCL/toys_root/robots/ORCL/6B55C70C
6B513743E0532133960AA917/datafile/o1_mf_sysaux_fqg3hro5_.dbf
input datafile file number=00121
name=/u02/app/oracle/oradata/ORCL/toys_root/robots/ORCL/6B55C70C
6B513743E0532133960AA917/datafile/o1_mf_system_fqg3hro4_.dbf
input datafile file number=00123
name=/u02/app/oracle/oradata/ORCL/toys_root/robots/ORCL/6B55C70C
6B513743E0532133960AA917/datafile/o1_mf_undotbs1_fqg3hro6_.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:57:58
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:58:13
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B55C70C6B513743E
0532133960AA917/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
756_fgq4zpk1_.bkp tag=TAG20180503T225756 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 03-05-2018 10:58:13

Starting Control File and SPFILE Autobackup at 03-05-2018
10:58:13
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
5_03/o1_mf_s_975193093_fgq506h7_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 03-05-2018
10:58:16

RMAN> BACKUP PLUGGABLE DATABASE dolls;

Starting backup at 03-05-2018 10:58:19
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00125
name=/u02/app/oracle/oradata/ORCL/toys_root/dolls/ORCL/6B55C70C6
B543743E0532133960AA917/datafile/o1_mf_sysaux_fqg3jf14_.dbf
input datafile file number=00124
name=/u02/app/oracle/oradata/ORCL/toys_root/dolls/ORCL/6B55C70C6
B543743E0532133960AA917/datafile/o1_mf_system_fqg3jf10_.dbf
input datafile file number=00126
name=/u02/app/oracle/oradata/ORCL/toys_root/dolls/ORCL/6B55C70C6
B543743E0532133960AA917/datafile/o1_mf_undotbs1_fgq3jf14_.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:58:20
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:58:35
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B55C70C6B543743E
0532133960AA917/backupset/2018_05_03/o1_mf_nnndf_TAG20180503T225
819_fgq50dd3_.bkp tag=TAG20180503T225819 comment=NONE

```

```

channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 03-05-2018 10:58:35

Starting Control File and SPFILE Autobackup at 03-05-2018
10:58:35
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
5_03/o1_mf_s_975193115_fgg50wjs_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 03-05-2018
10:58:38

RMAN>

```

3. Perform a partial PDB backup. Back up the users tablespace of the CDB root.

```

RMAN> BACKUP TABLESPACE "CDB$ROOT":users;

Starting backup at 03-05-2018 10:58:41
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00007
name=/u02/app/oracle/oradata/ORCL/users01.dbf
channel ORA_DISK_1: starting piece 1 at 03-05-2018 10:58:41
channel ORA_DISK_1: finished piece 1 at 03-05-2018 10:58:43
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05
_03/o1_mf_nnndf_TAG20180503T225841_fgg51235_.bkp
tag=TAG20180503T225841 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:02
Finished backup at 03-05-2018 10:58:43

Starting Control File and SPFILE Autobackup at 03-05-2018
10:58:43
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
5_03/o1_mf_s_975193123_fgg514ps_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 03-05-2018
10:58:46

RMAN> EXIT
$
```

## Practice 8-3: Duplicating a PDB into an Existing CDB

### Overview

In this practice, you will clone PDB1 from ORCL into CDB18 by using the RMAN DUPLICATE command without the use of a fresh auxiliary instance. As it is not easy or recommended to handle two PDBs with the same name in two different CDBs, duplicate PDB1 as PDB1\_IN\_CDB18 in CDB18.

### Tasks

1. Execute the \$HOME/labs/BD/HR.sh shell script that creates the HR schema tables in PDB1. It starts CDB18 up and also drops PDB1\_IN\_CDB18 in case it exists in CDB18.

```
$ $HOME/labs/BD/HR.sh
...
$
```

2. Before duplicating PDB1, check that the HR.REGIONS table contains rows.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus hr@PDB1
Enter password: password

SQL> SELECT * FROM regions;

REGION_ID REGION_NAME
-----
1 Europe
2 Americas
3 Asia
4 Middle East and Africa

SQL> EXIT
$
```

3. In the destination CDB, set the initialization parameter REMOTE\_RECOVERY\_FILE\_DEST to restore foreign archive logs.

```
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER SYSTEM SET
REMOTE_RECOVERY_FILE_DEST='/home/oracle/labs' SCOPE=BOTH;
```

```
System altered.
```

```
SQL> EXIT
$
```

- Open an rman session and connect to the source ORCL CDB root.

```
$ rman

RMAN> CONNECT TARGET sys@ORCL
target database Password: password
connected to target database: ORCL (DBID=1504077170)

RMAN>
```

- Connect to the auxiliary instance CDB18 into which you will duplicate PDB1 as PDB1\_IN\_CDB18.

```
RMAN> CONNECT AUXILIARY sys@CDB18

auxiliary database Password: password
connected to auxiliary database: CDB18 (DBID=1938746784)

RMAN>
```

- Duplicate PDB1 as PDB1\_IN\_CDB18 in CDB18.

```
RMAN> DUPLICATE PLUGGABLE DATABASE pdb1 AS pdb1_in_cdb18
      FROM ACTIVE DATABASE
      DB_FILE_NAME_CONVERT ('ORCL', 'CDB18');
2> 3>
Starting Duplicate PDB at 03-05-2018 11:02:43
using target database control file instead of recovery catalog
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=46 device type=DISK
current log archived

contents of Memory Script:
{
  set newname for datafile 127 to
"/u02/app/oracle/oradata/CDB18/pdb1/CDB18/6B55EB88BE0F3BB8E05321
33960AB60C/datafile/o1_mf_system_fgq42yzx_.dbf";
  set newname for datafile 128 to
"/u02/app/oracle/oradata/CDB18/pdb1/CDB18/6B55EB88BE0F3BB8E05321
33960AB60C/datafile/o1_mf_sysaux_fgq42yy3_.dbf";
  set newname for datafile 129 to
```

```
"/u02/app/oracle/oradata/CDB18/pdb1/CDB18/6B55EB88BE0F3BB8E05321  
3960AB60C/datafile/o1_mf_undotbs1_fgq42yy4_.dbf";  
    restore  
    from nonparse    clone foreign pluggable database  
    "PDB1"  
    from service  'ORCL'      ;  
}  
executing Memory Script  
  
executing command: SET NEWNAME  
  
executing command: SET NEWNAME  
  
executing command: SET NEWNAME  
  
Starting restore at 03-05-2018 11:02:50  
using channel ORA_AUX_DISK_1  
  
channel ORA_AUX_DISK_1: starting datafile backup set restore  
channel ORA_AUX_DISK_1: using network backup set from service  
ORCL  
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from  
backup set  
channel ORA_AUX_DISK_1: restoring foreign file 127 to  
/u02/app/oracle/oradata/CDB18/pdb1/CDB18/6B55EB88BE0F3BB8E053213  
3960AB60C/datafile/o1_mf_system_fgq42yxz_.dbf  
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:07  
channel ORA_AUX_DISK_1: starting datafile backup set restore  
channel ORA_AUX_DISK_1: using network backup set from service  
ORCL  
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from  
backup set  
channel ORA_AUX_DISK_1: restoring foreign file 128 to  
/u02/app/oracle/oradata/CDB18/pdb1/CDB18/6B55EB88BE0F3BB8E053213  
3960AB60C/datafile/o1_mf_sysaux_fgq42yy3_.dbf  
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:15  
channel ORA_AUX_DISK_1: starting datafile backup set restore  
channel ORA_AUX_DISK_1: using network backup set from service  
ORCL  
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from  
backup set  
channel ORA_AUX_DISK_1: restoring foreign file 129 to  
/u02/app/oracle/oradata/CDB18/pdb1/CDB18/6B55EB88BE0F3BB8E053213  
3960AB60C/datafile/o1_mf_undotbs1_fgq42yy4_.dbf  
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:03
```

```
Finished restore at 03-05-2018 11:03:18
current log archived

contents of Memory Script:
{
  set archivelog destination to '/home/oracle/labs';
  restore clone force from service 'ORCL'
    foreign archivelog from scn 8894480;
}
executing Memory Script

executing command: SET ARCHIVELOG DESTINATION

Starting restore at 03-05-2018 11:03:22
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting archived log restore to user-
specified destination
archived log destination=/home/oracle/labs
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: restoring archived log
archived log thread=1 sequence=926
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_AUX_DISK_1: starting archived log restore to user-
specified destination
archived log destination=/home/oracle/labs
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: restoring archived log
archived log thread=1 sequence=927
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
Finished restore at 03-05-2018 11:03:24

Performing import of metadata...
Finished Duplicate PDB at 03-05-2018 11:03:43

RMAN> EXIT
$
```

Observe that the initialization parameter `REMOTE_RECOVERY_FILE_DEST` was used to restore foreign archive log.

7. Check that the duplicated PDB holds the HR schema tables.

```
$ sqlplus / AS SYSDBA

SQL> SHOW PDBS

CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
 2 PDB$SEED           READ ONLY NO
 3 PDB18              READ WRITE NO
 4 PDB1_IN_CDB18     READ WRITE NO

SQL> ALTER SESSION SET CONTAINER=PDB1_IN_CDB18;

Session altered.

SQL> SELECT * FROM hr.regions;

REGION_ID REGION_NAME
-----
 1 Europe
 2 Americas
 3 Asia
 4 Middle East and Africa

SQL>
```

8. Drop PDB1\_IN\_CDB18 using the \$HOME/labs/BD/drop\_pdb1\_in\_cdb18.sql script.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> @$HOME/labs/BD/drop_pdb1_in_cdb18.sql
...
$
```

## Practice 8-4: Duplicating an On-Premises CDB for Cloud

---

### Overview

In this practice, you will duplicate ORCL with encrypted tablespaces to be compatible on Cloud. Duplication as encrypted or decrypted aids with the transition from on-premises databases to Cloud and vice versa.

### Tasks

1. Execute the \$HOME/labs/BD/tbspdb1\_enc.sh shell script to create an encrypted tablespace in the PDB1.

```
$ $HOME/labs/BD/tbspdb1_enc.sh
...
$
```

2. Verify that the TBSORCL\_ENC tablespace has been created as encrypted.

```
$ . oraenv
ORACLE_SID = [CDB18] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus system@PDB1
Enter password: password

SQL> SELECT tablespace_name, encrypted FROM dba_tablespaces;

TABLESPACE_NAME          ENCRYPTED
-----  -----
SYSTEM                  NO
SYSAUX                 NO
UNDOTBS1                NO
TEMP                   NO
TBSORCL_ENC             YES

SQL>
```

3. Prepare ORCL CDB to be duplicated as O\_CLOUD CDB.

- a. Copy the ORCL pfile for O\_CLOUD. First connect to the CDB root.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> CREATE PFILE FROM SPFILE;

File created.

SQL> EXIT
$
```

- b. Copy the ORCL pfile for O\_CLOUD.

```
$ cd $ORACLE_HOME/dbs
$ cp initORCL.ora initO_CLOUD.ora
$
```

- c. Edit the initO\_CLOUD PFILE.

- 1) Remove all entries that start with ORCL.\_.
- 2) Substitute all ORCL entries to O\_CLOUD.
- 3) Add the ENCRYPT\_NEW\_TABLESPACES parameter and set it to ALWAYS to simulate the Cloud CDB environment if you were migrating the on-premises CDB to Cloud.
- 4) Add two parameters

```
DB_FILE_NAME_CONVERT=(/u02/app/oracle/oradata/ORCL/, /u02/app/
oracle/oradata/O_CLOUD/)
LOG_FILE_NAME_CONVERT=(/u04/app/oracle/redo/ORCL/, /u04/app/or
acle/redo/O_CLOUD/)
```

- d. Create the TDE keystore directory for O\_CLOUD.

```
$ mkdir -p /u02/app/oracle/admin/O_CLOUD/tde_wallet
$
```

- e. Copy the ORCL keystore for O\_CLOUD to the directory.

```
$ cp /u02/app/oracle/admin/ORCL/tde_wallet/ewallet.p12
/u02/app/oracle/admin/O_CLOUD/tde_wallet
$
```

- f. Create the directories required for O\_CLOUD instance to start.

```
$ mkdir -p /u02/app/oracle/oradata/O_CLOUD
$ mkdir -p /u01/app/oracle/admin/O_CLOUD/adump
$ mkdir -p /u03/app/oracle/fast_recovery_area/O_CLOUD
$ mkdir -p /u04/app/oracle/redo/O_CLOUD
$
```

- g. Create the password file for O\_CLOUD with the same password as for ORCL.

```
$ orapwd file=$ORACLE_HOME/dbs/orapwO_CLOUD password=password
entries=5
$
```

- h. Set the ORACLE\_SID environment variable to O\_CLOUD to start a connection into O\_CLOUD.

```
$ export ORACLE_SID=O_CLOUD
$ sqlplus / AS SYSDBA
SQL>
```

- i. Create the SPFILE from PFILE for O\_CLOUD.

```
SQL> CREATE SPFILE FROM PFILE;
File created.
SQL>
```

- j. Start the instance.

```
SQL> STARTUP NOMOUNT
ORACLE instance started.

Total System Global Area 1426062800 bytes
Fixed Size                  8895952 bytes
Variable Size                486539264 bytes
Database Buffers            922746880 bytes
Redo Buffers                 7880704 bytes
SQL>
```

- k. Open the O\_CLOUD TDE keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE
      OPEN IDENTIFIED BY password CONTAINER = ALL;
2
keystore altered.

SQL> EXIT
$
```

4. Duplicate ORCL holding in PDB1 an encrypted tablespace to O\_CLOUD.

- a. Launch rman to create a connection to the auxiliary CDB instance that will be the new duplicated O\_CLOUD CDB and to the source ORCL that will be duplicated.

```
$ rman AUXILIARY sys TARGET sys@ORCL

target database Password: password
connected to target database: ORCL (DBID=1498512531)
auxiliary database Password: password
connected to auxiliary database: O_CLOUD (not mounted)

RMAN>
```

- b. Before starting the duplication, set the decryption on. During the duplicate operation, if the auxiliary instance is using a keystore that is not autologin, you need to supply the password for the keystore using the SET DECRYPTION command. This is because the backups are encrypted using the keystore on the source CDB, and those backups need to be decrypted in order for the resource to work. Similarly, the media recovery needs to apply redo, which could also be encrypted. For those operations to succeed, the auxiliary keystore needs to be opened during the duplicate. By supplying the password of the keystore in duplicate command, the duplicate command opens the keystore automatically after the auxiliary is bounced during the duplicate process.

```
RMAN> SET DECRYPTION WALLET OPEN IDENTIFIED BY password;
executing command: SET decryption
using target database control file instead of recovery catalog

RMAN>
```

c. Start the duplication.

```
RMAN> DUPLICATE TARGET DATABASE TO o_cloud
      FROM ACTIVE DATABASE
      DB_FILE_NAME_CONVERT ('ORCL', 'O_CLOUD');
2> 3>

Starting Duplicate Db at 04-MAY-18
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=21 device type=DISK
current log archived

contents of Memory Script:
{
  sql clone "alter system set db_name =
  ''ORCL'' comment=
  ''Modified by RMAN duplicate'' scope=spfile";
  sql clone "alter system set db_unique_name =
  ''O_CLOUD'' comment=
  ''Modified by RMAN duplicate'' scope=spfile";
  shutdown clone immediate;
  startup clone force nomount
  restore clone from service 'ORCL' primary controlfile;
  alter clone database mount;
}
executing Memory Script

sql statement: alter system set db_name = ''ORCL'' comment=
  ''Modified by RMAN duplicate'' scope=spfile

sql statement: alter system set db_unique_name = ''O_CLOUD''
comment= ''Modified by RMAN duplicate'' scope=spfile

Oracle instance shut down

Oracle instance started

Total System Global Area    1426062800 bytes
```

```
Fixed Size                      8895952 bytes
Variable Size                   486539264 bytes
Database Buffers                922746880 bytes
Redo Buffers                     7880704 bytes

Starting restore at 04-MAY-18
decryption password set for AUXILIARY database
allocated channel: ORA_AUX_DISK_1
channel ORA_AUX_DISK_1: SID=21 device type=DISK

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: restoring control file
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:05
output file name=/u02/app/oracle/oradata/O_CLOUD/control01.ctl
output file
name=/u03/app/oracle/fast_recovery_area/O_CLOUD/control02.ctl
Finished restore at 04-MAY-18

database mounted

contents of Memory Script:
{
  set newname for datafile 1 to
  "/u02/app/oracle/oradata/O_CLOUD/system01.dbf";
  set newname for datafile 3 to
  "/u02/app/oracle/oradata/O_CLOUD/sysaux01.dbf";
  set newname for datafile 4 to
  "/u02/app/oracle/oradata/O_CLOUD/undotbs01.dbf";
  set newname for datafile 5 to
  "/u02/app/oracle/oradata/O_CLOUD/pdbseed/system01.dbf";
  set newname for datafile 6 to
  "/u02/app/oracle/oradata/O_CLOUD/pdbseed/sysaux01.dbf";
  set newname for datafile 7 to
  "/u02/app/oracle/oradata/O_CLOUD/users01.dbf";
  set newname for datafile 8 to
  "/u02/app/oracle/oradata/O_CLOUD/pdbseed/undotbs01.dbf";
  set newname for datafile 116 to
  "/u02/app/oracle/oradata/O_CLOUD/cdata_01.dbf";
  set newname for datafile 141 to
```

```
"/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE0
532133960A63DF/datafile/o1_mf_system_fgg6bkw9_.dbf";
    set newname for datafile 142 to

"/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE0
532133960A63DF/datafile/o1_mf_sysaux_fgg6bkwb_.dbf";
    set newname for datafile 143 to

"/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE0
532133960A63DF/datafile/o1_mf_undotbs1_fgg6bkwb_.dbf";
    set newname for datafile 144 to
"/u02/app/oracle/oradata/O_CLOUD/pdb1/tbsenc01.dbf";
    restore
      from nspars e from service
'ORCL' clone database
;
sql 'alter system archive log current';
}
executing Memory Script

executing command: SET NEWNAME

executing command: SET NEWNAME
```

```
Starting restore at 04-MAY-18
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00001 to
/u02/app/oracle/oradata/O_CLOUD/system01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:29
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00003 to
/u02/app/oracle/oradata/O_CLOUD/sysaux01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:55
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00004 to
/u02/app/oracle/oradata/O_CLOUD/undotbs01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:11
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00005 to
/u02/app/oracle/oradata/O_CLOUD/pdbseed/system01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:11
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00006 to
/u02/app/oracle/oradata/O_CLOUD/pdbseed/sysaux01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:18
channel ORA_AUX_DISK_1: starting datafile backup set restore
```

```
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00007 to
/u02/app/oracle/oradata/O_CLOUD/users01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00008 to
/u02/app/oracle/oradata/O_CLOUD/pdbseed/undotbs01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:06
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00116 to
/u02/app/oracle/oradata/O_CLOUD/cdata_01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00141 to
/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE05
32133960A63DF/datafile/o1_mf_system_fgq6bkw9_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:11
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00142 to
/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE05
32133960A63DF/datafile/o1_mf_sysaux_fgq6bkwb_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:10
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
```

```
channel ORA_AUX_DISK_1: restoring datafile 00143 to
/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE05
32133960A63DF/datafile/o1_mf_undotbs1_fgq6bkwb_.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:06
channel ORA_AUX_DISK_1: starting datafile backup set restore
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_AUX_DISK_1: restoring datafile 00144 to
/u02/app/oracle/oradata/O_CLOUD/pdb1/tbsenc01.dbf
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:03
Finished restore at 04-MAY-18

sql statement: alter system archive log current
current log archived

contents of Memory Script:
{
    restore clone force from service 'ORCL'
        archivelog from scn 8922927;
    switch clone datafile all;
}
executing Memory Script

Starting restore at 04-MAY-18
using channel ORA_AUX_DISK_1

channel ORA_AUX_DISK_1: starting archived log restore to default
destination
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: restoring archived log
archived log thread=1 sequence=942
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:02
channel ORA_AUX_DISK_1: starting archived log restore to default
destination
channel ORA_AUX_DISK_1: using network backup set from service
ORCL
channel ORA_AUX_DISK_1: restoring archived log
archived log thread=1 sequence=943
channel ORA_AUX_DISK_1: restore complete, elapsed time: 00:00:01
Finished restore at 04-MAY-18
```

```

datafile 1 switched to datafile copy
input datafile copy RECID=16 STAMP=975222895 file
name=/u02/app/oracle/oradata/O_CLOUD/system01.dbf
datafile 3 switched to datafile copy
input datafile copy RECID=17 STAMP=975222896 file
name=/u02/app/oracle/oradata/O_CLOUD/sysaux01.dbf
datafile 4 switched to datafile copy
input datafile copy RECID=18 STAMP=975222897 file
name=/u02/app/oracle/oradata/O_CLOUD/undotbs01.dbf
datafile 5 switched to datafile copy
input datafile copy RECID=19 STAMP=975222897 file
name=/u02/app/oracle/oradata/O_CLOUD/pdbseed/system01.dbf
datafile 6 switched to datafile copy
input datafile copy RECID=20 STAMP=975222898 file
name=/u02/app/oracle/oradata/O_CLOUD/pdbseed/sysaux01.dbf
datafile 7 switched to datafile copy
input datafile copy RECID=21 STAMP=975222899 file
name=/u02/app/oracle/oradata/O_CLOUD/users01.dbf
datafile 8 switched to datafile copy
input datafile copy RECID=22 STAMP=975222899 file
name=/u02/app/oracle/oradata/O_CLOUD/pdbseed/undotbs01.dbf
datafile 116 switched to datafile copy
input datafile copy RECID=23 STAMP=975222900 file
name=/u02/app/oracle/oradata/O_CLOUD/cdata_01.dbf
datafile 141 switched to datafile copy
input datafile copy RECID=24 STAMP=975222901 file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E
6CE0532133960A63DF/datafile/o1_mf_system_fgg6bkw9_.dbf
datafile 142 switched to datafile copy
input datafile copy RECID=25 STAMP=975222901 file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E
6CE0532133960A63DF/datafile/o1_mf_sysaux_fgg6bkwb_.dbf
datafile 143 switched to datafile copy
input datafile copy RECID=26 STAMP=975222902 file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E
6CE0532133960A63DF/datafile/o1_mf_undotbs1_fgg6bkwb_.dbf
datafile 144 switched to datafile copy
input datafile copy RECID=27 STAMP=975222902 file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/tbsenc01.dbf

contents of Memory Script:
{
  set until scn  8923329;
  recover
  clone database

```

```
        delete archivelog
        ;
    }
executing Memory Script

executing command: SET until clause

Starting recover at 04-MAY-18
using channel ORA_AUX_DISK_1

starting media recovery

archived log for thread 1 with sequence 942 is already on disk
as file
/u03/app/oracle/fast_recovery_area/O_CLOUD/archivelog/2018_05_04
/o1_mf_1_942_fgr23b5f_.arc
archived log for thread 1 with sequence 943 is already on disk
as file
/u03/app/oracle/fast_recovery_area/O_CLOUD/archivelog/2018_05_04
/o1_mf_1_943_fgr23dxy_.arc
archived log file
name=/u03/app/oracle/fast_recovery_area/O_CLOUD/archivelog/2018_
05_04/o1_mf_1_942_fgr23b5f_.arc thread=1 sequence=942
archived log file
name=/u03/app/oracle/fast_recovery_area/O_CLOUD/archivelog/2018_
05_04/o1_mf_1_943_fgr23dxy_.arc thread=1 sequence=943
media recovery complete, elapsed time: 00:00:06
Finished recover at 04-MAY-18

contents of Memory Script:
{
    delete clone force archivelog all;
}
executing Memory Script

released channel: ORA_AUX_DISK_1
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=285 device type=DISK
deleted archived log
archived log file
name=/u03/app/oracle/fast_recovery_area/O_CLOUD/archivelog/2018_
05_04/o1_mf_1_942_fgr23b5f_.arc RECID=1 STAMP=975222890
deleted archived log
```

```

archived log file
name=/u03/app/oracle/fast_recovery_area/O_CLOUD/archivelog/2018_
05_04/o1_mf_1_943_fgr23dxy_.arc RECID=2 STAMP=975222893
Deleted 2 objects

Oracle instance started

Total System Global Area     1426062800 bytes

Fixed Size                  8895952 bytes
Variable Size                486539264 bytes
Database Buffers             922746880 bytes
Redo Buffers                 7880704 bytes

contents of Memory Script:
{
    sql clone "alter system set db_name =
'''O_CLOUD''' comment=
'''Reset to original value by RMAN''' scope=spfile";
    sql clone "alter system reset db_unique_name scope=spfile";
}
executing Memory Script

sql statement: alter system set db_name = '''O_CLOUD''' comment=
'''Reset to original value by RMAN''' scope=spfile

sql statement: alter system reset db_unique_name scope=spfile
Oracle instance started

Total System Global Area     1426062800 bytes

Fixed Size                  8895952 bytes
Variable Size                486539264 bytes
Database Buffers             922746880 bytes
Redo Buffers                 7880704 bytes
sql statement: CREATE CONTROLFILE REUSE SET DATABASE "O_CLOUD"
RESETLOGS ARCHIVELOG
    MAXLOGFILES      16
    MAXLOGMEMBERS    3
    MAXDATAFILES    1024
    MAXINSTANCES     8
    MAXLOGHISTORY   292
LOGFILE
```

```

        GROUP    4 ( '/u04/app/oracle redo/O_CLOUD redo01.log' ) SIZE
50 M REUSE,
        GROUP    5 ( '/u04/app/oracle redo/O_CLOUD redo02.log' ) SIZE
50 M REUSE,
        GROUP    6 ( '/u04/app/oracle redo/O_CLOUD redo03.log' ) SIZE
50 M REUSE
DATAFILE
  '/u02/app/oracle/oradata/O_CLOUD/system01.dbf',
  '/u02/app/oracle/oradata/O_CLOUD/pdbseed/system01.dbf',
'/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE0
532133960A63DF/datafile/o1_mf_system_fqg6bkw9_.dbf'
CHARACTER SET AL32UTF8

```

contents of Memory Script:

```

{
  set newname for tempfile 1 to
"/u02/app/oracle/oradata/O_CLOUD/temp01.dbf";
  set newname for tempfile 2 to
"/u02/app/oracle/oradata/O_CLOUD/pdbseed/temp012018-03-14_07-
44-30-864-AM.dbf";
  set newname for tempfile 6 to
"/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE0
532133960A63DF/datafile/o1_mf_temp_fqg6bkw_.dbf";
  set newname for tempfile 20 to
"/u02/app/oracle/oradata/O_CLOUD/temproot_01.dbf";
  switch clone tempfile all;
  catalog clone datafilecopy
"/u02/app/oracle/oradata/O_CLOUD/sysaux01.dbf",
"/u02/app/oracle/oradata/O_CLOUD/undotbs01.dbf",
"/u02/app/oracle/oradata/O_CLOUD/pdbseed/sysaux01.dbf",
"/u02/app/oracle/oradata/O_CLOUD/users01.dbf",
"/u02/app/oracle/oradata/O_CLOUD/pdbseed/undotbs01.dbf",
"/u02/app/oracle/oradata/O_CLOUD/cdata_01.dbf",
"/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE0
532133960A63DF/datafile/o1_mf_sysaux_fqg6bkw_.dbf",
"/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE0
532133960A63DF/datafile/o1_mf_undotbs1_fqg6bkw_.dbf",
"/u02/app/oracle/oradata/O_CLOUD/pdb1/tbsenc01.dbf";
  switch clone datafile all;
}
executing Memory Script

```

executing command: SET NEWNAME

```
executing command: SET NEWNAME

executing command: SET NEWNAME

executing command: SET NEWNAME

renamed tempfile 1 to /u02/app/oracle/oradata/O_CLOUD/temp01.dbf
in control file
renamed tempfile 2 to
/u02/app/oracle/oradata/O_CLOUD/pdbseed/temp012018-03-14_07-44-
30-864-AM.dbf in control file
renamed tempfile 6 to
/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E6CE05
32133960A63DF/datafile/o1_mf_temp_fqg6bkwd_.dbf in control file
renamed tempfile 20 to
/u02/app/oracle/oradata/O_CLOUD/temproot_01.dbf in control file

cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/sysaux01.dbf RECID=1
STAMP=975222997
cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/undotbs01.dbf RECID=2
STAMP=975222997
cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/pdbseed/sysaux01.dbf
RECID=3 STAMP=975222997
cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/users01.dbf RECID=4
STAMP=975222997
cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/pdbseed/undotbs01.dbf
RECID=5 STAMP=975222997
cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/cdata_01.dbf RECID=6
STAMP=975222997
cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E
6CE0532133960A63DF/datafile/o1_mf_sysaux_fgq6bkwb_.dbf RECID=7
STAMP=975222997
```

```
cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E
6CE0532133960A63DF/datafile/o1_mf_undotbs1_fgg6bkwb_.dbf RECID=8
STAMP=975222997
cataloged datafile copy
datafile copy file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/tbsenc01.dbf RECID=9
STAMP=975222998
datafile 3 switched to datafile copy
input datafile copy RECID=1 STAMP=975222997 file
name=/u02/app/oracle/oradata/O_CLOUD/sysaux01.dbf
datafile 4 switched to datafile copy
input datafile copy RECID=2 STAMP=975222997 file
name=/u02/app/oracle/oradata/O_CLOUD/undotbs01.dbf
datafile 6 switched to datafile copy
input datafile copy RECID=3 STAMP=975222997 file
name=/u02/app/oracle/oradata/O_CLOUD/pdbseed/sysaux01.dbf
datafile 7 switched to datafile copy
input datafile copy RECID=4 STAMP=975222997 file
name=/u02/app/oracle/oradata/O_CLOUD/users01.dbf
datafile 8 switched to datafile copy
input datafile copy RECID=5 STAMP=975222997 file
name=/u02/app/oracle/oradata/O_CLOUD/pdbseed/undotbs01.dbf
datafile 116 switched to datafile copy
input datafile copy RECID=6 STAMP=975222997 file
name=/u02/app/oracle/oradata/O_CLOUD/cdata_01.dbf
datafile 142 switched to datafile copy
input datafile copy RECID=7 STAMP=975222997 file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E
6CE0532133960A63DF/datafile/o1_mf_sysaux_fgg6bkwb_.dbf
datafile 143 switched to datafile copy
input datafile copy RECID=8 STAMP=975222997 file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/O_CLOUD/6B56741909D84E
6CE0532133960A63DF/datafile/o1_mf_undotbs1_fgg6bkwb_.dbf
datafile 144 switched to datafile copy
input datafile copy RECID=9 STAMP=975222998 file
name=/u02/app/oracle/oradata/O_CLOUD/pdb1/tbsenc01.dbf
decryption password set for AUXILIARY database

contents of Memory Script:
{
    Alter clone database open resetlogs;
}
executing Memory Script
```

```

database opened

contents of Memory Script:
{
    sql clone "alter pluggable database all open";
}
executing Memory Script

sql statement: alter pluggable database all open
Finished Duplicate Db at 04-MAY-18

RMAN> EXIT
$
```

5. Launch SQL\*Plus and log in to O\_CLOUD to verify that the duplicated PDB1 in O\_CLOUD holds the encrypted tablespace.

```

$ sqlplus / AS SYSDBA

SQL> SHOW PDBS

  CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
      2 PDB$SEED           READ ONLY NO
      7 PDB1                READ WRITE NO

SQL> ALTER SESSION SET CONTAINER=PDB1;

Session altered.

SQL> SELECT tablespace_name, encrypted FROM dba_tablespaces;

TABLESPACE_NAME          ENC
----- -----
SYSTEM                  NO
SYSAUX                 NO
UNDOTBS1               NO
TEMP                   NO
TBSORCL_ENC            YES

SQL>
```

6. Check if any new tablespace has been encrypted. Before completing this new operation, open the keystore in the PDB.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
password;
```

```
keystore altered.
```

```
SQL>
```

7. Create a new tablespace in the duplicated PDB1 in O\_CLOUD.

```
SQL> CREATE TABLESPACE new_tbs_enc;
CREATE TABLESPACE new_tbs_enc
*
ERROR at line 1:
ORA-28374: typed master key not found in wallet

SQL>
```

Remark: Even if the command raises the ORA-28374 error message, the tablespace is created.

8. Verify that the new tablespace is by default encrypted.

```
SQL> SELECT tablespace_name, encrypted FROM dba_tablespaces;

TABLESPACE_NAME          ENC
-----
SYSTEM                  NO
SYSAUX                 NO
UNDOTBS1                NO
TEMP                   NO
TBSORCL_ENC             YES
NEW_TBS_ENC              YES

6 rows selected.

SQL> EXIT
$
```

9. Clean up the practice by dropping the duplicated CDB, O\_CLOUD, and re-creating PDB1 in ORCL. Execute the \$HOME/labs/BD/cleanup\_DUP.sh shell script.

```
$ $HOME/labs/BD/cleanup_DUP.sh
...
$
```

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.

## **Practices for Lesson 9: Recovery and Flashback**

## Practices for Lesson 9: Overview

---

### Practices Overview

In the following practices, you will perform recovery and flashback operations on the CDB and regular and application PDBs.

- Recovery from SYSTEM PDB1 datafile loss
- Recovery from nonessential PDB1 datafile loss
- PDB point-in-time recovery
- Recovering a plugged non-CDB using preplugin backups
- Recovering a plugged PDB using preplugin backups
- Flashing back an application upgrade by using restore points
- Flashing back a PDB by using PDB snapshots
- Switching over refreshable cloned PDBs

## Practice 9-1: RMAN Recovery from SYSTEM PDB Datafile Loss

In this practice, you will perform a PDB recovery after the loss of the `SYSTEM` datafile in a regular PDB and in an application PDB.

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up the PDBs and re-create PDB1.

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

2. Execute the `$HOME/labs/RF/crash.sh` shell script. The shell script, after re-creating the `TOYS_APP` application in `TOYS_ROOT` application root, creating its application PDBs, and backing them up, crashes PDBs.

```
$ $HOME/labs/RF/crash.sh
...
$
```

Even if you see some error in removing files, continue to the next step.

3. 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 : password

ERROR:
ORA-00604: error occurred at recursive SQL level 1
ORA-01116: error in opening database file 64
ORA-01110: data file 64:
'/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B7F572B52CD7579E05
32133960A2E04/datafile/o1_mf_system_fgwtckw_.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?*

```
$ sqlplus system@robots

ERROR at line 1:
ORA-01116: error in opening database file 64
ORA-01110: data file 64:
'/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B7F572B52CD7579E05
32133960A2E04/datafile/o1_mf_system_fgwtckw_.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.**

```
$ sqlplus system@robots
Enter password : password
Connected.
SQL> SELECT * FROM toys_owner.codes;
SELECT * FROM toys_owner.codes
*
ERROR at line 1:
ORA-01116: error in opening database file 64
ORA-01110: data file 64:
'/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B7F572B52CD7579E05
32133960A2E04/datafile/o1_mf_system_fgwtckw_.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3

SQL> EXIT
$
```

```
$ sqlplus / AS SYSDBA
```

```
SQL> SHOW PDBS
```

CON_ID	CON_NAME	OPEN MODE	RESTRICTED
--------	----------	-----------	------------

```

-----  

2 PDB$SEED          READ ONLY NO  

4 PDB1              READ WRITE NO  

5 TOYS_ROOT         READ WRITE NO  

7 ROBOTS           READ WRITE NO  

3 DOLLS            READ WRITE NO  

SQL>

```

- a. In another terminal window called RMAN Session, proceed with SYSTEM datafile recovery in the toys\_root PDB.

```

$ . oraenv
ORACLE_SID = [O_CLOUD] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ rman target /

connected to target database: ORCL (DBID=1461390531)
RMAN> LIST FAILURE;

...
Failure ID Priority Status      Time Detected Summary
-----
39110    CRITICAL OPEN        06-MAY-18    System datafile 64:
'/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B7F572B52CD7579E05
32133960A2E04/datafile/o1_mf_system_fgwtckw_.dbf' is missing
39107    CRITICAL OPEN        06-MAY-18    System datafile 61:
'/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B7F51A1F45E74CBE0532133
960A7B4E/datafile/o1_mf_system_fgwkqchl_.dbf' is missing

RMAN>

```

```
RMAN> ADVISE FAILURE;
```

```

Database Role: PRIMARY

List of Database Failures
=====

Failure ID Priority Status      Time Detected Summary
-----
39110    CRITICAL OPEN        06-MAY-18    System datafile 64:
'/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B7F572B52CD7579E05
32133960A2E04/datafile/o1_mf_system_fgwtckw_.dbf' is missing
39107    CRITICAL OPEN        06-MAY-18    System datafile 61:
'/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B7F51A1F45E74CBE0532133
960A7B4E/datafile/o1_mf_system_fgwkqchl_.dbf' is missing

```

```
analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=37 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/ORCL/6B7F572B52CD7579E053
2133960A2E04/datafile/o1_mf_system_fgwtckw_.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
3. If file
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B7F51A1F45E74CBE05321339
60A7B4E/datafile/o1_mf_system_fgwkqchl_.dbf was unintentionally
renamed or moved, restore it

Automated Repair Options
=====
Option Repair Description
-----
1      Restore and recover datafile 64; Restore and recover
datafile 61
      Strategy: The repair includes complete media recovery with no
      data loss
      Repair script:
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_1874028958.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_1874028958.hm

contents of repair script:
# restore and recover datafile
```

```
sql 'TOYS_ROOT' 'alter database datafile 64 offline';
sql 'PDB1' 'alter database datafile 61 offline';
restore ( datafile 61, 64 );
recover datafile 64, 61;
sql 'TOYS_ROOT' 'alter database datafile 64 online';
sql 'PDB1' 'alter database datafile 61 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_1874028958.hm
```

contents of repair script:

```
# restore and recover datafile
sql 'TOYS_ROOT' 'alter database datafile 64 offline';
sql 'PDB1' 'alter database datafile 61 offline';
restore ( datafile 61, 64 );
recover datafile 64, 61;
sql 'TOYS_ROOT' 'alter database datafile 64 online';
sql 'PDB1' 'alter database datafile 61 online';
```

Do you really want to execute the above repair (enter YES or NO) ? **YES**

executing repair script

sql statement: alter database datafile 64 offline

```
RMAN-00571: =====
```

```
RMAN-00569: ====== ERROR MESSAGE STACK FOLLOWS =====
```

```
RMAN-00571: =====
```

```
RMAN-03002: failure of repair command at 05/06/2018 00:24:16
```

```
RMAN-03015: error occurred in stored script Repair Script
```

```
RMAN-03009: failure of sql command on default channel at
05/06/2018 00:24:16
```

```
RMAN-11003: failure during parse/execution of SQL statement:
alter database datafile 64 offline
```

```
ORA-01541: system tablespace cannot be brought offline; shut
down if necessary
```

```
RMAN>
```

*Before restoring and recovering the datafile, first close the PDBs. Because the PDBs cannot be closed normally, use the CLOSE ABORT clause.*

```
RMAN> ALTER PLUGGABLE DATABASE toys_root CLOSE ABORT;
```

Statement processed.

```
RMAN> ALTER PLUGGABLE DATABASE pdb1 CLOSE ABORT;
```

Statement processed.

```
RMAN>
```

- Now use the RMAN DRA (Data Recovery Advisor) script.

```
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_1874028958.hm
```

contents of repair script:

```
# restore and recover datafile
sql 'TOYS_ROOT' 'alter database datafile 64 offline';
sql 'PDB1' 'alter database datafile 61 offline';
restore ( datafile 61, 64 );
recover datafile 64, 61;
sql 'TOYS_ROOT' 'alter database datafile 64 online';
sql 'PDB1' 'alter database datafile 61 online';
```

Do you really want to execute the above repair (enter YES or NO)? **YES**

executing repair script

```
sql statement: alter database datafile 64 offline
```

```
sql statement: alter database datafile 61 offline
```

Starting restore at 06-MAY-18

allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=37 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 00061 to
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B7F51A1F45E74CBE05321339
60A7B4E/datafile/o1_mf_system_fgwkqchl_.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/6B7F51A1F45E74CBE0532133
960A7B4E/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T001112_fgwl
6dhl_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B7F51A1F45E74CBE
0532133960A7B4E/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T001
112_fgwl6dhl_.bkp tag=TAG20180506T001112
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:16
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 00064 to
/u02/app/oracle/oradata/ORCL/toys_root/ORCL/6B7F572B52CD7579E053
2133960A2E04/datafile/o1_mf_system_fgwtckw_.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/6B7F572B52CD7579E0532133
960A2E04/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T001112_fgwl
7jn8_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B7F572B52CD7579E
0532133960A2E04/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T001
112_fgwl7jn8_.bkp tag=TAG20180506T001112
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:07
Finished restore at 06-MAY-18

Starting recover at 06-MAY-18
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:02

Finished recover at 06-MAY-18
sql statement: alter database datafile 64 online
sql statement: alter database datafile 61 online
repair failure complete

RMAN>
```

- c. In the SQL\*Plus session, open `toys_root` and its application PDBs.

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password : password
Connected.
SQL> SHOW PDBS

CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
      5 TOYS_ROOT        MOUNTED
      7 ROBOTS           MOUNTED
      3 DOLLS            MOUNTED

SQL> STARTUP
Pluggable Database opened.
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> SHOW PDBS

CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
      5 TOYS_ROOT        READ WRITE NO
      7 ROBOTS           READ WRITE NO
      3 DOLLS            READ WRITE NO
SQL>
```

*Q/ It can happen that some other datafiles of the application PDBs and regular PDBs require recovery. Which reason could the root cause for 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 recovering them all in a script provided by the `ADVISE FAILURE ALL` command.**

**If this were the case, you would proceed with the following steps:**

- 1) In the RMAN session, proceed with other tablespaces recovery by using DRA.

```
RMAN> LIST FAILURE ALL;
Database Role: PRIMARY

List of Database Failures
=====

Failure ID Priority Status      Time Detected Summary
```

```
-----
3017      CRITICAL OPEN    25-JAN-17      System datafile 19:
'/u02/app/oracle/oradata/ORCL/toys_root/dolls/ORCL/46E75197BBE5
948E0535010ED0ABED2/datafile/o1_mf_system_d8johyhq_.dbf' needs
media recovery
3002      CRITICAL OPEN    25-JAN-17      System datafile 16:
'/u02/app/oracle/oradata/ORCL/toys_root/robots/ORCL/46E75197BBBD
5948E0535010ED0ABED2/datafile/o1_mf_system_d8joho32_.dbf' needs
media recovery
2631      HIGH      OPEN    25-JAN-17      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
-----
3017      CRITICAL OPEN    25-JAN-17      System datafile 19:
'/u02/app/oracle/oradata/ORCL/toys_root/dolls/ORCL/46E75197BBE5
948E0535010ED0ABED2/datafile/o1_mf_system_d8johyhq_.dbf' needs
media recovery
3002      CRITICAL OPEN    25-JAN-17      System datafile 16:
'/u02/app/oracle/oradata/ORCL/toys_root/robots/ORCL/46E75197BBBD
5948E0535010ED0ABED2/datafile/o1_mf_system_d8joho32_.dbf' needs
media recovery
2631      HIGH      OPEN    25-JAN-17      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/ORCL/46E75197BBE59
```

48E0535010ED0ABED2/datafile/o1\_mf\_system\_d8johyhq\_.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/ORCL/46E75197BBBD5948E0535010ED0ABED2/datafile/o1\_mf\_system\_d8joho32\_.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/ORCL/46E750A8FE57591FE0535010ED0A95E8/datafile/o1\_mf\_sysaux\_d8joh5nq\_.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/ORCL/46E750A8FE57591FE0535010ED0A95E8/datafile/o1\_mf\_undotbs1\_d8joh5nr\_.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/ORCL/46E750A8FE57591FE0535010ED0A95E8/datafile/o1\_mf\_toys\_tbs\_d8johltg\_.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/ORCL/46E75197BBBD5948E0535010ED0ABED2/datafile/o1\_mf\_sysaux\_d8joho34\_.dbf, then replace it with the correct one

8. If you restored the wrong version of data file  
/u02/app/oracle/oradata/ORCL/toys\_root/robots/ORCL/46E75197BBBD5948E0535010ED0ABED2/datafile/o1\_mf\_undotbs1\_d8joho35\_.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/ORCL/46E75197BBBE5948E0535010ED0ABED2/datafile/o1\_mf\_sysaux\_d8johyhr\_.dbf, then replace it with the correct one

10. If you restored the wrong version of data file  
/u02/app/oracle/oradata/ORCL/toys\_root/dolls/ORCL/46E75197BBBE5948E0535010ED0ABED2/datafile/o1\_mf\_undotbs1\_d8johyhs\_.dbf, then replace it with the correct one

11. If you restored the wrong version of data file  
/u02/app/oracle/oradata/ORCL/toys\_root/dolls/ORCL/46E75197BBBE5948E0535010ED0ABED2/datafile/o1\_mf\_toys\_tbs\_d8jojh5m\_.dbf, then replace it with the correct one

12. If you restored the wrong version of data file  
/u02/app/oracle/oradata/ORCL/toys\_root/robots/ORCL/46E75197BBBD5948E0535010ED0ABED2/datafile/o1\_mf\_toys\_tbs\_d8jojk8v\_.dbf, then replace it with the correct one

13. If you restored the wrong version of data file  
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/46E7B4CC58C664ECE0535010ED0ACE2C/datafile/o1\_mf\_sysaux\_d8jq3oof\_.dbf, then replace it with the correct one

14. If you restored the wrong version of data file  
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/46E7B4CC58C664ECE0535010E

```
D0ACE2C/datafile/o1_mf_undotbs1_d8jq3oog_.dbf, then replace it  
with the correct one
```

```
Automated Repair Options
```

```
=====
```

```
Option Repair Description
```

```
-----
```

```
1      Recover datafile 19; Recover datafile 16; Recover  
datafile 13; ...
```

```
Strategy: The repair includes complete media recovery with no  
data loss
```

```
Repair script:
```

```
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_2218296462.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_2218296462.hm
```

```
contents of repair script:
```

```
# recover datafile
sql 'DOLLS' 'alter database datafile 19, 20, 21, 22 offline';
sql 'ROBOTS' 'alter database datafile 16, 17, 18, 23
offline';
sql 'TOYS_ROOT' 'alter database datafile 13, 14, 15 offline';
sql 'PDB1' 'alter database datafile 25, 26 offline';
recover datafile 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 25, 26;
sql 'DOLLS' 'alter database datafile 19, 20, 21, 22 online';
sql 'ROBOTS' 'alter database datafile 16, 17, 18, 23 online';
sql 'TOYS_ROOT' 'alter database datafile 13, 14, 15 online';
sql 'PDB1' 'alter database datafile 25, 26 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_2218296462.hm  
  
contents of repair script:  
# recover datafile  
sql 'DOLLS' 'alter database datafile 19, 20, 21, 22 offline';  
sql 'ROBOTS' 'alter database datafile 16, 17, 18, 23  
offline';  
sql 'TOYS_ROOT' 'alter database datafile 13, 14, 15 offline';  
sql 'PDB1' 'alter database datafile 25, 26 offline';  
recover datafile 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,  
23, 25, 26;  
sql 'DOLLS' 'alter database datafile 19, 20, 21, 22 online';  
sql 'ROBOTS' 'alter database datafile 16, 17, 18, 23 online';  
sql 'TOYS_ROOT' 'alter database datafile 13, 14, 15 online';  
sql 'PDB1' 'alter database datafile 25, 26 online';  
  
Do you really want to execute the above repair (enter YES or  
NO) ? YES  
executing repair script  
  
sql statement: alter database datafile 19, 20, 21, 22 offline  
sql statement: alter database datafile 16, 17, 18, 23 offline  
sql statement: alter database datafile 13, 14, 15 offline  
sql statement: alter database datafile 25, 26 offline  
  
Starting recover at 25-JAN-17  
using channel ORA_DISK_1  
  
starting media recovery  
media recovery complete, elapsed time: 00:00:00  
  
Finished recover at 25-JAN-17  
  
sql statement: alter database datafile 19, 20, 21, 22 online  
sql statement: alter database datafile 16, 17, 18, 23 online  
sql statement: alter database datafile 13, 14, 15 online
```

```
sql statement: alter database datafile 25, 26 online
repair failure complete
```

RMAN>

- 2) In the SQL\*Plus session, open `toys_root`.

```
SQL> STARTUP
Pluggable Database opened.
SQL>
```

- 3) 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 ROBOTS              MOUNTED
 5 DOLLS               MOUNTED

SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE all OPEN;

Pluggable database altered.

SQL> SHOW PDBS

CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
 2 PDB$SEED            READ ONLY NO
 4 PDB1                READ WRITE NO
 3 TOYS_ROOT           READ WRITE NO
 4 ROBOTS              READ WRITE NO
 5 DOLLS               READ WRITE NO

SQL>
```

- d. Verify that the `TOYS_OWNER.CODES` shared table contains the expected rows.

```
SQL> CONNECT toys_owner@toys_root
Enter password : password
Connected.
```

```
SQL> SELECT * FROM toys_owner.codes;
```

CODE	LABEL
1	Puppet

2 Car

SQL>

- e. Verify that the TOYS\_OWNER.SALES\_DATA shared table contains the expected rows in both application PDBs.

SQL> CONNECT toys\_owner@robots

Enter password : password

Connected.

SQL> SELECT \* FROM toys\_owner.sales\_data;

YEAR	REGION	QUAR	REVENUE
2012	R_north	Q1	17238
2012	R_north	Q2	17397
2012	R_north	Q3	18617
2012	R_north	Q4	13833
2013	R_north	Q1	18829
2013	R_north	Q2	17799
2013	R_north	Q3	15794
2013	R_north	Q4	10211
2012	R_south	Q1	16495
2012	R_south	Q2	13951
2012	R_south	Q3	14776
2012	R_south	Q4	16435
2013	R_south	Q1	14916
2013	R_south	Q2	13917
2013	R_south	Q3	13560
2013	R_south	Q4	13357
2012	R_east	Q1	17278
2012	R_east	Q2	12199
2012	R_east	Q3	15189
2012	R_east	Q4	15706
2013	R_east	Q1	19411
2013	R_east	Q2	12501
2013	R_east	Q3	12213
2013	R_east	Q4	15166
2012	R_west	Q1	11679
2012	R_west	Q2	12164
2012	R_west	Q3	10428
2012	R_west	Q4	19579
2013	R_west	Q1	10941

2013	R_west	Q2	11947
2013	R_west	Q3	10102
2013	R_west	Q4	19812
32 rows selected.			
<b>SQL&gt; CONNECT toys_owner@dolls</b>			
Enter password : <i>password</i>			
Connected.			
<b>SQL&gt; SELECT * FROM toys_owner.sales_data;</b>			
YEAR	REGION	QUAR	REVENUE
-----			
2015	S_NO	Q1	640
2015	S_NO	Q2	432
2015	S_NO	Q3	437
2015	S_NO	Q4	520
2016	S_NO	Q1	436
2016	S_NO	Q2	497
2016	S_NO	Q3	773
2016	S_NO	Q4	462
2015	S_SO	Q1	471
2015	S_SO	Q2	610
2015	S_SO	Q3	569
2015	S_SO	Q4	747
2016	S_SO	Q1	440
2016	S_SO	Q2	624
2016	S_SO	Q3	483
2016	S_SO	Q4	539
2015	S_EA	Q1	734
2015	S_EA	Q2	486
2015	S_EA	Q3	443
2015	S_EA	Q4	464
2016	S_EA	Q1	494
2016	S_EA	Q2	539
2016	S_EA	Q3	777
2016	S_EA	Q4	777
2015	S_WE	Q1	793
2015	S_WE	Q2	648
2015	S_WE	Q3	560
2015	S_WE	Q4	607
2016	S_WE	Q1	660

```
2016 S_WE          Q2        485
2016 S_WE          Q3        497
2016 S_WE          Q4       700

32 rows selected.

SQL> EXIT
$
```

4. In the RMAN session, back up the whole CDB.

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG delete all input;
...
RMAN> DELETE OBSOLETE ;
...
RMAN>
```

Keep the RMAN session open for further backup and recovery operations.

## Practice 9-2: RMAN Recovery from Nonessential PDB Datafile Loss

### Overview

In this practice, you will recover from a nonessential PDB datafile.

### Tasks

1. Execute the `$HOME/labs/RF/ldatatbs_PDB1.sh` shell script to create a tablespace `LDATA` in PDB1, a table `SYSTEM.BIGTAB` stored in `LDATA` tablespace.

```
$ $HOME/labs/RF/ldatatbs_PDB1.sh
...
$
```

2. Remove a datafile of the `LDATA` tablespace of PDB1.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ sqlplus system@PDB1

Enter password : password

SQL> SELECT file_name FROM dba_data_files
      WHERE tablespace_name='LDATA';
2
FILE_NAME
-----
/u02/app/oracle/oradata/ORCL/pdb1/ldata01.dbf

SQL> !rm /u02/app/oracle/oradata/ORCL/pdb1/ldata01.dbf

SQL>
```

3. Proceed with the traditional procedure to restore the missing datafile and recover the tablespace as if it were a non-CDB.
  - a. Put the tablespace in OFFLINE mode if the PDB has not been automatically closed.

```
SQL> ALTER TABLESPACE ldata OFFLINE IMMEDIATE;
Tablespace altered.

SQL>
```

- b. In the RMAN session, restore and recover the tablespace.

```
RMAN> RESTORE TABLESPACE PDB1:ldata;

Starting restore at 06-MAY-18
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00073 to
/u02/app/oracle/oradata/ORCL/pdb1/ldata01.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/6B7F51A1F45E74CBE0532133
960A7B4E/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T012811_fg
pkfm8_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B7F51A1F45E74CBE
0532133960A7B4E/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T012
811_fgwpkfm8_.bkp tag=TAG20180506T012811
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
Finished restore at 06-MAY-18

RMAN> RECOVER TABLESPACE PDB1:ldata;

Starting recover at 06-MAY-18
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:01

Finished recover at 06-MAY-18

RMAN>
```

4. In the SQL\*Plus session, put the tablespace back ONLINE.

```
SQL> ALTER TABLESPACE ldata ONLINE;

Tablespace altered.

SQL> SELECT distinct label FROM system.bigtab;

LABEL
-----

```

```
DATA FROM system.bigtab ON TABLESPACE ldata  
SQL>
```

5. After a recovery, back up the whole PDB terminal window (2).

```
RMAN> BACKUP PLUGGABLE DATABASE pdb1;  
...  
RMAN>
```

## Practice 9-3: PDB PITR

---

### Overview

In this practice, you will perform a PDB point-in-time recovery. A table `SYSTEM.BIGTAB` in `PDB1` contains rows. A loading operation is executed in `PDB1`, and you realize later that rows were loaded into the wrong PDB. Rows should have been loaded into `SYSTEM.BIGTAB` in `PDB18` and not into `SYSTEM.BIGTAB` in `PDB1`. You have to restore the situation to the time before the rows were inserted inappropriately.

### Tasks

1. In the SQL\*Plus session, connect to `PDB1` and load new data (inappropriately) into `SYSTEM.BIGTAB` table after truncating the table by using the code written below. Before loading data, note the SCN value.

```
SQL> CONNECT system@PDB1
Enter password : password
Connected.
SQL> TRUNCATE TABLE system.bigtab;

Table truncated.

SQL> SELECT timestamp_to_scn(sysdate) FROM v$database;

TIMESTAMP_TO_SCN(SYSDATE)
-----
9688136

SQL>
```

```
SQL> BEGIN
  FOR i in 1.. 10000 LOOP
    insert into system.bigtab values
('aaaaaaaaaaaaaaaaaaaaaaaa');
  END LOOP;
  COMMIT;
END;
/
PL/SQL procedure successfully completed.

SQL>
```

```
SQL> SELECT timestamp_to_scn(sysdate) FROM v$database;

TIMESTAMP_TO_SCN (SYSDATE)
-----
9688150

SQL>
```

2. You realize that you loaded the data in the wrong PDB. You create a tablespace in the right PDB PDB18 to store table SYSTEM.BIGTAB before recovering PDB1 to the time when the table was still empty.

```
SQL> CONNECT system@PDB18
Enter password : password
Connected.
SQL>
```

```
SQL> DROP TABLESPACE ldata INCLUDING CONTENTS AND DATAFILES;
drop tablespace ldata INCLUDING CONTENTS AND DATAFILES
*
ERROR at line 1:
ORA-00959: tablespace 'LDATA' does not exist

SQL> CREATE TABLESPACE ldata DATAFILE
      '/u02/app/oracle/oradata/CDB18/PDB18/ldata01.dbf'
      SIZE 10m;
2      3
Tablespace created.

SQL> CREATE TABLE system.bigtab (label VARCHAR2(50))
      TABLESPACE ldata;
2
Table created.

SQL>
```

3. Load rows into SYSTEM.BIGTAB table in PDB18.

```
SQL> BEGIN
  FOR i in 1.. 10000 LOOP
    insert into system.bigtab values
('aaaaaaaaaaaaaaaaaaaaaaaaaaa');
  END LOOP;
  COMMIT;
END;
/
```

2	3	4	5	6	7
---	---	---	---	---	---

PL/SQL procedure successfully completed.

SQL>

4. Proceed to the PITR (point-in-time recovery) of PDB1 to the time when the table SYSTEM.BIGTAB was still empty.
  - a. Connect to ORCL and close PDB1.

```
SQL> CONNECT sys@PDB1 AS SYSDBA
Enter password: password
Connected.
SQL> ALTER PLUGGABLE DATABASE pdb1 CLOSE;

Pluggable database altered.

SQL>
```

- b. In the RMAN session, perform the PDB PITR of PDB1. Pay attention to **replace the SCN value noted in step 1.**

```
RMAN> RUN {
      SET UNTIL SCN = 9688136;
      RESTORE PLUGGABLE DATABASE pdb1;
      RECOVER PLUGGABLE DATABASE pdb1 AUXILIARY
          DESTINATION='/u02/app/oracle/oradata';
      ALTER PLUGGABLE DATABASE pdb1 OPEN RESETLOGS;
    }
2> 3> 4> 5> 6> 7>
executing command: SET until clause

Starting restore at 06-MAY-18
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00061 to
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B7F51A1F45E74CBE05321339
60A7B4E/datafile/o1_mf_system_fgwkqchl_.dbf
channel ORA_DISK_1: restoring datafile 00062 to
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B7F51A1F45E74CBE05321339
60A7B4E/datafile/o1_mf_sysaux_fgwkqchp_.dbf
channel ORA_DISK_1: restoring datafile 00063 to
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B7F51A1F45E74CBE05321339
60A7B4E/datafile/o1_mf_undotbs1_fgwkqchq_.dbf
```

```

channel ORA_DISK_1: restoring datafile 00073 to
/u02/app/oracle/oradata/ORCL/pdb1/1data01.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/6B7F51A1F45E74CBE0532133
960A7B4E/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T015900_fgw
rh441_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B7F51A1F45E74CBE
0532133960A7B4E/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T015
900_fgwrh441_.bkp tag=TAG20180506T015900
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:25
Finished restore at 06-MAY-18

Starting recover at 06-MAY-18
current log archived
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:02

Finished recover at 06-MAY-18

Statement processed

RMAN>

```

5. In the SQL\*Plus session, check that only PDB1 was restored to SCN 4261442, **SCN noted in step 1** and not PDB18.

```

SQL> CONNECT system@PDB1
Enter password : password
Connected.
SQL> SELECT * FROM system.bigtab;

no rows selected

SQL> CONNECT system@PDB18
Enter password : password
Connected.
SQL> SELECT COUNT(*) FROM system.bigtab;

COUNT(*)
-----

```

```
10000  
SQL>
```

You undid the load into system.bigtab in PDB1.

6. In the RMAN session, back up the CDB.

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG DELETE ALL INPUT;  
...  
RMAN> EXIT  
$
```

## Practice 9-4: Recovering a Plugged Non-CDB by Using Preplugin Backups

---

### Overview

In this practice, you will recover a plugged non-CDB by using preplugin backups. Preplugin backups are all backups of a non-CDB taken before unplugging/plugging operations, usable after plugging into a new CDB.

### Tasks

1. You will use the Oracle By Example “Recovering Plugged non-CDBs Using Preplugin Backups” to see how to proceed. On VM2, launch a browser and click the bookmark from the Bookmarks Toolbar of the browser. The URL is the following:  
[file:///home/oracle/labs/OBE/recovering\\_plugged\\_noncdb\\_using\\_preplugin\\_backups/recovering\\_plugged\\_noncdb\\_using\\_preplugin\\_backups.html](file:///home/oracle/labs/OBE/recovering_plugged_noncdb_using_preplugin_backups/recovering_plugged_noncdb_using_preplugin_backups.html)

## Practice 9-5: Recovering a Plugged PDB by Using Preplugin Backups

### Overview

In this practice, you will recover a plugged PDB by using preplugin backups.

### Tasks

- Before starting the practice, execute the `$HOME/labs/RF/glogin_9.sh` shell script. It sets formatting for all columns selected in queries.

```
$ $HOME/labs/RF/glogin_9.sh
...
$
```

- Re-create PDB1 and the HR schema into PDB1 by executing the `$HOME/labs/RF/HR.sh` shell script. The shell script also backs up PDB1.

```
$ $HOME/labs/RF/HR.sh
...
$
```

- Prepare the PDB before unplugging to use preplugin backups.

- In your current SQL\*Plus session, display the content of the `HR.DEPARTMENTS` table in PDB1.

```
$ sqlplus system@PDB1
Enter password: password

SQL> SELECT * FROM hr.departments;

DEPARTMENT_ID DEPARTMENT_NAME          MANAGER_ID LOCATION_ID
-----          -----
      10 Administration                 200        1700
      20 Marketing                      201        1800
      30 Purchasing                     114        1700
      40 Human Resources                203        2400
      50 Shipping                       121        1500
      60 IT                            103        1400
      70 Public Relations               204        2700
      80 Sales                          145        2500
      90 Executive                      100        1700
     100 Finance                       108        1700
     110 Accounting                     205        1700
     120 Treasury                      1700
     130 Corporate Tax                  1700
     140 Control And Credit             1700
     150 Shareholder Services           1700
     160 Benefits                       1700
```

170	Manufacturing	1700
180	Construction	1700
190	Contracting	1700
200	Operations	1700
210	IT Support	1700
220	NOC	1700
230	IT Helpdesk	1700
240	Government Sales	1700
250	Retail Sales	1700
260	Recruiting	1700
270	Payroll	1700

27 rows selected.

SQL>

- b. In the RMAN session, back up PDB1 before unplugging. It is recommended to back up all the archive logs before the unplug operation so that they can be restored from preplugin backups when a recovery might be required in the destination CDB.

```
$ rman target /
RMAN> BACKUP PLUGGABLE DATABASE pdb1 PLUS ARCHIVELOG;

Starting backup at 06-MAY-18
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=1015 RECID=1014
STAMP=975380179
input archived log thread=1 sequence=1016 RECID=1015
STAMP=975380299
channel ORA_DISK_1: starting piece 1 at 06-MAY-18
channel ORA_DISK_1: finished piece 1 at 06-MAY-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05
_06/o1_mf_annnn_TAG20180506T025819_fgwtclz_.bkp
tag=TAG20180506T025819 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 06-MAY-18

Starting backup at 06-MAY-18
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
```

```
input datafile file number=00096
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B81A3B3D0643F26E053
2133960A99D9/datafile/o1_mf_sysaux_fgwvgy1n_.dbf
input datafile file number=00095
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B81A3B3D0643F26E053
2133960A99D9/datafile/o1_mf_system_fgwvgy1m_.dbf
input datafile file number=00097
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6B81A3B3D0643F26E053
2133960A99D9/datafile/o1_mf_undotbs1_fgwvgy1o_.dbf
channel ORA_DISK_1: starting piece 1 at 06-MAY-18
channel ORA_DISK_1: finished piece 1 at 06-MAY-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B81A3B3D0643F26E
0532133960A99D9/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T025
821_fgwtgf0_.bkp tag=TAG20180506T025821 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 06-MAY-18

Starting backup at 06-MAY-18
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=1017 RECID=1016
STAMP=975380319
channel ORA_DISK_1: starting piece 1 at 06-MAY-18
channel ORA_DISK_1: finished piece 1 at 06-MAY-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05_
06/o1_mf_annnn_TAG20180506T025839_fgwvv019_.bkp
tag=TAG20180506T025839 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:02
Finished backup at 06-MAY-18

Starting Control File and SPFILE Autobackup at 06-MAY-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
5_06/o1_mf_s_975380321_fgwvv2r7_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 06-MAY-18

RMAN> EXIT
$
```

- c. Back to the SQL\*Plus session, export RMAN backup information that belongs to PDB1 to its dictionary before unplugging so that preplugin backups can be used in the target CDB even if no new backup will be performed after plugging. The metadata is transported along with the PDB during the migration.

```
SQL> exec DBMS_PDB.EXPORTRMANBACKUP('PDB1')

PL/SQL procedure successfully completed.

SQL>
```

4. Unplug and plug the PDB.

- a. Unplug PDB1 from ORCL.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER PLUGGABLE DATABASE pdb1 CLOSE;

Pluggable database altered.

SQL> HOST rm /tmp/pdb1.xml
rm: Cannot remove '/tmp/pdb1.xml': No such file or directory

SQL> ALTER PLUGGABLE DATABASE pdb1
      UNPLUG INTO '/tmp/pdb1.xml';
2
Pluggable database altered.

SQL>
```

- b. Archive the current redo log and quit the session.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL> SELECT name, next_change#, sequence#
      FROM v$archived_log ORDER BY 3;
2
NAME
-----
-----
NEXT_CHANGE#  SEQUENCE#
-----
...
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_06/o1
_mf_1_1016_fgwtc10_.arc
9698171        1016
```

```

/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_06/o1
_mf_1_1017_fgwtzhc_.arc
 9698198      1017

/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_06/o1
_mf_1_1018_fgwvzrqt_.arc
 9699159      1018

800 rows selected.

SQL> EXIT
$
```

- c. Plug PDB1 as PDB1\_IN\_CDB18 into CDB18 and open PDB1\_IN\_CDB18.

```

$ mkdir /u02/app/oracle/oradata/CDB18/pdb1_in_CDB18
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base for
ORACLE_HOME=/u01/app/oracle/product/18.1.0/dbhome_1 is
/u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> CREATE PLUGGABLE DATABASE pdb1_in_cdb18 AS CLONE
      USING '/tmp/pdb1.xml'
      CREATE_FILE_DEST =
        '/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18';
2      3
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb1_in_cdb18 OPEN;

Pluggable database altered.

SQL>
```

- d. In the RMAN session, check whether the preplugin backups for PDB1\_IN\_CDB18 are cataloged in CDB18.

```

$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ rman target /
```

```
connected to target database: CDB18 (DBID=1936885433)

RMAN> SET PREPLUGIN CONTAINER= pdb1_in_cdb18;

executing command: SET PREPLUGIN CONTAINER
using target database control file instead of recovery catalog
RMAN>
```

```
RMAN> LIST PREPLUGIN BACKUP;

List of Backup Sets
=====
BS Key  Size      Device Type Elapsed Time Completion Time
-----  -----  -----
149      5.65M    DISK        00:00:01  05-MAY-18
          BP Key: 149   Status: AVAILABLE Compressed: NO   Tag:
TAG20180505T231814
          Piece Name:
/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05_05/o1_
mf_annnn_TAG20180505T231814_fgwxq64_.bkp

List of Archived Logs in backup set 149
Thrd Seq  Low SCN     Low Time   Next SCN   Next Time
-----  -----
1       995    9544678   05-MAY-18  9546048   05-MAY-18
...
          Piece Name:
/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05_06/o1_
mf_annnn_TAG20180506T025839_fgwvv019_.bkp

List of Archived Logs in backup set 238
Thrd Seq  Low SCN     Low Time   Next SCN   Next Time
-----  -----
1       1017   9698171   06-MAY-18  9698198   06-MAY-18
RMAN>
```

- e. Check whether the preplugin archive log files for ORCL\_PDB1 are cataloged in ORCL.

```
RMAN> LIST PREPLUGIN ARCHIVELOG ALL;

List of Archived Log Copies for database with db_unique_name
CDB18
```

```
=====
Key      Thrd Seq      S Low Time
-----  -----
1014      1    1015      A 06-MAY-18
      Name:
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_06/o1
_mf_1_1015_fgwpmbc_.arc

1015      1    1016      A 06-MAY-18
      Name:
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_06/o1
_mf_1_1016_fgwtc10_.arc

1016      1    1017      A 06-MAY-18
      Name:
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_06/o1
_mf_1_1017_fgwtzhc_.arc

RMAN>
```

- f. Verify that the cataloged preplugin backups and archive log files are available on disk.

```
RMAN> CROSSCHECK PREPLUGIN BACKUP;

using channel ORA_DISK_1
crosschecked backup piece: found to be 'AVAILABLE'
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05
_05/o1_mf_annnn_TAG20180505T231814_fgwgxq64_.bkp RECID=149
STAMP=975367095
...
crosschecked backup piece: found to be 'AVAILABLE'
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B81A3B3D0643F26E
0532133960A99D9/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T025
821_fgwtgf0_.bkp RECID=237 STAMP=975380302
crosschecked backup piece: found to be 'AVAILABLE'
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_05
_06/o1_mf_annnn_TAG20180506T025839_fgwvv019_.bkp RECID=238
STAMP=975380319
Crosschecked 22 objects

RMAN>
```

All backups are available.

```
RMAN> CROSSCHECK PREPLUGIN ARCHIVELOG ALL;

released channel: ORA_DISK_1
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=36 device type=DISK
validation succeeded for archived log
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_
06/o1_mf_1_1015_fgwpmbc_.arc RECID=1014 STAMP=975380179
validation succeeded for archived log
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_
06/o1_mf_1_1016_fgwtcl0_.arc RECID=1015 STAMP=975380299
validation succeeded for archived log
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_
06/o1_mf_1_1017_fgwtzhc_.arc RECID=1016 STAMP=975380319
Crosschecked 3 objects

RMAN>
```

All archive log files are available.

5. In the SQL\*Plus session, a datafile is unintentionally removed from PDB1\_IN\_CDB18.
  - a. Remove the system datafile of PDB1\_IN\_CDB18.

```
SQL> CONNECT sys@PDB1_In_CDB18 AS SYSDBA
Enter password: password
Connected.
SQL> SELECT name FROM v$logfile;

NAME
-----
-----
/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_system_fgwwb44r_.dbf

/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_sysaux_fgwwb46g_.dbf

/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_undotbs1_fgwwb46j_.dbf
```

```
SQL> HOST rm
/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_system_fgwwb44r_.dbf

SQL>
```

- b. Try to connect to the PDB. You may be able to connect but then get an error or maybe not. In the latter case, you would get the “You are no longer connected to ORACLE.” error message.

```
SQL> CONNECT system@PDB1_IN_CDB18
Enter password: password
ERROR:
ORA-00604: error occurred at recursive SQL level 1
ORA-01116: error in opening database file 84
ORA-01110: data file 84:
'/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B4
561E0532133960AA019/datafile/o1_mf_system_fgwwb44r_.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3

Error accessing package DBMS_APPLICATION_INFO

Connected.
SQL>
```

- c. Close PDB1\_IN\_CDB18 if not already closed.

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

CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
2 PDB$SEED              READ ONLY NO
3 PDB18                 READ WRITE NO
4 PDB1_IN_CDB18          READ WRITE NO
SQL> ALTER PLUGGABLE DATABASE pdb1_in_cdb18 CLOSE;

Pluggable database altered.

SQL> SHOW PDBS

CON_ID CON_NAME          OPEN MODE RESTRICTED
```

```

-----  

2 PDB$SEED           READ ONLY NO  

3 PDB18              READ WRITE NO  

4 PDB1_IN_CDB18      MOUNTED  

SQL>

```

6. Recover the PDB from preplugin backups.

- a. To recover the situation, because there is no new backup completed after the plug-in operation, you will use the preplugin backups. The datafiles are restored from backups taken before the PDB was plugged in.

In the RMAN session, restore and recover `PDB1_IN_CDB18` from the preplugin backups. Run a normal recovery after preplugin recovery in order to synchronize the PDB with the CDB. If the channel used is different, ignore this potential discrepancy.

```

RMAN> RUN
  { RESTORE PLUGGABLE DATABASE pdb1_in_cdb18 FROM PREPLUGIN;
    RECOVER PLUGGABLE DATABASE pdb1_in_cdb18 FROM PREPLUGIN;
  }
2> 3> 4>

Starting restore at 06-MAY-18
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=36 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 00084 to
/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_system_fgwwb44r_.dbf
channel ORA_DISK_1: restoring datafile 00085 to
/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_sysaux_fgwwb46g_.dbf
channel ORA_DISK_1: restoring datafile 00086 to
/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_undotbs1_fgwwb46j_.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/6B81A3B3D0643F26E0532133
960A99D9/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T025821_fgw
vtgf0_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B81A3B3D0643F26E
0532133960A99D9/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T025
821_fgwtgf0_.bkp tag=TAG20180506T025821
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15
Finished restore at 06-MAY-18

```

```

Starting recover at 06-MAY-18
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 1017 is already on disk
as file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05/o1
_mf_1_1017_fgwtzhc_.arc
unable to find archived log
archived log thread=1 sequence=1018
RMAN-00571: =====
RMAN-00569: ======ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-03002: failure of recover command at 05/06/2018 03:47:12
RMAN-06054: media recovery requesting unknown archived log for
thread 1 with sequence 1018 and starting SCN of 9698198

RMAN>

```

*Q/ Has the last redo log file archived been cataloged?*

**A/ No, because it has been created after the PDB was unplugged.**

- b. Catalog the last archived log file missing, generated after the unplug operation.

```

RMAN> CATALOG PREPLUGIN ARCHIVELOG
'/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05/o1
_mf_1_1018_fgwvzrqt_.arc';

cataloged archived log
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_
06/o1_mf_1_1018_fgwvzrqt_.arc RECID=1038 STAMP=0

RMAN>

```

- c. Recover the PDB.

```

RMAN> RUN
{
  RESTORE PLUGGABLE DATABASE pdb1_in_cdb18 FROM PREPLUGIN;
  RECOVER PLUGGABLE DATABASE pdb1_in_cdb18 FROM PREPLUGIN;
}
2> 3> 4>
Starting restore at 06-MAY-18
using channel ORA_DISK_1

```

```

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00084 to
/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_system_fgwwb44r_.dbf
channel ORA_DISK_1: restoring datafile 00085 to
/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_sysaux_fgwwb46g_.dbf
channel ORA_DISK_1: restoring datafile 00086 to
/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B45
61E0532133960AA019/datafile/o1_mf_undotbs1_fgwwb46j_.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/6B81A3B3D0643F26E0532133
960A99D9/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T025821_fgw
vtgf0_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6B81A3B3D0643F26E
0532133960A99D9/backupset/2018_05_06/o1_mf_nnndf_TAG20180506T025
821_fgwtgf0_.bkp tag=TAG20180506T025821
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15
Finished restore at 06-MAY-18

Starting recover at 06-MAY-18
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 1017 is already on disk
as file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_06/o1
_mf_1_1017_fgwtzhc_.arc
archived log for thread 1 with sequence 1018 is already on disk
as file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_05_06/o1
_mf_1_1018_fgwvzrqj_.arc
media recovery complete, elapsed time: 00:00:02
Finished recover at 06-MAY-18

RMAN>

```

- d. In SQL\*Plus session, open PDB1\_IN\_CDB18.

```

SQL> ALTER PLUGGABLE DATABASE pdb1_in_cdb18 OPEN;
ALTER PLUGGABLE DATABASE pdb1_in_cdb18 OPEN
*
```

```

ERROR at line 1:
ORA-01122: database file 86 failed verification check
ORA-01110: data file 86:
'/u02/app/oracle/oradata/CDB18/pdb1_in_CDB18/CDB18/6B81D7CB0F6B4
561E
0532133960AA019/datafile/o1_mf_undotbs1_fgwwb46j_.dbf'
ORA-01204: file number is 97 rather than 86 - wrong file

SQL>

```

- e. Because the PDB is in LOCAL UNDO mode, you have to recover the UNDO tablespace. In the RMAN session, perform the recovery operation for the PDB.

```

RMAN> RECOVER PLUGGABLE DATABASE pdb1_in_cdb18;

Starting recover at 06-MAY-18
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:04

Finished recover at 06-MAY-18

RMAN> EXIT
$
```

- f. In the SQL\*Plus session, open PDB1\_IN\_CDB18.

```

SQL> ALTER PLUGGABLE DATABASE pdb1_in_cdb18 OPEN;

Pluggable database altered.

SQL>
```

- g. Verify the content of the HR.DEPARTMENTS table in PDB1\_IN\_CDB18.

```

SQL> CONNECT hr@PDB1_IN_CDB18
Enter password: password
Connected.
SQL> SELECT * FROM hr.departments;

DEPARTMENT_ID DEPARTMENT_NAME          MANAGER_ID LOCATION_ID
-----          -----
      10 Administration                200        1700
      20 Marketing                   201        1800
      30 Purchasing                  114        1700
      40 Human Resources             203        2400
      50 Shipping                     121        1500
```

```
60 IT 103 1400
70 Public Relations 204 2700
80 Sales 145 2500
90 Executive 100 1700
100 Finance 108 1700
110 Accounting 205 1700
120 Treasury 1700
130 Corporate Tax 1700
140 Control And Credit 1700
150 Shareholder Services 1700
160 Benefits 1700
170 Manufacturing 1700
180 Construction 1700
190 Contracting 1700
200 Operations 1700
210 IT Support 1700
220 NOC 1700
230 IT Helpdesk 1700
240 Government Sales 1700
250 Retail Sales 1700
260 Recruiting 1700
270 Payroll 1700

27 rows selected.

SQL> EXIT
$
```

7. Execute the \$HOME/labs/RF/cleanup\_preplugin\_PDBs.sh shell script to clean up preplugin backups and drop the PDBs.

```
$ $HOME/labs/RF/cleanup_preplugin_PDBs.sh
...
$
```

## Practice 9-6: Flashing Back an Application Upgrade by Using Restore Points

### 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 restore points to flashback the `toys_root` application container to the time before the application upgrade was committed.

### Tasks

1. Set `ORCL` in FLASHBACK mode.

```
$ . oraenv
ORACLE_SID = [CDB18] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> SELECT flashback_on FROM v$database;

FLASHBACK_ON
-----
NO

SQL> ALTER SYSTEM SET
      DB_FLASHBACK_RETENTION_TARGET=2880 SCOPE=BOTH;
      2
System altered.

SQL> ALTER DATABASE FLASHBACK ON;

Database altered.

SQL> SELECT flashback_on FROM v$database;

FLASHBACK_ON
-----
YES

SQL> EXIT
$
```

2. Before starting the practice, execute the `$HOME/labs/RF/setup_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/RF/setup_toys_app.sh
...
$
```

3. You plan to apply an application upgrade by executing the `@$HOME/labs/RF/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.
- Create a restore point for each of the PDBs of the application container 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
  4 TOYS_ROOT          READ WRITE NO
  5 ROBOTS             READ WRITE NO
  6 DOLLS              READ WRITE NO

SQL> CREATE RESTORE POINT start_upgrade_toys_root
      FOR PLUGGABLE DATABASE toys_root;
2
Restore point created.

SQL> CREATE RESTORE POINT start_upgrade_robots
      FOR PLUGGABLE DATABASE robots;
2
Restore point created.
```

```
SQL> CREATE RESTORE POINT start_upgrade_dolls
      FOR PLUGGABLE DATABASE dolls;
2
Restore point created.

SQL>
```

*Q/ How can you check that the restore points are 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_TOYS_ROOT YES NO        4
START_UPGRADE_ROBOTS    YES NO        5
START_UPGRADE_DOLLS    YES NO        6

SQL>
```

- b. Apply an application upgrade. The script synchronizes the application PDBs.

```
SQL> @$HOME/labs/RF/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 : password
Connected.
SQL> SELECT * FROM toys_owner.categories;

C1 CATEGORY
-----
1 GAMES
2 PUPPETS
3 VEHICLES

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

```

SQL> SELECT * FROM toys_owner.categories;

          C1 CATEGORY
-----
1 GAMES
2 PUPPETS
3 VEHICLES

SQL> CONNECT toys_owner@dolls
Enter password : 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.
- Close `toys_root`.

```

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

Pluggable database altered.

SQL>

```

- Flashback `toys_root` and application PDBs, `robots` and `dolls`, to the restore point.

```

SQL> FLASHBACK PLUGGABLE DATABASE toys_root
      TO RESTORE POINT start_upgrade_toys_root;
2
Flashback complete.

SQL> FLASHBACK PLUGGABLE DATABASE robots
      TO RESTORE POINT start_upgrade_robots;
2
Flashback complete.

```

```
SQL> FLASHBACK PLUGGABLE DATABASE dolls
      TO RESTORE POINT start_upgrade_dolls;
2
Flashback complete.

SQL>
```

*Q/ Would the flashback operation work if no restore point was created for each of the application PDBs?*

**A/ No, the flashback operation would not have completed. The error would have been the following one:**

```
SQL> FLASHBACK PLUGGABLE DATABASE robots TO RESTORE POINT
start_upgrade
*
ERROR at line 1:
ORA-38780: Restore point 'START_UPGRADE' does not exist.
```

5. Open the application root and the application PDBs.

- a. Open the application root.

```
SQL> ALTER PLUGGABLE DATABASE toys_root OPEN RESETLOGS;
ALTER PLUGGABLE DATABASE toys_root OPEN RESETLOGS
*
ERROR at line 1:
ORA-39862: RESETLOGS option only valid after a Pluggable Database incomplete recovery

SQL> ALTER PLUGGABLE DATABASE toys_root CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE toys_root OPEN;

Pluggable database altered.

SQL>
```

- b. Open the application PDBs:

```
SQL> CONNECT sys@toys_root AS SYSDBA
Enter password: password
Connected.

SQL> ALTER PLUGGABLE DATABASE robots OPEN RESETLOGS;
```

```
Pluggable database altered.
```

```
SQL> ALTER PLUGGABLE DATABASE dolls OPEN RESETLOGS;
```

```
Pluggable database altered.
```

```
SQL>
```

- c. Check that the table has been dropped as it was before the application upgrade.

```
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 : 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 : 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 not be forgotten to avoid performance degradation?*

**A/ Drop the restore points that increase the flashback log volume.**

```
SQL> CONNECT / AS SYSDBA
```

```
Connected.
```

```
SQL> DROP RESTORE POINT start_upgrade_toys_root  
      FOR PLUGGABLE DATABASE toys_root;  
2  
Restore point dropped.  
  
SQL> DROP RESTORE POINT start_upgrade_robots  
      FOR PLUGGABLE DATABASE robots;  
2  
Restore point dropped.  
  
SQL> DROP RESTORE POINT start_upgrade_dolls  
      FOR PLUGGABLE DATABASE dolls;  
2  
Restore point dropped.  
  
SQL> EXIT  
$
```

6. After the flashback operation, back up the CDB. In the RMAN session, complete the backup operation.

```
$ rman target /  
  
RMAN> BACKUP DATABASE PLUS ARCHIVELOG delete all input;  
...  
RMAN> DELETE OBSOLETE;  
...  
RMAN> EXIT  
$
```

7. Execute the \$HOME/labs/admin/cleanup\_PDBs.sh shell script to drop the PDBs created in ORCL.

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

## Practice 9-7: Managing and Using PDB Snapshots

---

### Overview

In this practice, you will enable PDBs for creating PDB snapshots and use the PDB snapshots created manually or automatically to create new PDBs or flashback a PDB to a point in time.

### Tasks

1. You will use the Oracle By Example “Managing and Using PDB Snapshots in a Carousel” to see how to proceed. On VM2, launch a browser and click the bookmark from the Bookmarks Toolbar of the browser. The URL is the following:

[file:///home/oracle/labs/OBE/managing\\_and\\_using\\_pdb\\_snapshots\\_carousel/managing\\_and\\_using\\_pdb\\_snapshots\\_carousel.html](file:///home/oracle/labs/OBE/managing_and_using_pdb_snapshots_carousel/managing_and_using_pdb_snapshots_carousel.html)

## Practice 9-8: Switching Over Refreshable Cloned PDBs

---

### Overview

In this practice, you will reverse the roles of a PDB and its refreshable cloned PDB. The refreshable cloned PDB can be made the primary PDB while the primary PDB would become the refreshable cloned PDB.

### Tasks

1. You will use the Oracle By Example “Switching Over a Refreshable Clone PDB” to see how to proceed. On VM2, launch a browser and click the bookmark from the Bookmarks Toolbar of the browser. The URL is the following:

[file:///home/oracle/labs/OBE/switching\\_over\\_refreshable\\_pdb/switching\\_over\\_refreshable\\_pdb.html](file:///home/oracle/labs/OBE/switching_over_refreshable_pdb/switching_over_refreshable_pdb.html)

## **Practices for Lesson 10: Performance**

## Practices for Lesson 10: Overview

---

### Practices Overview

In these practices, you monitor performance at CDB and PDB levels, run ADDM and get recommendations at CDB and PDB levels, and finally monitor and tune SQL execution based on application-shared objects in application PDBs.

## Practice 10-1: 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. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. Then execute the `$HOME/labs/perf/glogin_10.sh` shell script. It appends `COL` commands to format all columns selected in queries.

```
$ $HOME/labs/perf/glogin_10.sh
$
```

3. Then use the `$HOME/labs/perf/setup_tuning.sh` shell script to create shared application tables in the `hr_root` application root for the `hr_app` application. The script takes at least 15 minutes to load data in data-linked and metadata-linked tables.

```
$ $HOME/labs/perf/setup_tuning.sh
...
$
```

4. Then you start an application workload in `sales` and `research`.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ cd $HOME/labs/perf
$ $HOME/labs/perf/start_workload.sh 1 sales
$ $HOME/labs/perf/start_workload.sh 1 research
$
```

Until you remove the `$HOME/labs/perf/runload` file, the workload continues.

5. In another terminal window session, 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
$ $HOME/labs/perf/glogin_10.sh
```

```
$ sqlplus / AS SYSDBA

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 is listening to the localhost (*yourserver*) by using TCP protocol, the port 5500 for ORCL, the http presentation with RAW session data.

```
$ lsnrctl status
...
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=your_server) (PORT=5500)) (Security=(my_wallet_directory=/u01/app/oracle/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**.

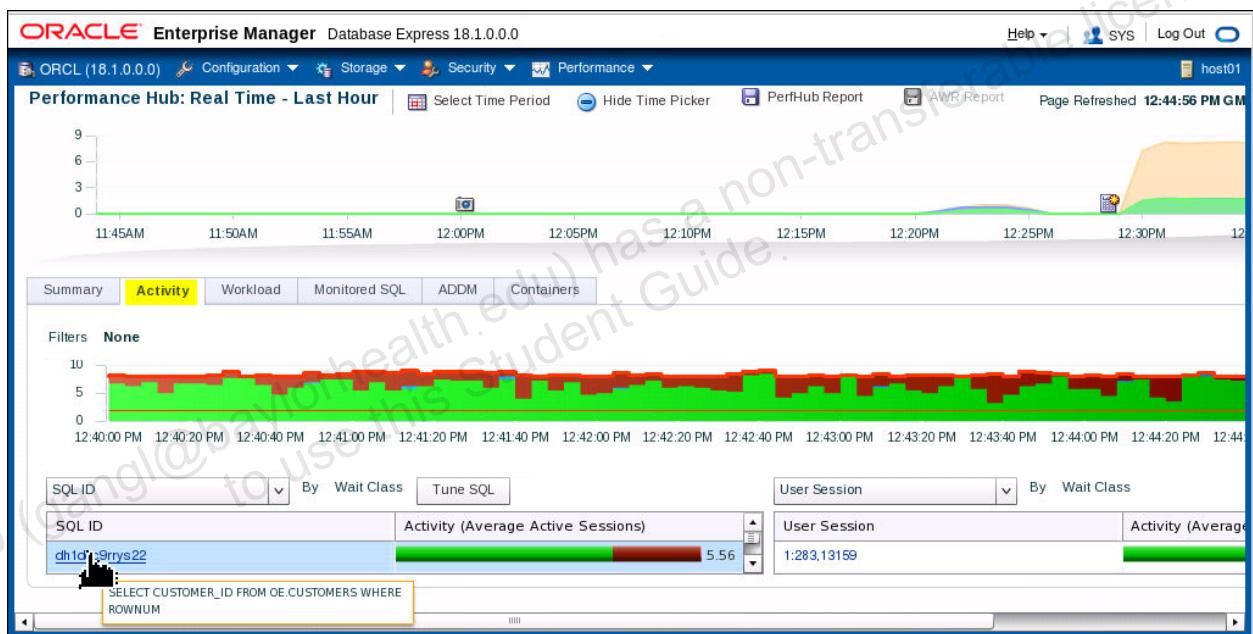
- h. Enter **sys** in the User Name field. Enter the password in the Password field and check AS SYSDBA. Then click **Login**.

*Observe that the ORACLE\_HOME for the database instance is 18.1.0.0 and the database name is ORCL.*

*Q/ In the Performance pane on the top right, in the Containers tab (second tab), what happens when you move your mouse to the name of one of the PDBs?*

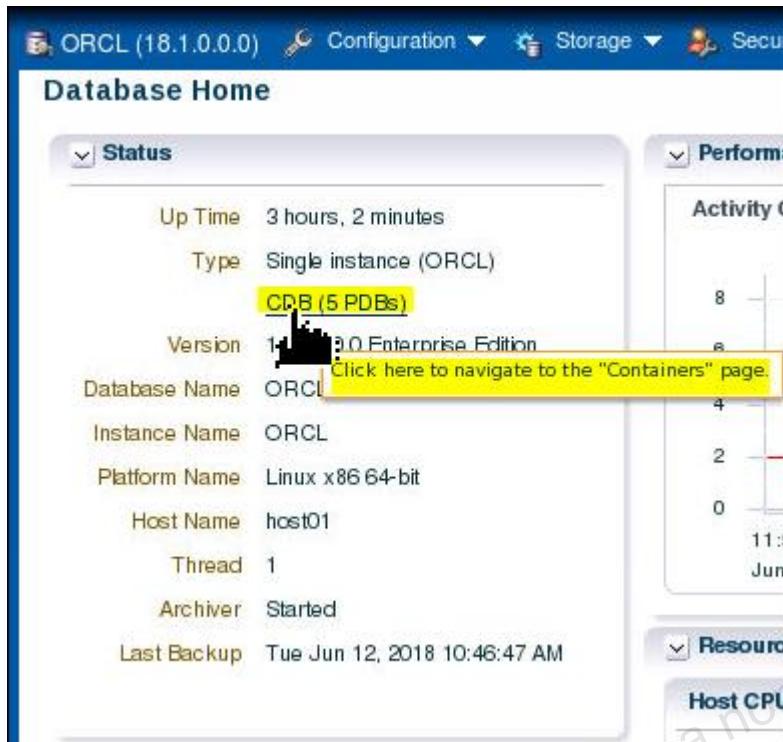
**A/ You have a global idea of resources consumed by each PDB and at which time.**

- i. To get all statements executed in all containers, click the Performance Hub option in the Performance menu. Then click the Activity tab.

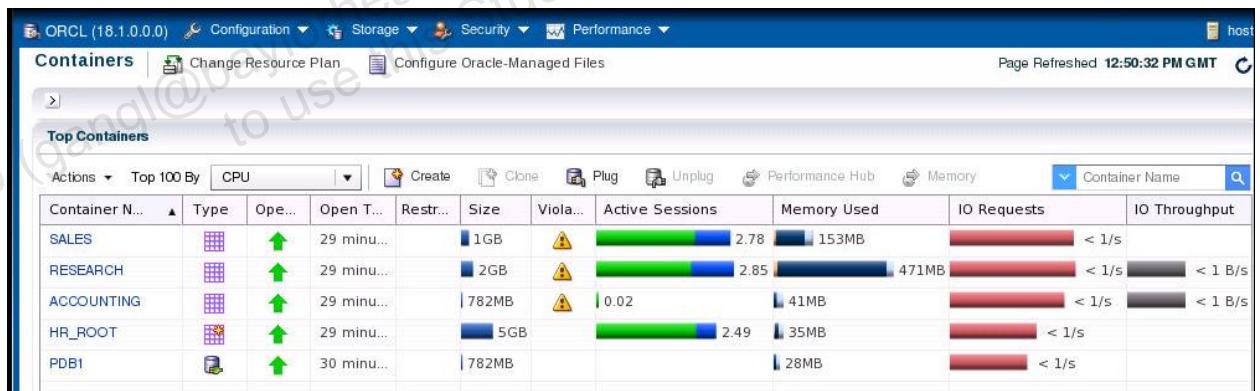


*Q/ Do you get details on Host CPU consumed by each PDB?*

**A/ 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.**



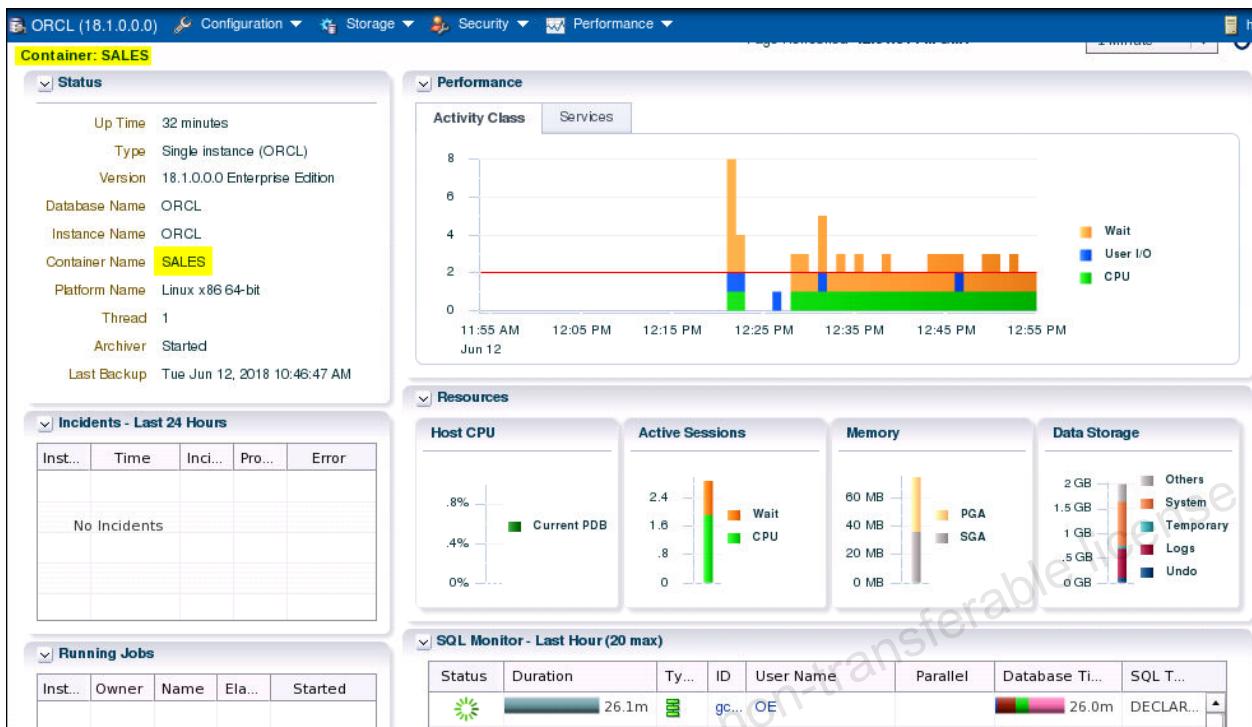
**Q2/ How would you get information on resources consumed for a specific PDB?**



**A2/ To get the summary on resources consumed by each PDB, click the link CDB (5PDDBs) .**

**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.**



6. Stop the workload by removing the \$HOME/labs/perf/runload file.

```
$ rm runload  
$
```

## Practice 10-2: Getting Performance ADDM 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 12.2. The centralized AWR serves as the repository for the performance data for the whole database—CDB root container and all its PDBs. Oracle Enterprise Manager 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.

Since Oracle Database 12.2, AWR data can be collected, viewed, and managed from both the CDB root level and 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 and the password in the Password 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. 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/perf/loop.sh  
...  
$
```

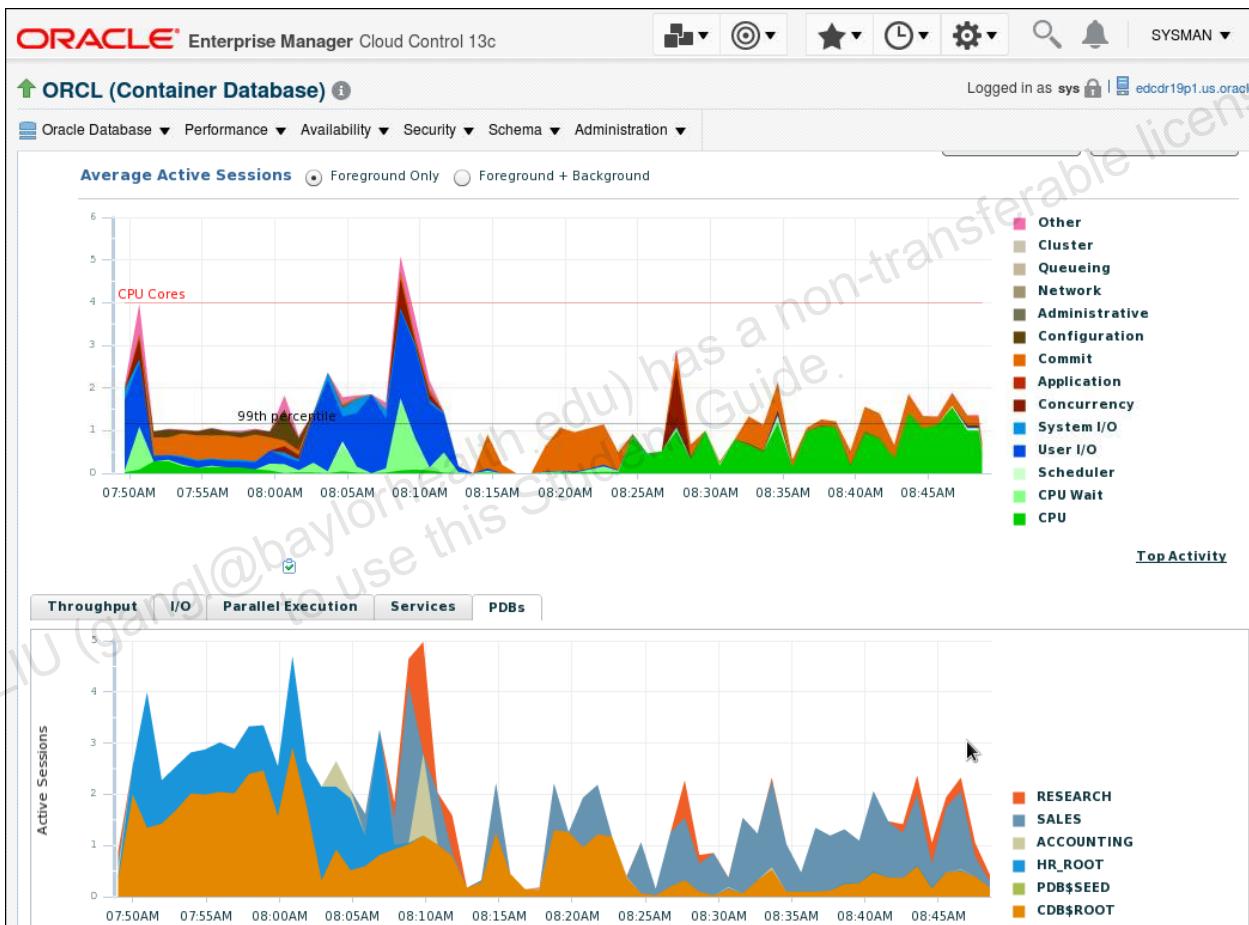
- b. In EM Cloud Control, once connected to `ORCL`, click Performance Home from the Performance menu. The `CREDORCL` preferred credentials appear. Click Login. To display the active sessions per container, click the Services or PDBs tab in the Active Sessions bottom section. If you do not see anything in the Active Sessions page, allow the Adobe Flash plugin first. If you still do not see anything, execute the following SQL command in `ORCL` on VM1.

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

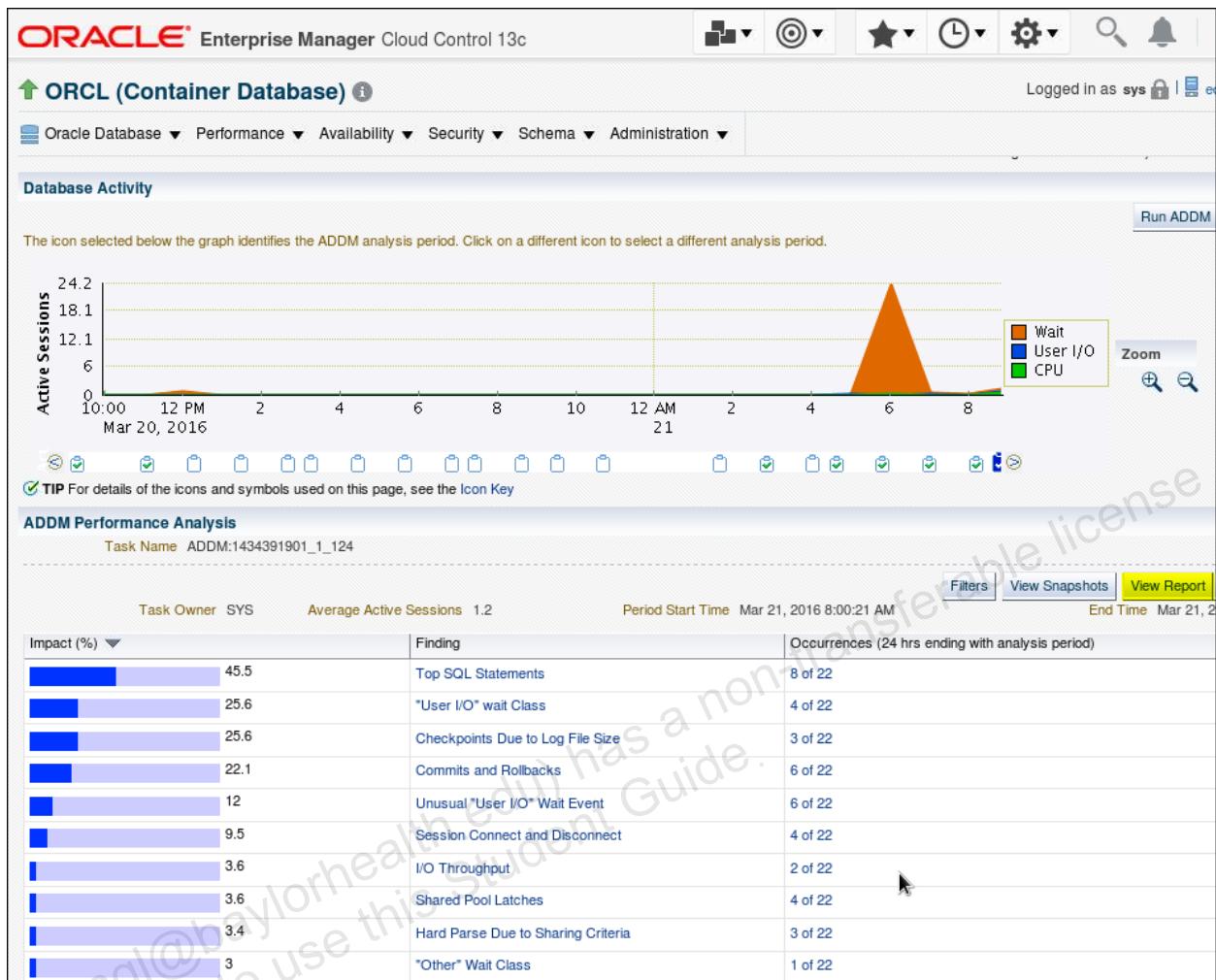
```
$ sqlplus / AS SYSDBA

SQL> ALTER USER dbsnmp IDENTIFIED BY password ACCOUNT UNLOCK
      CONTAINER=ALL;
2
User altered.

SQL> EXIT
$
```



- c. When you see the load growing, click Run ADDM Now.
- d. 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, in the ADDM Performance Analysis section, click View Report to display the recommendations.



Q/ Are the recommendations related to CDB level only?

#### Analysis Target

Database 'ORCL' with DB ID 1434391901.

Database version 18.0.0.0.0.

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

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_CDB_PDB_INSTANCE  
AWR_CDB_PDB_IN_SNAP  
AWR_CDB_RSRC_PDB_METRIC  
AWR_PDB_ACTIVE_SESS_HISTORY  
AWR_PDB_APPLY_SUMMARY  
AWR_PDB_ASH_SNAPSHOT  
...  
AWR_PDB_PDB_IN_SNAP  
...  
AWR_PDB_WAITSTAT  
AWR_PDB_WR_CONTROL  
AWR_PDB_WR_SETTINGS  
AWR_ROOT_PDB_INSTANCE  
AWR_ROOT_PDB_IN_SNAP  
AWR_ROOT_RSRC_PDB_METRIC  
  
155 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
AWR_ROOT_WR_SETTINGS

```

150 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.*

- 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: password
Connected.
SQL> SELECT dbid FROM v$pdbs;

          DBID
-----
3696566982

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
-----
70 1506381859 3696566982      13

SQL> SELECT snap_id, dbid, con_dbid, con_id
      FROM awr_pdb_pdb_in_snap;
2
      SNAP_ID      DBID      CON_DBID CON_ID
-----
1 3696566982 3696566982      13

```

SNAP_ID	DBID	CON_DBID	CON_ID
1	3696566982	3696566982	13

```

SQL> exec dbms_workload_repository.create_snapshot(FLUSH_LEVEL
=> 'ALL', DBID => 3696566982)

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
-----
70   1506381859 3696566982      13

SQL> SELECT snap_id, dbid, con_dbid, con_id
  FROM awr_pdb_pdb_in_snap;
2
SNAP_ID          DBID      CON_DBID CON_ID
-----
1   3696566982 3696566982      13
2   3696566982 3696566982      13

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
-----
70   1506381859 994516396      14
70   1506381859 1021175578      7
70   1506381859 1430544403     12
70   1506381859 1506381859      1
70   1506381859 2481570555     8
70   1506381859 3293973357      2
70 1506381859 3696566982 13
71   1506381859 994516396      14
71   1506381859 1021175578      7
71   1506381859 1430544403     12
71   1506381859 1506381859      1
71   1506381859 2481570555     8
71   1506381859 3293973357      2
```

```

71 1506381859 3696566982      13
 1 3696566982 3696566982      13
 2 3696566982 3696566982      13

```

488 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.**

- b. Create another snapshot.

```

SQL> CONNECT sys@sales AS SYSDBA
Enter password: password
Connected.
SQL> exec dbms_workload_repository.create_snapshot(FLUSH_LEVEL
=> 'ALL', DBID => 3696566982)

```

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 3696566982 3696566982      13
 2 3696566982 3696566982      13
 3 3696566982 3696566982      13

```

SQL>

*Q/ Did the PDB-level AWR snapshot collection performed in the SALES PDB whose DBID is 3696566982 collect AWR data for this single PDB?*

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

```
Enter password: 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.**

- c. Ask ADDM to analyze the period between the last two PDB-level snapshots.

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

```
Enter password: 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', 3696566982);
    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 ADVISED", line 5927
```

```
ORA-06512: at "SYS.PRVT ADVISED", line 1707
```

```

ORA-06512: at "SYS.DBMS_SYS_ERROR", line 79
ORA-06512: at "SYS.PRVT_ADVISOR", line 6892
ORA-06512: at "SYS.PRVT_ADVISOR", line 6950
ORA-06512: at "SYS.PRVT_ADVISOR", line 1480
ORA-06512: at "SYS.PRVT_ADVISOR", line 5889
ORA-06512: at "SYS.DBMS_ADVISOR", line 107
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', 3696566982);
    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 3696566982
and instance_id [] are not found in the current repository.
ORA-06512: at "SYS.DBMS_ADVISOR", line 212
ORA-06512: at "SYS.PRVT_ADVISOR", line 3389
ORA-06512: at "SYS.PRVT_ADVISOR", line 817
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 800
ORA-06512: at "SYS.PRVT_ADVISOR", line 3294
ORA-06512: at "SYS.DBMS_ADVISOR", line 262
ORA-06512: at "SYS.DBMS_ADVISOR", line 207
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', 70);
    dbms_advisor.set_task_parameter(:task_name,
                                    'END_SNAPSHOT', 71);
    dbms_advisor.set_task_parameter(:task_name,
                                    'DB_ID', 1506381859);
    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> EXIT
$
```

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 10-2. 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.

7. Stop the workload by killing the loop execution.

```
$ pgrep -lf loop
13691 /bin/sh /home/oracle/labs/perf/loop.sh
$ kill -9 13691
$
```

## Practice 10-3: 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 (`SALES` Session), configure the port number for Enterprise Manager Database Express for the `SALES` application PDB.

```
$ sqlplus sys@sales AS SYSDBA
Enter password: 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 is listening to the localhost (`yourserver`) by using TCP protocol, the port 5510 for `SALES`, the https presentation with RAW session data.

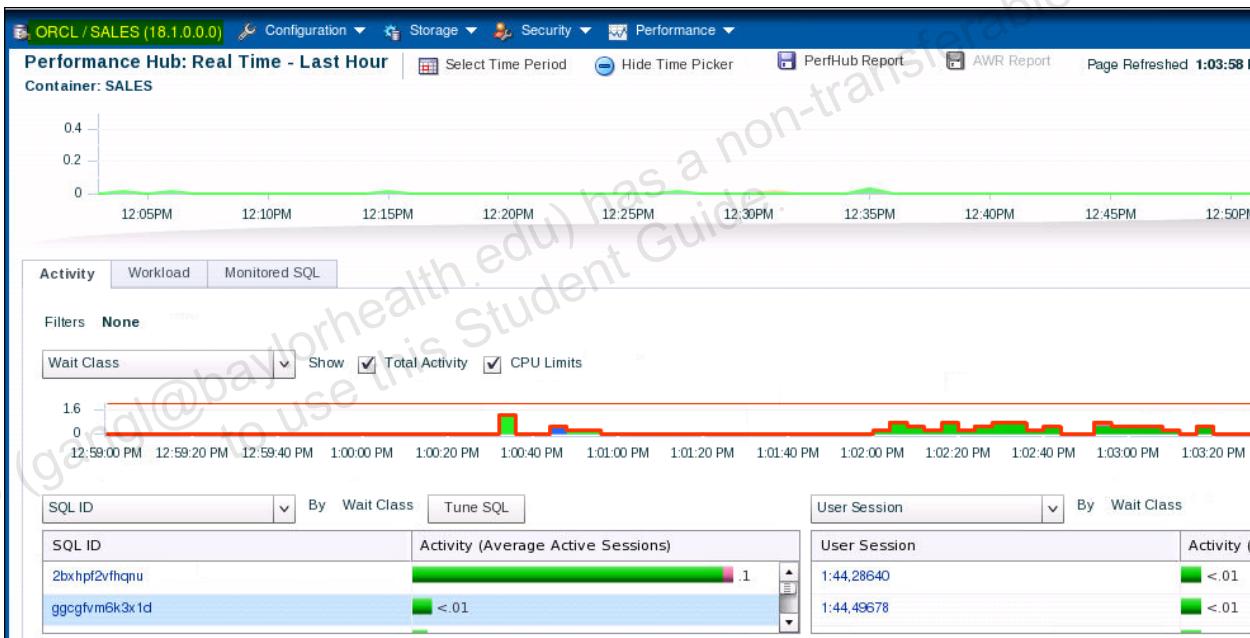
```
SQL> host lsnrctl status
...
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=your_server) (PORT=5500)) (Security=(my_wallet_directory=/u01/app/oracle/admin/ORCL/xdb_wallet)) (Presentation=HTTP) (Session=RAW))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=your_server) (PORT=5510)) (Security=(my_wallet_directory=/u01/app/oracle/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 18.1.0.0 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.**



**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 (RESEARCH Window), repeat the same operation for RESEARCH by using port 5520.

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

```
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 the SALES Window, 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.

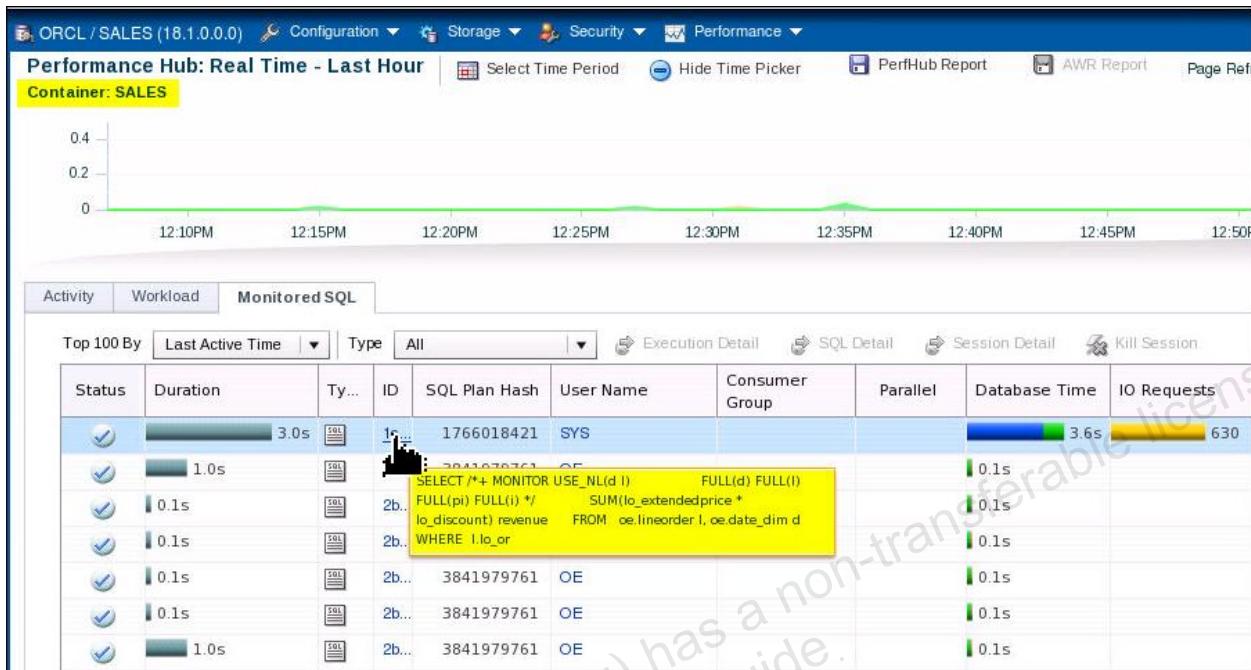
```
SQL> SET timing on
SQL> ALTER SESSION SET NLS_DATE_FORMAT='DD-MM-YYYY';

Session altered.

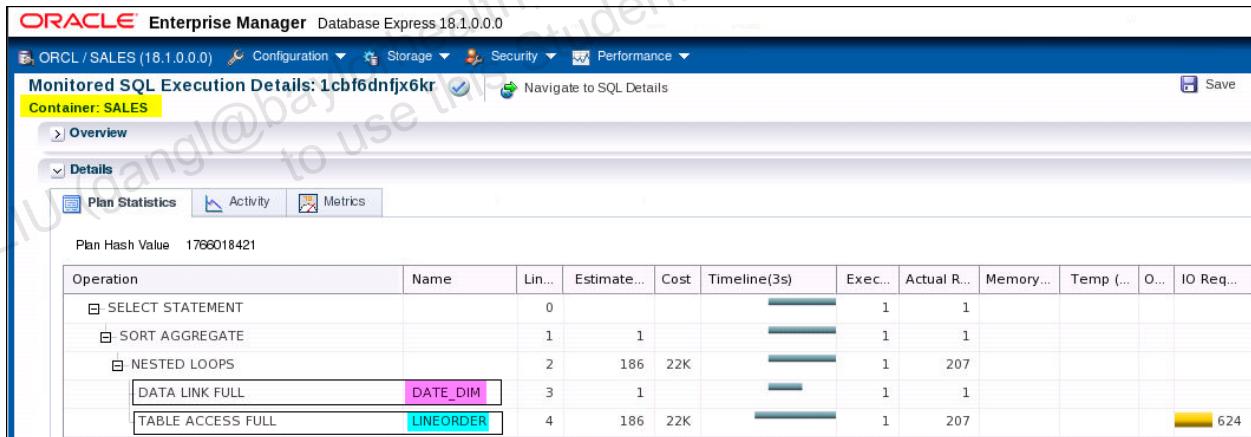
Elapsed: 00:00:00.12
SQL> SELECT /*+ MONITOR USE_NL(d l)
          FULL(d) FULL(l) FULL(pi) FULL(i) */
          SUM(lo_extendedprice * lo_discount) revenue
     FROM oe.lineorder l, oe.date_dim d
    WHERE l.lo_orderdate = d.d_datekey
      AND l.lo_discount BETWEEN 2 AND 3
      AND l.lo_quantity < 24
      AND d.d_date='December 24, 1996';
2      3      4      5      6      7      8
REVENUE
-----
939172011

Elapsed: 00:05:38.31
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.



4. Click the ID link of the SQL statement to display the execution plan and statistics.



*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 nonshared by application PDBs. This is the reason that 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 the RESEARCH Window, 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      8
REVENUE
-----
923381472

Elapsed: 00:08:37.60
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 and select Tune SQL. Enter the SQLTUNE\_request\_in\_RESEARCH name for the SQL tuning task. Then click OK.
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: 1cbf6dnfx6kr**  
Container: RESEARCH

**SQL Details**

Task Name	SQLTUNE_request_in_RESEARCH
Task Owner	SYS
SQL ID	1cbf6dnfx6kr
Schema	SYS
Container Name	RESEARCH

**SQL Text**

```

SELECT /*+ MONITOR USE_NL(d) */
       FULL(d) FULL(l) FULL(pi) FULL(i) */
       SUM(l.o_extendedprice * l.o_discount) revenue
  FROM oe.lineorder l, oe.date_dim d
 WHERE l.l_orderdate = d.d_datekey
   AND l.l_discount BETWEEN 2 AND 3
   AND l.l_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. A potentially better execution plan was found for this statement. The execution plan of this statement can be improved by creating one or more indices.
SQL Profile	
Index	

- 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.

The screenshot shows the Oracle Database Performance Advisor interface. In the 'Recommendation Details' section, under 'Container: RESEARCH', there is a 'Stale or Missing Statistics' section which states: 'The query optimizer relies on object statistics to generate execution plans. If these statistics are stale or missing, the optimizer does not have the necessary information to produce an optimal plan.' Below this, the 'Recommendation(s)' section lists: 'Table 'OE''.LINEORDER' was not analyzed. Consider collecting optimizer statistics for this table.' The 'Compare Explain Plans' section shows an 'Original Plan' table:

Operation	Object	Line ...	Predicate	Pruning
SELECT STATEMENT		0		
SORT AGGREGATE		1		
NESTED LOOPS		2		
DATA LINK FULL	DATE_DIM	3		
TABLE ACCESS FULL	LINEORDER	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.*



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')
```

- Verify that the statistics are collected for the tables in research.

```
SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: 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            12
LINEORDER            13
LINEORDER            8
LINEORDER           14    11219440      159737        98

Elapsed: 00:00:00.74

SQL>

```

**Q/ Are the statistics on the application shared tables collected in SALES?**

**A/ 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: 1cbf6dnfx6kr. Then click SQL Profile. Check the box to answer the question: "Do you want to implement new profile(s)?" and click OK. Then click OK again.
- d. 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.
- 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: password
Connected.
SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;

System altered.

SQL> ALTER SYSTEM FLUSH SHARED_POOL;

```

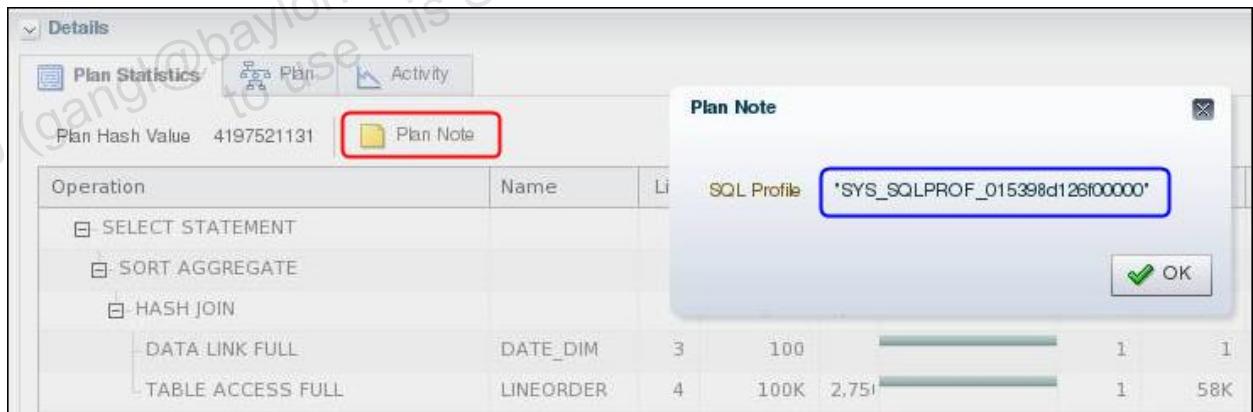
System altered.

```
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      8

REVENUE
-----
923381472

Elapsed: 00:00:05.76
SQL>
```

- 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.



**Q/ Is the execution time different from the first execution?**

**A/ 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: 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 1)
          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: 00:03:57.75

SQL>

**Q/ What do you notice?**

**A/ The performance is still not good.**

10. Collect the statistics in SALES 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: 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  |        | 8        |        |             |
| LINEORDER  | 13     | 5609720  | 79635  | 98          |
| LINEORDER  |        | 12       |        |             |
| LINEORDER  | 14     | 11219440 | 159737 | 98          |


Elapsed: 00:00:00.34
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 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: 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: 00:04:30.20  
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: password  
Connected.  
SQL> SELECT name, sql_text FROM dba_sql_profiles;  
  
NAME  
-----  
SQL_TEXT  
-----  
SYS_SQLPROF_016402b742500000  
SELECT /*+ MONITOR USE_NL(d l) FULL(d) FULL(l) FULL(pi) FULL(i)  
  
SQL> CONNECT sys@sales AS SYSDBA  
Enter password: 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 as yet on the DATE\_DIM shared table.

```
SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: 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';
2
TABLE_NAME          CON_ID    NUM_ROWS     BLOCKS AVG_ROW_LEN
-----  -----
DATE_DIM              8
DATE_DIM              14
DATE_DIM              12
DATE_DIM              13

Elapsed: 00:00:00.22

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';
2      3
```

TABLE_NAME	CON_ID	NUM_ROWS	BLOCKS	AVG_ROW_LEN
DATE_DIM	8	2556	39	100
DATE_DIM	13			
DATE_DIM	14			
DATE_DIM	12			

Elapsed: 00:00:00.83

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: 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')
*
ERROR at line 1:
ORA-20000: Unable to analyze the object, the operation cannot
happen outside App Root
ORA-06512: at "SYS.DBMS_STATS", line 39094
ORA-06512: at "SYS.DBMS_STATS", line 38382
ORA-06512: at "SYS.DBMS_STATS", line 37739
ORA-06512: at "SYS.DBMS_STATS", line 38530
ORA-06512: at "SYS.DBMS_STATS", line 39076
ORA-06512: at line 1

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   | 13     |          |        |             |


SQL> EXIT
$
```

A/ Values would be set to null because the rows of the segment are stored in the application root tablespace.

## Practice 10-4: Configuring CDB Fleet

---

### Overview

In this practice, you will create a CDB fleet with its CDB lead and its CDB members. Then you monitor PDBs across all CDBs from the CDB lead.

### Tasks

- Before starting the practice, execute the `$HOME/labs/perf/glogin_10.sh` shell script. It sets formatting for all columns selected in queries.

```
$ $HOME/labs/perf/glogin_10.sh
$
```

- Set `ORCL` as the CDB lead in a CDB fleet. This session is the CDB Lead Session.

```
$ sqlplus / AS SYSDBA

SQL> SELECT property_name, property_value
      FROM database_properties
     WHERE property_name LIKE 'LEAD%';
2      3
no rows selected

SQL> ALTER DATABASE SET LEAD_CDB = true;

Database altered.

SQL>
```

- Check that `ORCL` is the CDB lead of a CDB fleet.

```
SQL> SELECT property_name, property_value
      FROM database_properties
     WHERE property_name LIKE 'LEAD%';
2      3
PROPERTY_NAME          PROPERTY_VALUE
-----
LEAD_CDB                TRUE

SQL> SELECT sys_context('userenv', 'is_lead_cdb') LEAD_CDB
      FROM dual;
2
LEAD_CDB
-----
YES

SQL>
```

**Q/ How can the information of the lead role of the CDB be retrieved?**

**A/ The information of the lead role of the CDB is visible in the alert log file.**

```
SQL> HOST tail -5
/u01/app/oracle/diag/rdbms/orcl/ORCL/trace/alert_ORCL.log
2018-05-07T05:31:44.793610+00:00
ALTER DATABASE SET LEAD_CDB = true
2018-05-07T05:31:44.793724+00:00
The role of current CDB in the Fleet is: LEAD
Completed: ALTER DATABASE SET LEAD_CDB = true
SQL>
```

- Grant the user that will be used in the database link in the CDB members the required system privileges.

```
SQL> GRANT connect, resource, create pluggable database,
      sysoper, unlimited tablespace, select any table
      TO system CONTAINER=ALL;
2      3
Grant succeeded.

SQL>
```

- In another terminal window, connect to CDB18 to define CDB18 as a member of the fleet. This session is the CDB Member Session. Use the common user SYSTEM in the CDB lead to be used for future CDB members to exchange with the CDB lead through database links.

  - Check first whether CDB18 is a member of the fleet.

```
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ $HOME/labs/perf/glogin_10.sh
$
$ sqlplus / AS SYSDBA

SQL> SELECT sys_context('userenv', 'is_lead_cdb') LEAD_CDB
      FROM dual;
2
LEAD_CDB
-----
NO

SQL> SELECT sys_context('userenv', 'is_member_cdb') MEMBER_CDB
      FROM dual;
2
```

```
 MEMBER_CDB  
-----  
NO  
  
SQL>
```

- b. Create the fixed user database link by using a user identical to the user created in the CDB lead.

```
SQL> CREATE PUBLIC DATABASE LINK lorcl  
      CONNECT TO system IDENTIFIED BY password  
      USING 'ORCL';  
2   3   4  
Database link created.  
  
SQL>
```

- c. Set CDB18 as a member of the fleet.

```
SQL> ALTER DATABASE SET LEAD_CDB_URI = 'dblink:lorcl';  
  
Database altered.  
  
SQL> SELECT sys_context('userenv','is_member_cdb') MEMBER_CDB  
      FROM dual;  
2  
MEMBER_CDB  
-----  
YES  
  
SQL> SELECT property_name, property_value  
      FROM database_properties  
      WHERE property_name LIKE 'LEAD%';  
2   3  
PROPERTY_NAME                      PROPERTY_VALUE  
-----  
LEAD_CDB_URI                         dblink:lorcl  
  
SQL> EXIT  
$
```

*Q/ How can the information of the member role of the CDB be retrieved?*

**A/ The information of the member role of the CDB is visible in the alert log file.**

```
$ tail -5
/u01/app/oracle/diag/rdbms/cdb18/CDB18/trace/alert_CDB18.log
2018-05-07T05:33:52.566968+00:00
ALTER DATABASE SET LEAD_CDB_URI = 'dblink:lorcl'
2018-05-07T05:33:52.843490+00:00
The role of current CDB in the Fleet is: MEMBER
Completed: ALTER DATABASE SET LEAD_CDB_URI = 'dblink:lorcl'
...
$
```

- d. In the CDB Lead Session, list the PDBs of the CDB members. Connect to the CDB lead and check if you can report information from all PDBs of all CDBs in the fleet.

```
SQL> SHOW PDBS

CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
  2 PDB$SEED           READ ONLY NO
  3 PDB1              READ WRITE NO
  5 HR_ROOT            READ WRITE NO
  6 PDB18             MOUNTED
  9 ACCOUNTING        READ WRITE NO
 10 SALES             READ WRITE NO
 11 RESEARCH           READ WRITE NO
SQL>
```

```
SQL> SELECT name, open_mode, member_cdb "MEMBER_CDB"
  FROM v$containers;
  2
NAME      OPEN_MODE MEMBER_CDB
----- -----
CDB$ROOT    READ WRITE NO
PDB$SEED    READ ONLY NO
PDB1        READ WRITE NO
CDB18      MOUNTED YES
HR_ROOT     READ WRITE NO
PDB18      MOUNTED NO
ACCOUNTING READ WRITE NO
SALES       READ WRITE NO
RESEARCH    READ WRITE NO

9 rows selected.

SQL> SELECT pdb_id, pdb_name, status,
```

```

      is_proxy_pdb "IS_PROXY_PDB"
FROM    dba_pdbs;
2      3
PDB_ID PDB_NAME      STATUS   IS_PROXY_PDB
-----
3     PDB1          NORMAL   NO
2     PDB$SEED       NORMAL   NO
9     ACCOUNTING    NORMAL   NO
5     HR_ROOT        NORMAL   NO
10    SALES          NORMAL   NO
11    RESEARCH       NORMAL   NO
4     CDB18          STUB     YES
6     PDB18          STUB     YES

8 rows selected.

SQL>
```

**Q/ Why is the `OPEN_MODE` value of the `PDB18` members `MOUNTED` in the CDB lead whereas they are `OPENED`, respectively, in `CDB18`?**

**A/ Members can only be opened in the CDB where they physically exist. The `STUB` value in the `STATUS` column defines the CDB and its PDBs as members in the fleet.**

## 6. Unset the CDB lead.

```

SQL> ALTER DATABASE SET LEAD_CDB = false;

Database altered.

SQL> SELECT property_name, property_value
  FROM database_properties
 WHERE property_name LIKE 'LEAD%';
2      3      4
PROPERTY_NAME                      PROPERTY_VALUE
-----
LEAD_CDB                            FALSE

SQL> SELECT sys_context('userenv', 'is_lead_cdb') LEAD_CDB
  FROM dual;
2
LEAD_CDB
-----
```

```
NO
SQL>
```

*Q/ Does it deactivate CDB members?*

**A/ No, because the membership property is still maintained in CDB members.  
The CDB members can be re-attached to ORCL once ORCL will have been reset as the lead of the fleet.**

In the CDB Member Session:

```
SQL> SELECT sys_context('userenv', 'is_member_cdb') MEMBER_CDB
      FROM dual;

MEMBER_CDB
-----
YES

SQL>
```

7. In the CDB Lead Session, reset the CDB lead.

```
SQL> ALTER DATABASE SET LEAD_CDB = true;
Database altered.

SQL>
```

*Q/ Do the members of the fleet reappear in the list of monitored PDBs?*

```
SQL> SELECT distinct (pdb_id), pdb_name, status,
      is_proxy_pdb "IS_PROXY_PDB"
      FROM cdb_pdbs;
2      3
PDB_ID PDB_NAME      STATUS    IS_PROXY_PDB
----- -----
 4  PDB1        NORMAL   NO
 2  PDB$SEED    NORMAL   NO
11  RESEARCH    NORMAL   NO
 6  ACCOUNTING  NORMAL   NO
 7  SALES       NORMAL   NO
 3  HR_ROOT     NORMAL   NO
6 rows selected.

SQL>
```

A/ Yes, and if there were several CDB members, they would automatically register to the lead, one by one.

```
SQL> SELECT distinct (pdb_id), pdb_name, status,
           is_proxy_pdb "IS_PROXY_PDB"
      FROM cdb_pdbs;
2      3
PDB_ID PDB_NAME      STATUS IS_PROXY_PDB
-----
 5  CDB18        STUB     YES
 4  PDB1         NORMAL   NO
 3  HR_ROOT      NORMAL   NO
 8  PDB18        STUB     YES
 2  PDB$SEED     NORMAL   NO
11  RESEARCH     NORMAL   NO
 6  ACCOUNTING   NORMAL   NO
 7  SALES        NORMAL   NO
8 rows selected.

SQL>
```

- In the CDB Member Session, unset a CDB member: release the membership of CDB18 from the fleet. You may prefer to associate it to another fleet.

```
SQL> ALTER DATABASE SET LEAD_CDB_URI = '';
Database altered.

SQL> EXIT
$
```

- In the CDB Lead Session, check that the CDB lead does not monitor cdb1 anymore.

```
SQL> SELECT distinct (pdb_id), pdb_name, status,
           is_proxy_pdb "IS_PROXY_PDB"
      FROM cdb_pdbs;
2      3
PDB_ID PDB_NAME      STATUS IS_PROXY_PDB
-----
 4  PDB1         NORMAL   NO
 2  PDB$SEED     NORMAL   NO
11  RESEARCH     NORMAL   NO
 6  ACCOUNTING   NORMAL   NO
 7  SALES        NORMAL   NO
 3  HR_ROOT      NORMAL   NO
```

```
6 rows selected.
```

```
SQL> EXIT  
$
```

10. Execute the \$HOME/labs/perf/cleanup\_fleet.sh shell script to unset the CDB fleet lead and members.

```
$ $HOME/labs/perf/cleanup_fleet.sh  
...  
$
```

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.

## **Practices for Lesson 11: Resources Allocation**

## Practices for Lesson 11: Overview

---

### Practices Overview

In the practices for this lesson, you create two CDB Resource Manager plans and performance profiles and associated directives to limit CPU resources used by two PDBs. You will also benefit from Resource Manager new directives to avoid excessive PGA consumption.

## Practice 11-1: Managing PDB Performance Profiles

---

### 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 PDB1 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. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. Then execute the `$HOME/labs/RM/glogin_11.sh` shell script. It appends COL commands to format all columns selected in queries.

```
$ $HOME/labs/RM/glogin_11.sh
$
```

3. Then execute the `$HOME/labs/RM/setup_hr_app.sh` shell script to create the HR\_ROOT application container and its SALES and RESEARCH application PDBs.

```
$ $HOME/labs/RM/setup_hr_app.sh
...
$
```

4. Open a terminal window (it will be referred to as SALES Window) 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 `$HOME/labs/RM/create_burn_cpu.sql` script to create the procedure after connecting to SALES. If SALES and RESEARCH application PDBs are mounted, open them after opening HR\_ROOT.

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

Enter password : password

SQL> @$HOME/labs/RM/create_burn_cpu.sql
```

```

...
Procedure created.

SQL>
```

5. Open a second terminal window (it will be referred to as RESEARCH Window) 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 \$HOME/labs/RM/create\_burn\_cpu.sql script to create the procedure after connecting to RESEARCH.

```

$ . oraenv
ORACLE_SID = [oracle] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ sqlplus system@research

Enter password : password

SQL> @"$HOME/labs/RM/create_burn_cpu.sql
...
Procedure created.

SQL>
```

6. From SALES Window, execute the \$HOME/labs/RM/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 percent CPU limit to the PDB being assigned the performance profile directive and prof\_low in the HR\_plan plan gives two shares and 50 percent CPU limit to the PDB being assigned the performance profile.

```

SQL> @"$HOME/labs/RM/setup_prof_directives.sql
...
SQL>
```

7. Still from SALES Window, make sure both performance profiles in the Toys\_plan plan and associated directives were created correctly. Execute the \$HOME/labs/RM/display\_prof\_directives.sql script.

```

SQL> @"$HOME/labs/RM/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

SQL>

```

8. Still from SALES Window, 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.**

9. **First test:**

- a. Still from SALES Window, connect as the SYS user in SALES.

```

SQL> CONNECT sys@sales AS SYSDBA
Enter password: password
Connected.
SQL>

```

- b. Set the appropriate performance profile.

```

SQL> ALTER SYSTEM SET db_performance_profile='PROF_HIGH'
              SCOPE = spfile;
2
System altered.

SQL>

```

- c. 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

- d. Connect as SYSTEM and set SERVEROUTPUT variable to ON.

```
SQL> CONNECT system@sales
```

Enter password: *password*

Connected.

```
SQL> SET serveroutput on
```

```
SQL>
```

- e. From RESEARCH Window, connect as the SYS user in research.

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

Enter password: *password*

Connected.

```
SQL>
```

- f. Set the appropriate performance profile.

```
SQL> ALTER SYSTEM SET db_performance_profile='PROF_LOW'
      SCOPE = spfile;
```

2

System altered.

```
SQL>
```

- g. Restart the PDB. If the command used in the following code box does not end up, use the **ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE**.

```
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

- h. Connect as SYSTEM and set SERVEROUTPUT variable to ON.

```
SQL> CONNECT system@research
Enter password: password
Connected.
SQL> SET serveroutput on
SQL>
```

- i. **DO NOT WAIT AND GO TO THE NEXT STEP IMMEDIATELY:** From SALES Window, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU:    76.4 Wall:  113.8 k: 2000000000

PL/SQL procedure successfully completed.

SQL>
```

- j. From RESEARCH Window, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU:    76.4 Wall:  189.2 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.**

#### 10. Second test:

- a. From SALES Window, 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: password
Connected.
```

```
SQL>
SQL> ALTER SYSTEM SET db_performance_profile='' SCOPE = spfile;

System altered.

SQL> CONNECT sys@research AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM SET db_performance_profile='' SCOPE = spfile;

System altered.

SQL> CONNECT sys@hr_root AS SYSDBA
Enter password: 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 SERVEROUTPUT variable to ON.

```
SQL> CONNECT system@sales
Enter password: password
Connected.
SQL> SET serveroutput on
SQL>
```

- d. From RESEARCH Window, connect as SYSTEM and set SERVEROUTPUT variable to ON. Be ready to execute the CPU burner procedure.

```
$ sqlplus system@research
Enter password: password
SQL> SET serveroutput on
SQL>
```

- e. Open a third terminal window (it will be referred to as PDB1 Window) to connect to PDB1 in ORCL and create a PL/SQL procedure that burns CPU as the SYSTEM user.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL
The Oracle base has been set to /u01/app/oracle
$ sqlplus system@PDB1
Enter password: password
SQL> @"$HOME/labs/RM/create_burn_cpu.sql
...
Procedure created.

SQL>
```

- f. Connect to PDB1 as user SYS and set the performance profile to PROF\_HIGH.

```
SQL> CONNECT sys@PDB1 AS SYSDBA
Enter password: password
Connected.
SQL> ALTER SYSTEM SET db_performance_profile='PROF_HIGH'
          SCOPE = spfile;
2
System altered.

SQL>
```

- g. Restart the PDB. If the command used in the following code box does not end up, use the **ALTER PLUGGABLE DATABASE CLOSE IMMEDIATE**

```
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

- h. Connect as SYSTEM and set SERVEROUTPUT 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@PDB1
```

Enter password: *password*

Connected.

```
SQL> SET serveroutput on
```

```
SQL> EXEC Burn_CPU_For_RM_Demo()
```

CPU: 73.1 Wall: 112.0 k: 2000000000

PL/SQL procedure successfully completed.

```
SQL>
```

- i. From SALES Window, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
```

CPU: 73.2 Wall: 223.7 k: 2000000000

PL/SQL procedure successfully completed.

```
SQL> EXIT
```

\$

- j. From RESEARCH Window, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
```

CPU: 73.7 Wall: 225.5 k: 2000000000

PL/SQL procedure successfully completed.

```
SQL> EXIT
```

\$

Q/ What do you observe?

A/ **The procedure in PDB1 finishes its execution much earlier than both application PDBs and has consumed much less wall-clock time. This is expected because PDB1 is granted four shares of CPU.**

**The procedures 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.**

11. From the session left open, 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
$
```

## Practice 11-2: Managing Resource Allocation Between PDBs

### Overview

In this practice, you will create two CDB Resource Manager plans and associated directives to limit CPU resources used by two PDBs.

### Tasks

1. Clean up your environment by executing the \$HOME/labs/RM/rsrc\_cleanup.sh shell script. The script will close all PDBs except SALES and RESEARCH.

```
$ $HOME/labs/RM/rsrc_cleanup.sh  
...  
$
```

2. From SALES Window, create a PL/SQL procedure that burns CPU in SALES as the SYSTEM user. You can use the \$HOME/labs/RM/create\_burn\_cpu.sql script to create the procedure after connecting to SALES.

```
$ sqlplus system@SALES  
  
Enter password: password  
  
SQL> @"$HOME/labs/RM/create_burn_cpu.sql  
...  
Procedure created.  
  
SQL>
```

3. From the RESEARCH Window, create a PL/SQL procedure that burns CPU in RESEARCH as the SYSTEM user. You can use the \$HOME/labs/RM/create\_burn\_cpu.sql script to create the procedure after connecting to RESEARCH.

```
$ sqlplus system@RESEARCH  
  
Enter password: password  
  
SQL> @"$HOME/labs/RM/create_burn_cpu.sql  
...  
Procedure created.  
  
SQL>
```

4. From any of the two windows, create two new CDB plans called FAIRPLAN and UNFAIRPLAN.

FAIRPLAN should give one share to both SALES and RESEARCH and UNFAIRPLAN should give one share to SALES and five shares to RESEARCH.

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

Session altered.

SQL> EXEC DBMS_Resource_Manager.Clear_Pending_Area()

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_Resource_Manager.Create_Pending_Area()

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_Resource_Manager.Create_CDB_Plan('fairplan', 'One
share each')

PL/SQL procedure successfully completed.

SQL> EXEC
DBMS_Resource_Manager.Create_CDB_Plan_Directive('fairplan',
'sales', shares => 1)

PL/SQL procedure successfully completed.

SQL> EXEC
DBMS_Resource_Manager.Create_CDB_Plan_Directive('fairplan',
'research', shares => 1)

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_Resource_Manager.Create_CDB_Plan('unfairplan',
'one share to sales and five to research')

PL/SQL procedure successfully completed.

SQL> EXEC
DBMS_Resource_Manager.Create_CDB_Plan_Directive('unfairplan',
'sales', shares => 1)

PL/SQL procedure successfully completed.

SQL> EXEC
DBMS_Resource_Manager.Create_CDB_Plan_Directive('unfairplan',
'research', shares => 5)
```

```
PL/SQL procedure successfully completed.
```

```
SQL> EXEC DBMS_Resource_Manager.Validate_Pending_Area()
```

```
PL/SQL procedure successfully completed.
```

```
SQL> EXEC DBMS_Resource_Manager.Submit_Pending_Area()
```

```
PL/SQL procedure successfully completed.
```

```
SQL>
```

- Still from the window, make sure both plans and associated directives were created correctly.

```
SQL> SELECT Plan from CDB_CDB_Rsrc_Plans
      WHERE Con_ID = 1 AND Plan IN ('FAIRPLAN', 'UNFAIRPLAN')
      ORDER BY 1;
```

```
2      3
```

```
PLAN
```

```
-----
```

```
FAIRPLAN
```

```
UNFAIRPLAN
```

```
SQL> SELECT plan, pluggable_database, shares
      FROM cdb_cdb_rsrc_plan_directives
      WHERE con_id = 1
      AND plan IN ('FAIRPLAN', 'UNFAIRPLAN')
      AND pluggable_database IN ('SALES', 'RESEARCH')
      ORDER BY 1, 2;
```

```
2      3      4      5      6
```

PLAN	PLUGGABLE_DATABASE	SHARES
------	--------------------	--------

```
-----
```

FAIRPLAN	RESEARCH	1
----------	----------	---

FAIRPLAN	SALES	1
----------	-------	---

UNFAIRPLAN	RESEARCH	5
------------	----------	---

UNFAIRPLAN	SALES	1
------------	-------	---

```
SQL>
```

- Still from the SALES window, activate the CDB plan FAIRPLAN.

```
SQL> CONNECT / AS SYSDBA
```

```
Enter password: password
```

```
Connected.
```

```
SQL> ALTER SYSTEM SET resource_manager_plan = fairplan;
```

System altered.

```
SQL> SELECT name FROM v$rsrc_plan WHERE con_id = 1;
```

NAME

-----  
FAIRPLAN

SQL>

- From the SALES window, connect as the SYSTEM user in SALES and set SERVEROUTPUT variable to ON.

```
SQL> CONNECT system@SALES
Enter password: password
Connected.
SQL> set serveroutput on
SQL>
```

- From the RESEARCH window, connect as the SYSTEM user in RESEARCH and set SERVEROUTPUT variable to ON.

```
SQL> CONNECT system@RESEARCH
Enter password: password
Connected.
SQL> set serveroutput on
SQL>
```

- DO NOT WAIT AND GO TO STEP 10 RIGHT AFTER:** From the SALES window, execute the CPU burner procedure you created in step 2.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU: 73.2 Wall: 149.3 k: 2000000000

PL/SQL procedure successfully completed.

SQL>
```

- From the RESEARCH window, execute the CPU burner procedure you created in step 3.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU: 73.4 Wall: 145.0 k: 2000000000

PL/SQL procedure successfully completed.

SQL>
```

11. What do you observe?

Both procedures finish their execution almost at the same time, and both have consumed almost the same CPU and wall-clock time during their execution.  
This is expected because each PDB is receiving one share of CPU.

12. From the SALES window, connect as user `SYS` in the CDB root and change the Resource Manager plan to `UNFAIRPLAN`.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> ALTER SYSTEM SET resource_manager_plan = unfairplan;
System altered.

SQL> SELECT name FROM v$rsrc_plan WHERE con_id = 1;

NAME
-----
UNFAIRPLAN

SQL>
```

13. Go to Step 14 right after starting the execution of the procedure in Step 13: From the SALES window, connect as user `SYSTEM` in `SALES` and execute the CPU burner procedure you created in step 2.

```
SQL> CONNECT system@SALES
Enter password: password
Connected.
SQL> set serveroutput on
SQL>
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU: 73.2 Wall: 148.9 k: 2000000000

PL/SQL procedure successfully completed.

SQL>
```

14. From the RESEARCH window, execute the CPU burner procedure you created in step 3.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
CPU: 73.2 Wall: 89.2k : 2000000000

PL/SQL procedure successfully completed.

SQL> EXIT
$
```

15. What do you observe?

Now, execution of the CPU burner procedure takes much longer to execute in SALES than in RESEARCH.

This is expected because RESEARCH is assigned five shares while SALES only one.

However, the difference is not five times slower simply because once the procedure is executed in RESEARCH, all CPU cycles go to SALES.

16. Make sure you set the CDB plan back to its default.

```
SQL> CONNECT / AS SYSDBA

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
$
```

## Practice 11-3: Avoiding Excessive Session PGA Memory Usage in PDBs

---

### Overview

In this practice, you will use Resource Manager to avoid excessive PGA memory consumption in SALES.

### Tasks

1. Execute the \$HOME/labs/RM/setup\_OE.sh shell script to create the HR\_ROOT application container and its SALES and RESEARCH application PDBs. It also creates a large table, OE.LINEORDER in SALES.

```
$ $HOME/labs/RM/setup_OE.sh
...
$
```

2. Some users informed you that their query performance decreased in SALES when OE was generating some reporting queries on OE.LINEORDER table. You don't know yet which limit to set on this user.
  - a. In the SALES window, start the reporting query on OE.LINEORDER table.

```
$ sqlplus oe@SALES

Enter password : password

SQL> SELECT * FROM oe.lineorder
      ORDER BY lo_orderkey,lo_custkey,lo_partkey,lo_suppkey;
...
SQL>
```

- b. Meanwhile, in another terminal window (Window2), check the PGA memory used by OE.

```
$ sqlplus system@SALES

Enter password : password

SQL> SELECT username, name, value
      FROM v$sesstat m, v$statname a, v$session s
      WHERE m.statistic# = a.statistic# AND m.sid = s.sid
      AND a.name LIKE 'session pga memory%'
      AND username = 'OE';
2      3      4      5
USERNAME NAME                                VALUE
----- -----

```

```

OE      session pga memory          97380072
OE      session pga memory max     97380072

SQL>

```

*Q/ What is the PGA memory maximum used by OE?*

*A/ The maximum PGA used is about 96Mb.*

- c. Use the Resource Manager directive to limit the PGA memory usage for OE in SALES to 80Mb.
  - 1) Still in Window2, use the \$HOME/labs/RM/script\_rm.sql SQL script that creates the pga\_plan resource manager plan, the reporting\_users consumer group, and associates OE to the reporting\_users consumer group.

```

SQL> @$HOME/labs/RM/script_rm.sql
...
PLAN      GROUP_OR_SUBPLAN      SESSION_PGA_LIMIT
-----
PGA_PLAN  OTHER_GROUPS        200

SQL>

```

- 2) Now set the PGA limit for the reporting\_users and activate the plan in SALES.

```

SQL> EXEC dbms_resource_manager.clear_pending_area()
PL/SQL procedure successfully completed.

SQL> EXEC dbms_resource_manager.create_pending_area()
PL/SQL procedure successfully completed.

SQL> EXEC dbms_resource_manager.create_plan_directive(
      'PGA_plan', 'Reporting_Users', session_pga_limit => 3)
PL/SQL procedure successfully completed.

SQL> EXEC dbms_resource_manager.validate_pending_area()
PL/SQL procedure successfully completed.

SQL> EXEC dbms_resource_manager.submit_pending_area()
PL/SQL procedure successfully completed.

```

```

SQL> ALTER SYSTEM SET resource_manager_plan='PGA_PLAN';

System altered.

SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;

System altered.

SQL> SELECT plan, group_or_subplan, session_pga_limit
   FROM dba_rsrc_plan_directives
 WHERE plan = 'PGA_PLAN';
 2      3
PLAN      GROUP_OR_SUBPLAN      SESSION_PGA_LIMIT
-----  -----
PGA_PLAN    OTHER_GROUPS          200
PGA_PLAN    REPORTING_USERS        3

SQL>

```

- d. Back in the SALES window, in OE session, interrupt the query and restart the query on OE.LINEORDER table. Remain connected until you complete step f.

```

SQL> SELECT * FROM oe.lineorder
      ORDER BY LO_ORDERKEY,LO_CUSTKEY,LO_PARTKEY,LO_SUPPKEY;
 2  SELECT * FROM oe.lineorder
*
ERROR at line 1:
ORA-10260: PGA limit (3 MB) exceeded - process terminated

SQL> EXIT
$ 

```

- e. Back in Window2 in OE session, check the PGA memory limit by OE. You can then quit the SALES window session.

```

SQL> SELECT username, name, value
      FROM v$sesstat m, v$statname a, v$session s
     WHERE m.statistic# = a.statistic# AND m.sid = s.sid
       AND a.name LIKE 'session pga memory%'
       AND username = 'OE';
 2      3      4      5
USERNAME      NAME          VALUE
-----  -----
OE            session pga memory      2549480
OE            session pga memory max  31319784

```

```
SQL> EXIT  
$
```

3. Clean up the Resource Manager plan.

```
$ $HOME/labs/RM/cleanup_rm.sh  
...  
$
```

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.

## **Practices for Lesson 12: Data Movement**

## Practices for Lesson 12: Overview

---

### Practices Overview

In these practices, you complete export and import operations between non-CDBs and PDBs by using the FULL TRANSPORTABLE feature.

## Practice 12-1: Performing a Full Transportable Export/Import from a 12c Non-CDB into an 18c PDB

### Overview

In this practice, you export an Oracle Database 12c non-CDB and import it into an Oracle Database 18c PDB by using the Oracle Data Pump FULL TRANSPORTABLE feature.

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. In NONCDB, set the `USERS` user-defined tablespace that stores the `HR.EMPLOYEES` table to `READ ONLY` before exporting.

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

SQL> SELECT count(*) FROM hr.employees;

COUNT (*)
-----
          107

SQL> ALTER TABLESPACE users READ ONLY;

Tablespace altered.

SQL> EXIT
$
```

3. Use the Oracle Data Pump FULL TRANSPORTABLE feature to export the NONCDB data.

```
$ expdp \"sys@NONCDB as sysdba\" full=y dumpfile=noncdb.dmp
transportable=always
...
Password: password
...
```

```

. . exported "SYS"."SQLOBJ$_DATAPUMP" 0
KB      0 rows
. . exported "SYSTEM"."SCHEDULER_JOB_ARGS" 8.671
KB      4 rows
. . exported "SYSTEM"."SCHEDULER_PROGRAM_ARGS" 10.29
KB     23 rows
. . exported "WMSYS"."WM$EXP_MAP" 7.718
KB      3 rows
. . exported "WMSYS"."WM$METADATA_MAP" 0
KB      0 rows
Master table "SYS"."SYS_EXPORT_FULL_01" successfully
loaded/unloaded
*****
Dump file set for SYS.SYS_EXPORT_FULL_01 is:
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/log/noncdb.dmp
*****
Datafiles required for transportable tablespace USERS:
/u02/app/oracle/oradata/NONCDB/users01.dbf
Job "SYS"."SYS_EXPORT_FULL_01" successfully completed at Fri Jun
15 12:31:12 2018 elapsed 0 00:04:19
$
```

4. Create a directory in the target CDB for the new PDB.

```
$ mkdir /u02/app/oracle/oradata/ORCL/pdb_noncdb
$
```

5. Copy the export dump file and the datafiles of the USERS tablespace of NONCDB to the target directory of ORCL.

```
$ cp
/u01/app/oracle/product/12.2.0/dbhome_1/rdbms/log/noncdb.dmp
/u01/app/oracle/admin/ORCL/dpdump
$ cp /u02/app/oracle/oradata/NONCDB/users01.dbf
/u02/app/oracle/oradata/ORCL/pdb_noncdb
$
```

6. In ORCL, create the PDB\_NONCDB PDB to import the NONCDB data.

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

SQL> ALTER SESSION SET
DB_CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/pdb_noncdb';

Session altered.
```

```

SQL> CREATE PLUGGABLE DATABASE pdb_noncdb
      ADMIN USER pdb_admin IDENTIFIED BY password
      ROLES=(CONNECT)
      CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/pdb_noncdb';
2   3   4
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb_noncdb OPEN;

Pluggable database altered.

SQL>

```

7. Check that the USERS tablespace does not exist in PDB\_NONCDB before importing.

```

SQL> ALTER SESSION SET CONTAINER=pdb_noncdb;

Session altered.

SQL> SELECT tablespace_name FROM dba_tablespaces;

TABLESPACE_NAME
-----
SYSTEM
SYSAUX
UNDOTBS1
TEMP

SQL>

```

8. Create the data pump directory in PDB\_NONCDB.

```

SQL> CREATE DIRECTORY dp_dump AS
  '/u01/app/oracle/admin/ORCL/dpdump';

Directory created.

SQL> EXIT
$ 

```

9. Import the NONCDB data into PDB\_NONCDB.

```

$ impdp '\sys@pdb_noncdb as sysdba\'
      DIRECTORY=dp_dump DUMPFILE=noncdb.dmp
      TRANSPORT_DATAFILES='/u02/app/oracle/oradata/ORCL/pdb_noncdb/use
rs01.dbf'
...

```

```
Password: password
...
Processing object type
DATABASE_EXPORT/AUDIT_UNIFIED/AUDIT_POLICY_ENABLE
Processing object type
DATABASE_EXPORT/POST_SYSTEM_IMPCALLOUT/MARKER
Job "SYS"."SYS_IMPORT_FULL_01" completed with 5 error(s) at Fri
Jun 15 12:52:23 2018 elapsed 0 00:02:36
$
```

10. Verify that the HR.EMPLOYEES table is present in PDB\_NONCDB.

```
$ sqlplus hr@PDB_NONCDB
Enter password: password

SQL> SELECT count(*) FROM hr.employees;

COUNT (*)
-----
107

SQL> EXIT
$
```

11. Reclaim disk space by dropping the NONCDB non-CDB. Execute the \$HOME/labs/admin/drop\_NONCDB.sh shell script.

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

## Practice 12-2: Performing a Full Transportable Export/Import from a 12c PDB into an 18c PDB

### Overview

In this practice, you export an Oracle Database 12c PDB and import it into an Oracle Database 18c PDB by using the Oracle Data Pump FULL TRANSPORTABLE feature.

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs. The shell script drops all PDBs that may have been created and finally re-creates PDB1.

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

2. In `cdb12` and particularly in `PDB12`, set the `USERS` user-defined tablespace that stores the `HR.EMPLOYEES` table to **READ ONLY** before exporting. In case `cdb12` and `PDB12` are not open, start the instance up and open the PDB.

```
$ . oraenv
ORACLE_SID = [oracle] ? cdb12
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> STARTUP
ORACLE instance started.

Total System Global Area 788529152 bytes
Fixed Size                  8797728 bytes
Variable Size                318767584 bytes
Database Buffers             452984832 bytes
Redo Buffers                 7979008 bytes
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE pdb12 OPEN;

Pluggable database altered.

SQL> EXIT
$
```

```
$ sqlplus system@PDB12
Enter password: password

SQL> SELECT count(*) FROM hr.employees;

COUNT (*)
-----
107

SQL> ALTER TABLESPACE users READ ONLY;

Tablespace altered.

SQL> EXIT
$
```

3. Use Oracle Data Pump FULL TRANSPORTABLE feature to export the PDB12 data.

```
$ expdp \"sys@PDB12 as sysdba\" full=y dumpfile=PDB12.dmp
transportable=always
...
Password: password
...
. . exported "WMSYS"."WM$METADATA_MAP" 0
KB      0 rows
Master table "SYS"."SYS_EXPORT_FULL_01" successfully
loaded/unloaded
*****
Dump file set for SYS.SYS_EXPORT_FULL_01 is:
/u01/app/oracle/admin/cdb12/dpdump/6EAF37C8B1893F25E0535410ED0A1
9DF/PDB12.dmp
*****
Datafiles required for transportable tablespace USERS:
/u02/app/oracle/oradata/cdb12/pdb12/users01.dbf
Job "SYS"."SYS_EXPORT_FULL_01" successfully completed at Fri Jun
15 13:39:40 2018 elapsed 0 00:04:43
$
```

4. Create a directory in the target CDB for the new PDB.

```
$ mkdir /u02/app/oracle/oradata/ORCL/PDB12_IN_18
$
```

5. Copy the export dump file and the datafiles of the USERS tablespace of PDB12 of cdb12 to the target directory of ORCL.

```
$ cp
/u01/app/oracle/admin/cdb12/dpdump/6EAFC8B1893F25E0535410ED0A1
9DF/PDB12.dmp /u01/app/oracle/admin/ORCL/dpdump
$ cp /u02/app/oracle/oradata/cdb12/pdb12/users01.dbf
/u02/app/oracle/oradata/ORCL/PDB12_IN_18
$
```

6. In ORCL, create the PDB12\_IN\_18 PDB to import the PDB12 data.

```
$ . oraenv
ORACLE_SID = [cdb12] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA

SQL> ALTER SESSION SET
DB_CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/PDB12_IN_18';

Session altered.

SQL> CREATE PLUGGABLE DATABASE pdb12_in_18
      ADMIN USER pdb_admin IDENTIFIED BY password
      ROLES=(CONNECT)
CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/PDB12_IN_18';
2   3   4
Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE pdb12_in_18 OPEN;

Pluggable database altered.

SQL>
```

7. Check that the USERS tablespace does not exist in PDB12\_IN\_18 before importing.

```
SQL> ALTER SESSION SET CONTAINER=pdb12_in_18;

Session altered.

SQL> SELECT tablespace_name FROM dba_tablespaces;

TABLESPACE_NAME
-----
SYSTEM
SYSAUX
```

```
UNDOTBS1  
TEMP  
  
SQL>
```

8. Create the data pump directory in PDB12\_IN\_18.

```
SQL> CREATE DIRECTORY dp_dump AS  
'/u01/app/oracle/admin/ORCL/dpdump' ;
```

```
Directory created.
```

```
SQL> EXIT  
$
```

9. Import the PDB12 data into PDB12\_IN\_18.

```
$ impdp \'sys@PDB12_IN_18 as sysdba\' FULL=Y  
      DIRECTORY=dp_dump DUMPFILE=PDB12.dmp  
TRANSPORT_DATAFILES='/u02/app/oracle/oradata/ORCL/PDB12_IN_18/us  
ers01.dbf'  
...  
Password: password  
...  
Processing object type  
DATABASE_EXPORT/AUDIT_UNIFIED/AUDIT_POLICY_ENABLE  
Processing object type  
DATABASE_EXPORT/POST_SYSTEM_IMPCALLOUT/MARKER  
Job "SYS"."SYS_IMPORT_FULL_01" completed with 5 error(s) at Fri  
Jun 15 12:52:23 2018 elapsed 0 00:02:36  
$
```

10. Verify that the HR.EMPLOYEES table is present in PDB12\_IN\_18.

```
$ sqlplus hr@PDB12_IN_18  
Enter password: password  
  
SQL> SELECT count(*) FROM hr.employees;  
  
COUNT (*)  
-----  
107  
  
SQL> EXIT  
$
```

## **Practices for Lesson 13: Upgrade Methods**

## Practices for Lesson 13: Overview

---

### Practices Overview

During upgrade operations, you will:

- Upgrade the Oracle Database 12.2 PDB12 PDB from `cdb12` to Oracle Database 18c `PDB18_IN_ROOT` into `ORCL` as an application PDB of the `HR_ROOT` application container.
- Optionally upgrade the Oracle Database 12.2 `cdb12` to the Oracle Database 18c 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 12.2 and `ORACLE_SID` set to `cdb12` (*named session 12c*)
- Another one with `ORACLE_HOME` set to 18.1.0.0 and `ORACLE_SID` set to `ORCL` (*named session 18c*)

## Practice 13-1: Upgrading a 12.2 Regular PDB to an 18c Application PDB

---

### Overview

You will upgrade the Oracle Database 12.2 PDB12 PDB from `cdb12` to Oracle Database 18c `PDB18_IN_ROOT` into `ORCL` as an application PDB of the `HR_ROOT` application container.

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to clean up your CDB and PDBs in `ORCL`. The shell script drops all PDBs that may have been created and finally re-creates `PDB1`.

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

2. Open two terminal windows, one with Oracle Database 12.2 environment variables set and another with Oracle Database 12.2 environment variables set.
  - a. Before starting the practice, execute the `$HOME/labs/upgrade/glogin_13.sh` shell script. The script sets formatting for all columns selected in queries in both Oracle Database 12.2 and Oracle Database 18c environments.

```
$ $HOME/labs/upgrade/glogin_13.sh
$
```

- b. Open a terminal window and set title to 12.2 (*Terminal -> Set Title...*). This will be session 12c. If `cdb12` is not started, start it up.

```
$ . oraenv
ORACLE_SID = [oracle] ? cdb12
The Oracle base remains unchanged with value /u01/app/oracle
$ env|grep ORACLE
```

```
ORACLE_SID=cdb12
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/12.2.0/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> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database opened.

SQL> SHOW PDBS

CON_ID CON_NAME           OPEN MODE RESTRICTED
-----
2 PDB$SEED              READ ONLY NO
3 PDB12                  READ WRITE NO
SQL>

```

3. Connect to PDB12 and check the existence of the HR.EMPLOYEES table.

```

SQL> ALTER SESSION SET CONTAINER=PDB12;

Session altered.

SQL> SELECT count(*) FROM hr.employees;

COUNT(*)
-----
107

SQL> EXIT
$
```

4. Open another terminal window and set title to 18c (*Terminal -> Set Title...*). This will be **session 18c**. The \$HOME/labs/upgrade/setup\_hr\_app.sh shell script creates the HR\_ROOT application container and its SALES application PDB. If the instance is not started, restart it.

While the \$HOME/labs/upgrade/setup\_hr\_app.sh shell script is executing, proceed with step 5.

```

$ $HOME/labs/upgrade/glogin_13.sh
$
```

```

$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ env|grep ORACLE
```

```

ORACLE_SID=ORCL
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/18.1.0/dbhome_1
$ $HOME/labs/upgrade/setup_hr_app.sh
...
$
```

5. In session 12c, prepare PDB12 to be unplugged from cdb12 and upgraded in ORCL as PDB18\_IN\_ROOT.

- a. Execute the Pre-Upgrade Information Tool on the 12.2 PDB12 by executing the preupgrade.jar file.

```

$ cd /u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin
$ $ORACLE_HOME/jdk/bin/java -jar preupgrade.jar -c 'PDB12'
=====
PREUPGRADE SUMMARY
=====
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade.log
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups.sql
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.sql
```

Execute fixup scripts across the entire CDB:

**Before upgrade:**

1. Execute **preupgrade\_fixups** with the below command

```
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -
I$ORACLE_HOME/rdbms/admin $ORACLE_HOME/rdbms/admin/catcon.pl -l
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/ -b preup_cdb12
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups.sql
```

2. Review logs under

```
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/
```

**After the upgrade:**

1. Execute **postupgrade\_fixups** with the below command

```
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -
I$ORACLE_HOME/rdbms/admin $ORACLE_HOME/rdbms/admin/catcon.pl -l
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/ -b postup_cdb12
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.sql
```

```
2. Review logs under
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/
Preupgrade complete: 2018-06-15T14:28:00
$
```

*Q/ Did the Pre-Upgrade Information Tool work successfully?*

*A/ The default output is a directory. This directory is by default  
\$ORACLE\_BASE/cfgtoollogs/cdb12/preupgrade.*

Re-execute the same command to get the report on your terminal.

```
$ $ORACLE_HOME/jdk/bin/java -jar preupgrade.jar -c 'PDB12'
TERMINAL TEXT
Report generated by Oracle Database Pre-Upgrade Information Tool
Version
18.0.0.0.0 on 2018-06-15T14:29:46

Upgrade-To version: 18.0.0.0.0

=====
Status of the database prior to upgrade
=====

Database Name: CDB12
Container Name: PDB12
Container ID: 3
Version: 12.2.0.1.0
Compatible: 12.2.0
Blocksize: 8192
Platform: Linux x86 64-bit
Timezone File: 26
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 OLAP API	[to be upgraded]	VALID

**=====**  
**BEFORE UPGRADE**  
**=====****REQUIRED ACTIONS**  
**=====**

None

**RECOMMENDED ACTIONS**  
**=====**

1. (AUTOFIXUP) Gather stale data dictionary statistics prior to database

upgrade in off-peak time using:

```
EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;
```

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.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

2. (AUTOFIXUP) Gather statistics on fixed objects prior the upgrade.

None of the fixed object tables have had stats collected.

Gathering statistics on fixed objects, if none have been gathered yet, is recommended prior to upgrading.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

INFORMATION ONLY

=====

3. To help you keep track of your tablespace allocations, the following

AUTOEXTEND tablespaces are expected to successfully EXTEND during the

upgrade process.

Tablespace	Size	Min Size For Upgrade
SYSAUX	370 MB	587 MB
SYSTEM	250 MB	690 MB
TEMP	64 MB	150 MB
UNDOTBS1	100 MB	439 MB

Minimum tablespace sizes for upgrade are estimates.

4. No action needed.

Using default parallel upgrade options, this CDB with 1 PDBs will first

upgrade the CDB\$ROOT, and then upgrade at most 1 PDBs at a time using 2

parallel processes per PDB.

The number of PDBs upgraded in parallel and the number of parallel

processes per PDB can be adjusted as described in Database Upgrade Guide.

ORACLE GENERATED FIXUP SCRIPT

=====

All of the issues in database CDB12 container PDB12 which are identified above as BEFORE UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade\_fixups.sql

=====

**AFTER UPGRADE**

=====

REQUIRED ACTIONS

=====

None

RECOMMENDED ACTIONS

=====

5. Upgrade the database time zone file using the DBMS\_DST package.

The database is using time zone file version 26 and the target 18.0.0.0.0

release ships with time zone file version 31.

Oracle recommends upgrading to the desired (latest) version of the time

zone file. For more information, refer to "Upgrading the Time Zone File

and Timestamp with Time Zone Data" in the 18.0.0.0.0 Oracle Database

Globalization Support Guide.

6. (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.

7. Gather statistics on fixed objects after the upgrade and when there is a representative workload on the system using the command:

```
EXECUTE DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

This recommendation is given for all preupgrade runs.

Fixed object statistics provide essential information to the Oracle

optimizer to help it find efficient SQL execution plans. Those

statistics are specific to the Oracle Database release that generates

them, and can be stale upon database upgrade.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

ORACLE GENERATED FIXUP SCRIPT

=====

All of the issues in database CDB12 container PDB12 which are identified above as AFTER UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

```
SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.sql
```

=====

```
PREUPGRADE SUMMARY
```

=====

```
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade.log
```

```
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups.sql
```

```
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.sql
```

Execute fixup scripts across the entire CDB:

Before upgrade:

1. Execute preupgrade fixups with the below command

```
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -  
I$ORACLE_HOME/rdbms/admin $ORACLE_HOME/rdbms/admin/catcon.pl -l  
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/ -b preup_cdb12  
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups.sql
```

2. Review logs under

```
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/
```

After the upgrade:

1. Execute postupgrade fixups with the below command

```
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -  
I$ORACLE_HOME/rdbms/admin $ORACLE_HOME/rdbms/admin/catcon.pl -l  
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/ -b postup_cdb12  
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.sql
```

2. Review logs under

```
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/
```

Preupgrade complete: 2018-06-15T14:29:46

\$

*Q2/ Which containers in the CDB does the Pre-Upgrade Information Tool provide actions for?*

**A2/ The Pre-Upgrade Information Tool provides actions for PDB12 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 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 PDB12.

```
$ cd /u01/app/oracle/cfgtoollogs/cdb12/preupgrade/  
$ ls *PDB12.sql  
postupgrade_fixups_PDB12.sql  preupgrade_fixups_PDB12.sql  
$
```

**Q/ Is there a script created to fix the recommendations?**

**A/ Yes. There is the `preupgrade_fixups_PDB12.sql` SQL script.**

**Note:** Read the script before executing it.

```
$ sqlplus / AS SYSDBA  
  
SQL> ALTER SESSION SET CONTAINER = pdb12;  
  
Session altered.  
  
SQL>  
@$ORACLE_BASE/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups_PDB12.sql  
Executing Oracle PRE-Upgrade Fixup Script
```

```
Auto-Generated by:          Oracle Preupgrade Script
                        Version: 18.0.0.0.0 Build: 1
Generated on:            2018-06-15 14:29:43

For Source Database:      CDB12
Source Database Version:  12.2.0.1.0
For Upgrade to Version:  18.0.0.0.0

Executing in container:   PDB12

Preup                                Preupgrade
Action                               Issue Is
Number    Preupgrade Check Name     Remedied   Further DBA Action
-----  -----  -----  -----
-----  -----  -----  -----
1.  dictionary_stats                YES        None.
2.  pre_fixed_objects              YES        None.
3.  tablespaces_info only.         NO         Informational
                                         Further action is
optional.
4.  cycle_number only.             NO         Informational
                                         Further action is
optional.
```

The fixup scripts have been run and resolved what they can. However, there are still issues originally identified by the preupgrade that have not been remedied and are still present in the database.

Depending on the severity of the specific issue, and the nature of the issue itself, that could mean that your database is not ready for upgrade. To resolve the outstanding issues, start by reviewing the preupgrade\_fixups.sql and searching it for the name of the failed CHECK NAME or Preupgrade Action Number listed above.

There you will find the original corresponding diagnostic message from the preupgrade which explains in more detail what still needs to be done.

PL/SQL procedure successfully completed.

```
SQL> EXIT
$
```

*Q2/ Did the script fix all required and recommended actions?*

**A2/ Yes, it did. But some components like APEX could be installed and would require a manual operation to be completed if the 18c CDB did not hold APEX at the CDB root level. There would be several solutions.**

- In the target 18c 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 in common.
- Alternatively, which is more cumbersome, use the command-line utility APEX Export to export all workspaces, applications, scripts, users, and artifacts and then plug in the PDB, remove APEX from the PDB, install APEX locally into the PDB, and finally import all of the artifacts that were exported.

- c. Still in session 12c, back up PDB12.

```
$ rman target /  
  
Recovery Manager: Release 12.2.0.1.0 - Production on Fri Jun 15  
14:42:08 2018  
  
connected to target database: CDB12 (DBID=1996891072)  
  
RMAN> BACKUP PLUGGABLE DATABASE pdb12 PLUS ARCHIVELOG;  
  
Starting backup at 15-JUN-18  
current log archived  
using target database control file instead of recovery catalog  
allocated channel: ORA_DISK_1  
channel ORA_DISK_1: SID=18 device type=DISK  
channel ORA_DISK_1: starting archived log backup set  
channel ORA_DISK_1: specifying archived log(s) in backup set  
input archived log thread=1 sequence=2 RECID=3 STAMP=978874284  
input archived log thread=1 sequence=3 RECID=1 STAMP=978874283  
input archived log thread=1 sequence=4 RECID=2 STAMP=978874283  
input archived log thread=1 sequence=5 RECID=4 STAMP=978877778  
input archived log thread=1 sequence=6 RECID=5 STAMP=978878129  
input archived log thread=1 sequence=7 RECID=6 STAMP=978878553  
channel ORA_DISK_1: starting piece 1 at 15-JUN-18  
channel ORA_DISK_1: finished piece 1 at 15-JUN-18  
piece  
handle=/u03/app/oracle/fast_recovery_area/CDB12/backupset/2018_0  
6_15/o1_mf_annnn_TAG20180615T144234_f17n2v3q_.bkp  
tag=TAG20180615T144234 comment=NONE  
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:03
```

```
Finished backup at 15-JUN-18

Starting backup at 15-JUN-18
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00010
name=/u02/app/oracle/oradata/cdb12/pdb12/sysaux01.dbf
input datafile file number=00009
name=/u02/app/oracle/oradata/cdb12/pdb12/system01.dbf
input datafile file number=00011
name=/u02/app/oracle/oradata/cdb12/pdb12/undotbs01.dbf
input datafile file number=00012
name=/u02/app/oracle/oradata/cdb12/pdb12/users01.dbf
channel ORA_DISK_1: starting piece 1 at 15-JUN-18
channel ORA_DISK_1: finished piece 1 at 15-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/CDB12/6EAF37C8B1893F25
E0535410ED0A19DF/backupset/2018_06_15/o1_mf_nnndf_TAG20180615T14
4238_f17n2y10_.bkp tag=TAG20180615T144238 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 15-JUN-18

Starting backup at 15-JUN-18
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=8 RECID=7 STAMP=978878573
channel ORA_DISK_1: starting piece 1 at 15-JUN-18
channel ORA_DISK_1: finished piece 1 at 15-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/CDB12/backupset/2018_0
6_15/o1_mf_annnn_TAG20180615T144254_f17n3gf4_.bkp
tag=TAG20180615T144254 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 15-JUN-18

Starting Control File and SPFILE Autobackup at 15-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/CDB12/autobackup/2018_
06_15/o1_mf_s_978878575_f17n3jhj_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 15-JUN-18

RMAN> EXIT
$
```

6. Unplug PDB12.

```
$ rm /tmp/xmlfilePDB12.xml
rm: cannot remove `/tmp/xmlfilePDB12.xml': No such file or
directory
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE pdb12 CLOSE;

Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb12
      UNPLUG INTO '/tmp/xmlfilePDB12.xml';
2
Pluggable database altered.

SQL> SELECT pdb_name, status FROM cdb_pdbs
      WHERE pdb_name = 'PDB12';
2
PDB_NAME      STATUS
-----
PDB12          UNPLUGGED

SQL> EXIT
$
```

7. Plug PDB12 into ORCL.

- a. Before performing the plugging operation, you can optionally check whether the unplugged PDB12 is compatible with ORCL. Execute the DBMS\_PDB.CHECK\_PLUG\_COMPATIBILITY function in **session 18c**:

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle $ 
sqlplus / AS SYSDBA
Connected.
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
      compat BOOLEAN := FALSE;
      BEGIN
      compat := DBMS_PDB.CHECK_PLUG_COMPATIBILITY(pdb_descr_file =>
'/tmp/xmlfilePDB12.xml', pdb_name => 'pdb12');
      if compat then
      DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? YES');
      else DBMS_OUTPUT.PUT_LINE('Is pluggable compatible? NO');
      END IF;
      END;
/
Is pluggable compatible? YES
```

```
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_inViolations
      WHERE name = 'PDB12';
2
MESSAGE
-----
ACTION
-----
PDB's version does not match CDB's version: PDB's version
12.2.0.1.0. CDB's version 18.0.0.0.0.
Either upgrade the PDB or reload the components in the PDB.

CDB parameter compatible mismatch: Previous '12.2.0' Current
'18.0.0'
Please check the parameter in the current CDB

CDB parameter pga_aggregate_target mismatch: Previous 250M
Current 450M
Please check the parameter in the current CDB

SQL>
```

*Q/ Which of the 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. Another message refers to the initialization parameter value that can be updated after the upgrade. If there were an APEX version of the plugged PDB error, lower than the APEX version recently installed in the target CDB, you would have to upgrade the PDB APEX version to the one of the target CDB.**

- c. Plug PDB12 into ORCL as the regular PDB18\_IN\_ROOT PDB. Copy the original files into the /u02/app/oracle/oradata/ORCL/PDB18\_IN\_ROOT directory.

```
SQL> !mkdir /u02/app/oracle/oradata/ORCL/PDB18_IN_ROOT
```

```
SQL> ALTER SESSION SET DB_CREATE_FILE_DEST =
      '/u02/app/oracle/oradata/ORCL/PDB18_IN_ROOT';
2
Session altered.

SQL> CREATE PLUGGABLE DATABASE pdb18_in_root
      USING '/tmp/xmlfilePDB12.xml'
CREATE_FILE_DEST='/u02/app/oracle/oradata/ORCL/PDB18_IN_ROOT'
      COPY;
2      3      4
Pluggable database created.

SQL>
```

8. Open PDB18\_IN\_ROOT in ORCL in UPGRADE mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb18_in_root OPEN UPGRADE;

Pluggable database altered.

SQL> SHOW PDBS
CON_ID CON_NAME          OPEN MODE RESTRICTED
-----
2 PDB$SEED              READ ONLY NO
3 HR_ROOT                READ WRITE NO
4 SALES                  READ WRITE NO
5 PDB18_IN_ROOT          MIGRATE YES
6 PDB1                   READ WRITE NO

SQL>
```

9. Set the archive log recovery destination size to a large value.

```
SQL> ALTER SYSTEM SET db_recovery_file_dest_size='40G'
      SCOPE = BOTH;
2
System altered.

SQL> EXIT
$
```

10. Upgrade PDB18\_IN\_ROOT in ORCL. 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 'PDB18_IN_ROOT'
catupgrd.sql

Argument list for [catctl.pl]
Run in          c = PDB18_IN_ROOT
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: [18.0.0.0.0]
    STATUS: [Production]
    BUILD: [RDBMS_18.1.0.0.0_LINUX.X64_180103.1]

/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/18.1.0/dbhome_1]
/u01/app/oracle/product/18.1.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/18.1.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/18.1.0/dbhome_1]
```

```
Analyzing file
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/catupgrql

Log file directory = [/tmp/cfgtoollogs/upgrade20180615145336]

catcon::set_log_file_base_path: ALL catcon-related output will
be written
[/tmp/cfgtoollogs/upgrade20180615145336/catupgrd_catcon_19829.ls
t]

catcon::set_log_file_base_path: catcon: See
[/tmp/cfgtoollogs/upgrade20180615145336/catupgrd*.log] files for
output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/tmp/cfgtoollogs/upgrade20180615145336/catupgrd_*.lst] files for
spool files, if any

Number of Cpus          = 1
Database Name          = ORCL
DataBase Version        = 18.0.0.0.0
catcon::set_log_file_base_path: ALL catcon-related output will
be written
[/u01/app/oracle/product/18.1.0/dbhome_1/cfgtoollogs/ORCL/upgrad
e20180615145339/catupgrd_catcon_19829.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/cfgtoollogs/ORCL/upgrade20
180615145339/catupgrd*.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/cfgtoollogs/ORCL/upgrade20
180615145339/catupgrd_*.lst] files for sl files, if any

Log file directory =
[/u01/app/oracle/product/18.1.0/dbhome_1/cfgtoollogs/L/upgrade20
180615145339]

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:[PDB18_IN_ROOT]
CDB$ROOT  Open Mode = [OPEN]

Start processing of PDB18_IN_ROOT
[/u01/app/oracle/product/18.1.0/dbhome_1/perl/bin/perl catctl.pl
-c 'PDB18_ROOT' -I -i pdb18_in_root -n 2 -l
/u01/app/oracle/product/18.1.0/dbhome_1gtoollogs/ORCL/upgrade201
80615145339 catupgrd.sql]

Argument list for [catctl.pl]
Run in          c = PDB18_IN_ROOT
Do not run in   C = 0
Input Directory d = 0
Echo OFF        e = 1
Simulate        E = 0
Forced cleanup  F = 0
Log Id          i = pdb18_in_root
Child Process   I = 1
Log Dir         l =
/u01/app/oracle/product/18.1.0/dbhome_1/cfgtool1/ORCL/upgrade201
80615145339
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: [18.0.0.0.0]
STATUS: [Production]
BUILD: [RDBMS_18.1.0.0.0_LINUX.X64_180103.1]

/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/18.1.0/dbhome_1]
```

```

/u01/app/oracle/product/18.1.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/18.1.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/18.1.0/dbhome_1]

Analyzing file
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/catupgrql

Log file directory =
[/u01/app/oracle/product/18.1.0/dbhome_1/cfgtoollogs/L/upgrade20
180615145339]

catcon::set_log_file_base_path: ALL catcon-related output will
be written
[/u01/app/oracle/product/18.1.0/dbhome_1/cfgtoollogs/ORCL/upgrad
e20180615139/catupgrdpdb18_in_root_catcon_20014.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.bhome_1/cfgtoollogs/ORCL/upgrade20
180615145339/catupgrdpdb18_in_root*.log]les for output generated
by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.bhome_1/cfgtoollogs/ORCL/upgrade20
180615145339/catupgrdpdb18_in_root*.lstiles for spool files, if
any

Number of Cpus      = 1
Database Name       = ORCL
DataBase Version    = 18.0.0.0.0
Generated PDB Inclusion:[PDB18_IN_ROOT]
CDB$ROOT Open Mode = [OPEN]
Components in [PDB18_IN_ROOT]
   Installed [APS CATALOG CATJAVA CATPROC CONTEXT DV JAVAVM OLS
ORDIM OWM SDO XDB XML XOO]
   Not Installed [APEX EM MGW ODM RAC WK]

-----
Phases [0-108]          Start Time:[2018_06_15 14:53:47]
Container Lists Inclusion:[PDB18_IN_ROOT] Exclusion:[NONE]
-----
***** Executing Change Scripts *****
Serial Phase #:0      [PDB18_IN_ROOT] Files:1
***** Catalog Core SQL *****
Serial Phase #:1      [PDB18_IN_ROOT] Files:5      Time: 49s

```

```

Restart Phase #:2      [PDB18_IN_ROOT] Files:1      Time: 0s
***** Catalog Tables and Views *****
Parallel Phase #:3      [PDB18_IN_ROOT] Files:19     Time: 23s
Restart Phase #:4      [PDB18_IN_ROOT] Files:1      Time: 1s
***** Catalog Final Scripts *****
Serial   Phase #:5      [PDB18_IN_ROOT] Files:7      Time: 18s
***** Catproc Start *****
Serial   Phase #:6      [PDB18_IN_ROOT] Files:1      Time: 11s
***** Catproc Types *****
Serial   Phase #:7      [PDB18_IN_ROOT] Files:2      Time: 11s
Restart Phase #:8      [PDB18_IN_ROOT] Files:1      Time: 0s
***** Catproc Tables *****
Parallel Phase #:9      [PDB18_IN_ROOT] Files:66     Time: 30s
Restart Phase #:10     [PDB18_IN_ROOT] Files:1      Time: 0s
***** Catproc Package Specs *****
Serial   Phase #:11     [PDB18_IN_ROOT] Files:1      Time: 75s
Restart Phase #:12     [PDB18_IN_ROOT] Files:1      Time: 1s
***** Catproc Procedures *****
Parallel Phase #:13     [PDB18_IN_ROOT] Files:94     Time: 8s
Restart Phase #:14     [PDB18_IN_ROOT] Files:1      Time: 0s
Parallel Phase #:15     [PDB18_IN_ROOT] Files:117    Time: 13s
Restart Phase #:16     [PDB18_IN_ROOT] Files:1      Time: 0s
Serial   Phase #:17     [PDB18_IN_ROOT] Files:17     Time: 3s
Restart Phase #:18     [PDB18_IN_ROOT] Files:1      Time: 0s
***** Catproc Views *****
Parallel Phase #:19     [PDB18_IN_ROOT] Files:32     Time: 19s
Restart Phase #:20     [PDB18_IN_ROOT] Files:1      Time: 0s
Serial   Phase #:21     [PDB18_IN_ROOT] Files:3      Time: 10s
Restart Phase #:22     [PDB18_IN_ROOT] Files:1      Time: 0s
Parallel Phase #:23     [PDB18_IN_ROOT] Files:24     Time: 119s
Restart Phase #:24     [PDB18_IN_ROOT] Files:1      Time: 0s
Parallel Phase #:25     [PDB18_IN_ROOT] Files:12     Time: 74s
Restart Phase #:26     [PDB18_IN_ROOT] Files:1      Time: 0s
Serial   Phase #:27     [PDB18_IN_ROOT] Files:1      Time: 0s
Serial   Phase #:28     [PDB18_IN_ROOT] Files:3      Time: 4s
Serial   Phase #:29     [PDB18_IN_ROOT] Files:1      Time: 0s
Restart Phase #:30     [PDB18_IN_ROOT] Files:1      Time: 0s
***** Catproc CDB Views *****
Serial   Phase #:31     [PDB18_IN_ROOT] Files:1      Time: 1s
Restart Phase #:32     [PDB18_IN_ROOT] Files:1      Time: 0s
Serial   Phase #:34     [PDB18_IN_ROOT] Files:1      Time: 0s
***** Catproc PLBs *****

```

```

Serial    Phase #:35      [PDB18_IN_ROOT] Files:288 Time: 21s
Serial    Phase #:36      [PDB18_IN_ROOT] Files:1   Time: 0s
Restart   Phase #:37      [PDB18_IN_ROOT] Files:1   Time: 0s
Serial    Phase #:38      [PDB18_IN_ROOT] Files:2   Time: 3s
Restart   Phase #:39      [PDB18_IN_ROOT] Files:1   Time: 0s
***** Catproc DataPump *****
Serial    Phase #:40      [PDB18_IN_ROOT] Files:3   Time: 51s
Restart   Phase #:41      [PDB18_IN_ROOT] Files:1   Time: 1s
***** Catproc SQL *****
Parallel  Phase #:42      [PDB18_IN_ROOT] Files:13  Time: 75s
Restart   Phase #:43      [PDB18_IN_ROOT] Files:1   Time: 0s
Parallel  Phase #:44      [PDB18_IN_ROOT] Files:11  Time: 5s
Restart   Phase #:45      [PDB18_IN_ROOT] Files:1   Time: 0s
Parallel  Phase #:46      [PDB18_IN_ROOT] Files:3   Time: 2s
Restart   Phase #:47      [PDB18_IN_ROOT] Files:1   Time: 1s
***** Final Catproc scripts *****
Serial    Phase #:48      [PDB18_IN_ROOT] Files:1   Time: 6s
...
*** End PDB Application Upgrade Post-Shutdown ***
Serial    Phase #:106     [PDB18_IN_ROOT] Files:1   Time: 1s
Serial    Phase #:107     [PDB18_IN_ROOT] Files:1   Time: 1s
Serial    Phase #:108     [PDB18_IN_ROOT] Files:1   Time: 0s

```

-----  
Phases [0-108] End Time:[2018\_06\_15 15:16:44]  
Container Lists Inclusion:[PDB18\_IN\_ROOT] Exclusion:[NONE]  
-----

Grand Total Time: 1380s [PDB18\_IN\_ROOT]

LOG FILES:  
(/u01/app/oracle/product/18.1.0/dbhome\_1/cfgtoollogs/ORCL/upgrad  
e20180615145339/catupgrdpdb18\_in\_root\*.log)

Upgrade Summary Report Located in:  
/u01/app/oracle/product/18.1.0/dbhome\_1/cfgtoollogs/ORCL/upgrade  
20180615145339/upg\_summary.log

Total Upgrade Time: [0d:0h:23m:0s]

Time: 1392s For PDB(s)

Grand Total Time: 1392s

```
LOG FILES:
(/u01/app/oracle/product/18.1.0/dbhome_1/cfgtoollogs/ORCL/upgrad
e20180615145339/catupgrd*.log)
```

```
Grand Total Upgrade Time: [0d:0h:23m:12s]
```

```
$
```

11. Finally execute the `postupgrade_fixups_PDB12.sql` and `utlrp.sql` scripts in the upgraded PDB to complete the upgrade step.

- a. First open the upgraded PDB.

```
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE pdb18_in_root OPEN;

Pluggable database altered.

SQL> EXIT
$
```

- b. Execute the postupgrade scripts, namely,

```
$ORACLE_BASE/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups_PDB
12.sql and $ORACLE_HOME/rdbms/admin/utlrp.sql.
```

```
$ cd $ORACLE_HOME/rdbms/admin
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c PDB18_IN_ROOT -b
postupgrade
$ORACLE_BASE/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups_PDB
12.sql

catcon::set_log_file_base_path: ALL catcon-related output will
be written to
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
_catcon_25880.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/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 PDB18_IN_ROOT -b
postupgrade $ORACLE_HOME/rdbms/admin/utlrp.sql
catcon::set_log_file_base_path: ALL catcon-related output will
be written to
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
_catcon_25958.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.lst] files for spool files, if any

catcon.pl: completed successfully
$
```

*Q/ How do you get details about issues that were fixed?*

*A/ In the output above, you know the directory where the output log files are generated and their names,*

*/u01/app/oracle/product/18.1.0/dbhome\_1/rdbms/admin/postupgrade\*.log. Read the*  
*/u01/app/oracle/product/18.1.0/dbhome\_1/rdbms/admin/postupgrade0.log file.*

```
$ more
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade0
.log
```

```
...
SQL>
NOW_CONNECTED_TO
-----
===== Current Container = PDB18_IN_ROOT Id = 5 =====

SQL>    2
CATCONSECTION
-----
===== CATCON EXEC IN CONTAINERS =====

SQL>
BEGIN_RUNNING
-----
```

```
=====
@/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/utlrp.sql
Container:PDB18_IN_ROOT Id:5 18-06-15 03:18:01 Proc:0 =====

SQL>
BEGIN_RUNNING
-----
=====

@/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/utlrp.sql
Container:PDB18_IN_ROOT Id:5 18-06-15 03:18:01 Proc:0 =====

...
OBJECTS WITH ERRORS
-----
0
ERRORS DURING RECOMPILATION
-----
0
SQL> ===== PROCESS ENDED =====
SQL> ===== Process Terminated by catcon =====
SQL> Disconnected from Oracle Database 18c Enterprise Edition
Release 18.0
.0.0.0 - Production
Version 18.1.0.0.0
$
```

12. In session 12c, now that the PDB is successfully upgraded to 18c, you drop the source 12c PDB. If it had not been successfully upgraded, it was still possible to plug the source PDB back to the source CDB.

```
$ . oraenv
ORACLE_SID = [oracle] ? cdb12
The Oracle base remains unchanged with value /u01/app/oracle $  

sqlplus / AS SYSDBA

SQL> DROP PLUGGABLE DATABASE pdb12 INCLUDING DATAFILES;

Pluggable database dropped.

SQL> EXIT
$
```

13. In session 18c, in the upgraded PDB18\_IN\_ROOT, check that the HR\_EMPLOYEES table exists.

```
$ sqlplus sys@PDB18_IN_ROOT AS SYSDBA
```

```
SQL> SELECT count(*) FROM hr.employees;

  COUNT (*)
-----
      107

SQL>
```

14. Your final goal was to upgrade the regular PDB18\_IN\_ROOT as the RESEARCH application PDB in the HR\_ROOT application container in ORCL.

In session 18c, use script\_pdb\_convert.sql to unplug PDB18\_IN\_ROOT and plug it as RESEARCH in the HR\_ROOT application container in ORCL. The script uses the exact same process as the one executed in practice 4-3 “Converting a Regular PDB to an Application PDB.”

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> @$HOME/labs/upgrade/script_pdb_convert
...
Pluggable database dropped.

SQL> CONNECT hr_mgr@RESEARCH
Enter password: password
Connected.
SQL> SELECT count(*) FROM hr_mgr.mgr_tab;

  COUNT (*)
-----
      0

1 row selected.

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.**

15. Remove all preupgrade and postupgrade scripts and log files.

```
$ cd /u01/app/oracle/cfgtoollogs/cdb12  
$ rm -rf preupgrade  
$
```

## Practice 13-2: Plugging Remote PDBs Through XTTs

---

### Overview

In this practice, you will perform a cross-platform PDB transport because you have to transport PDB1 from a CDB on a Linux platform to a CDB on a Solaris platform. The PDB transport from one platform to another 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, execute the

```
/home/oracle/labs/upgrade/cleanup_PDBs.sh shell script to clean up your CDB and PDBs in ORCL. The shell script drops all PDBs that may have been created and finally re-creates PDB1. PDB1 holds the HR.EMPLOYEES table.
```

```
$ $HOME/labs/upgrade/cleanup_PDBs.sh
...
$
```

2. Before backing up the PDB, ensure the prerequisites are satisfied.

- a. The database compatibility should be equal or greater than 18.0

```
$ sqlplus / AS SYSDBA

SQL> SHOW PARAMETER COMPATIBLE

NAME                      TYPE        VALUE
-----
compatible                string      18.0.0
noncdb_compatible         boolean    FALSE
SQL>
```

- b. The PDB must be closed before transportation starts.

```
SQL> ALTER PLUGGABLE DATABASE pdb1 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 endianness.

```
$ rman target /
connected to target database: ORCL (DBID=1434391901)

RMAN> BACKUP TO PLATFORM 'Solaris[tm] OE (64-bit)'
      UNPLUG INTO '/tmp/pdb1.xml'
      PLUGGABLE DATABASE pdb1
      FORMAT
      '/home/oracle/backup/ORCL/transportssolaris_%U';
2> 3> 4>

Starting backup at 15-JUN-18
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

RMAN> EXIT
$
```

*Q/ Why does it fail?*

**A/ 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?*

```
$ rm /tmp/pdb1.xml
$ rman target /

RMAN> BACKUP TO PLATFORM 'Solaris Operating System (x86-64)'
      UNPLUG INTO '/tmp/pdb1.xml'
      PLUGGABLE DATABASE pdb1
      FORMAT
      '/home/oracle/backup/ORCL/transportsolaris_%U';

2> 3> 4>
Starting backup at 15-JUN-18
using channel ORA_DISK_1
running UNPLUG on the specified pluggable database: PDB1
UNPLUG file path : /tmp/pdb1.xml
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00153
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB128EFC3FA68E7E053
5410ED0A945B/datafile/o1_mf_sysaux_f17r048z_.dbf
input datafile file number=00152
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB128EFC3FA68E7E053
5410ED0A945B/datafile/o1_mf_system_f17r048s_.dbf
input datafile file number=00154
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB128EFC3FA68E7E053
5410ED0A945B/datafile/o1_mf_undotbs1_f17r0490_.dbf
channel ORA_DISK_1: starting piece 1 at 15-JUN-18
channel ORA_DISK_1: finished piece 1 at 15-JUN-18
piece
handle=/home/oracle/backup/ORCL/transportsolaris_1tt5h6j6_1_1
tag=TAG20180615T163339 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 15-JUN-18

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 endianness like the source host.

*Q/ If you immediately executed BACKUP UNPLUG INTO /tmp/pdb1.xml using the xml file recently created for Solaris, the backup would fail with the following error ORA-65343: cannot unplug pluggable database PDB1.*

**A/ The PDB is considered unplugged by the first BACKUP UNPLUG INTO that succeeded.**

```
$ $HOME/labs/upgrade/cleanup_PDBs.sh
...
$ sqlplus / AS SYSDBA

SQL> ALTER PLUGGABLE DATABASE pdb1 CLOSE;

Pluggable database altered.

SQL> EXIT
$ rm /tmp/pdb1.xml
$ 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/pdb1.xml'
      PLUGGABLE DATABASE pdb1
      FORMAT '/home/oracle/backup/ORCL/transport_%U';
2> 3> 4>
Starting backup at 15-JUN-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=279 device type=DISK
running UNPLUG on the specified pluggable database: PDB1
UNPLUG file path : /tmp/pdb1.xml
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00156
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB1CF70F5106D69E053
5410ED0A44AD/datafile/o1_mf_sysaux_f17tqfbp_.dbf
```

```

input datafile file number=00155
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB1CF70F5106D69E053
5410ED0A44AD/datafile/o1_mf_system_f17tqfb0_.dbf
input datafile file number=00157
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB1CF70F5106D69E053
5410ED0A44AD/datafile/o1_mf_undotbs1_f17tqfbq_.dbf
channel ORA_DISK_1: starting piece 1 at 15-JUN-18
channel ORA_DISK_1: finished piece 1 at 15-JUN-18
piece handle=/home/oracle/backup/ORCL/transport_1ut5h6q3_1_1
tag=TAG20180615T163720 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 15-JUN-18

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 on 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 backup set, and xml file.

```

$ sqlplus / AS SYSDBA

SQL> DROP PLUGGABLE DATABASE pdb1 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 /u02/app/oracle/oradata/CDB18/pdb18_in_cdb18
$
```

- 2) Restore the converted data files.

```

$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$
$ rman target /
$
```

```

connected to target database: CDB18 (DBID=1938746784)

RMAN> ALTER SYSTEM SET
DB_CREATE_FILE_DEST='/u02/app/oracle/oradata/CDB18/pdb18_in_cdb18';'

Statement processed

RMAN> RESTORE USING '/tmp/pdb1.xml'
      FOREIGN pluggable database pdb18_in_cdb18 TO NEW
      FROM BACKUPSET
      '/home/oracle/backup/ORCL/transport_1ut5h6q3_1_1'
2> 3> 4>
Starting restore at 15-JUN-18
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=288 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_1ut5h6q3_1_1
channel ORA_DISK_1: restoring foreign file 156 to
/u02/app/oracle/oradata/CDB18/pdb18_in_cdb18/CDB18/datafile/o1_m
f_sysaux_f17v28j3_.dbf
channel ORA_DISK_1: restoring foreign file 155 to
/u02/app/oracle/oradata/CDB18/pdb18_in_cdb18/CDB18/datafile/o1_m
f_system_f17v28jy_.dbf
channel ORA_DISK_1: restoring foreign file 157 to
/u02/app/oracle/oradata/CDB18/pdb18_in_cdb18/CDB18/datafile/o1_m
f_undotbs1_f17v28k2_.dbf
channel ORA_DISK_1: foreign piece
handle=/home/oracle/backup/ORCL/transport_1ut5h6q3_1_1
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:08
channel ORA_DISK_1: plugging file 155 for
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB1CF70F5106D69E0535410E
D0A44AD/datafile/o1_mf_system_f17tqfb0_.dbf
channel ORA_DISK_1: plugging file 156 for
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB1CF70F5106D69E0535410E
D0A44AD/datafile/o1_mf_sysaux_f17tqfbp_.dbf
channel ORA_DISK_1: plugging file 157 for
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB1CF70F5106D69E0535410E
D0A44AD/datafile/o1_mf_undotbs1_f17tqfbq_.dbf

```

```
channel ORA_DISK_1: plugging file 3 for
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB1CF70F5106D69E0535410E
D0A44AD/datafile/o1_mf_temp_f17tqfbr_.dbf
```

```
Performing import of metadata...
Finished restore at 15-JUN-18
```

```
RMAN>
```

- d. Open the transported PDB.

```
RMAN> ALTER PLUGGABLE DATABASE pdb18_in_cdb18 OPEN;
```

```
Statement processed
```

```
RMAN> EXIT
```

```
$
```

- e. Display the HR.EMPLOYEES data.

```
$ sqlplus hr@PDB18_IN_CDB18
Enter password: password
Connected
SQL> SELECT count(*) FROM hr.employees;
```

```
COUNT (*)
```

```
-----
107
```

```
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. First re-create PDB1 in ORCL and drop the transported PDB in CDB18.

```
$ $HOME/labs/upgrade/cleanup_all_PDBs.sh
...
$
```

```
$ rm /home/oracle/backup/ORCL/transport*
$ rm /tmp/pdb1.xml
$ . oraenv
ORACLE_SID = [CDB18] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ rman target /
```

```
connected to target database: ORCL (DBID=1506381859)

RMAN> ALTER PLUGGABLE DATABASE pdb1 CLOSE;

Statement processed

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> BACKUP FOR TRANSPORT UNPLUG INTO '/tmp/pdb1.xml'
      PLUGGABLE DATABASE pdb1
      FORMAT '/home/oracle/backup/ORCL/transport_%U';

2> 3> 4>
Starting backup at 15-JUN-18
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=298 device type=DISK
running UNPLUG on the specified pluggable database: PDB1
UNPLUG file path : /tmp/pdb1.xml
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00168
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB24D9378FB7214E053
5410ED0A6F2F/datafile/o1_mf_sysaux_f17wskhh_.dbf
input datafile file number=00167
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB24D9378FB7214E053
5410ED0A6F2F/datafile/o1_mf_system_f17wskhf_.dbf
input datafile file number=00169
name=/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB24D9378FB7214E053
5410ED0A6F2F/datafile/o1_mf_undotbs1_f17wskhj_.dbf
channel ORA_DISK_1: starting piece 1 at 15-JUN-18
channel ORA_DISK_1: finished piece 1 at 15-JUN-18
piece handle=/home/oracle/backup/ORCL/transport_23t5h8r0_1_1
tag=TAG20180615T171156 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 15-JUN-18

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 backup set, and xml file.

```
RMAN> DROP PLUGGABLE DATABASE pdb1 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 /u02/app/oracle/oradata/CDB18/PDB18_IN_CDB18
$ mkdir /u02/app/oracle/oradata/CDB18/PDB18_IN_CDB18
$
```

- 2) Restore the converted data files into CDB18.

```
$ ls /home/oracle/backup/ORCL/transport*
/home/oracle/backup/ORCL/transport_23t5h8r0_1_1
$ . oraenv
ORACLE_SID = [ORCL] ? CDB18
The Oracle base remains unchanged with value /u01/app/oracle
$ rman target /
connected to target database: CDB2 (DBID=1434391901)

RMAN> ALTER SYSTEM SET
DB_CREATE_FILE_DEST='/u02/app/oracle/oradata/CDB18/PDB18_IN_CDB1
8';
using target database control file instead of recovery catalog
Statement processed

RMAN> RESTORE FROM PLATFORM 'Linux x86 64-bit'
      USING '/tmp/pdb1.xml'
      FOREIGN pluggable database pdb18_in_cdb18 to new
      FROM BACKUPSET
      '/home/oracle/backup/ORCL/transport_23t5h8r0_1_1';
2> 3> 4> 5>
Starting restore at 15-JUN-18
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=237 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_23t5h8r0_1_1
channel ORA_DISK_1: restoring foreign file 168 to
/u02/app/oracle/oradata/CDB18/PDB18_IN_CDB18/CDB18/datafile/o1_m
f_sysaux_f17x1z61_.dbf
channel ORA_DISK_1: restoring foreign file 167 to
/u02/app/oracle/oradata/CDB18/PDB18_IN_CDB18/CDB18/datafile/o1_m
f_system_f17x1z68_.dbf
channel ORA_DISK_1: restoring foreign file 169 to
/u02/app/oracle/oradata/CDB18/PDB18_IN_CDB18/CDB18/datafile/o1_m
f_undotbs1_f17x1z6d_.dbf
channel ORA_DISK_1: foreign piece
handle=/home/oracle/backup/ORCL/transport_23t5h8r0_1_1
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:08
channel ORA_DISK_1: plugging file 167 for
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB24D9378FB7214E0535410E
D0A6F2F/datafile/o1_mf_system_f17wskhf_.dbf
channel ORA_DISK_1: plugging file 168 for
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB24D9378FB7214E0535410E
D0A6F2F/datafile/o1_mf_sysaux_f17wskhh_.dbf
channel ORA_DISK_1: plugging file 169 for
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB24D9378FB7214E0535410E
D0A6F2F/datafile/o1_mf_undotbs1_f17wskhj_.dbf
channel ORA_DISK_1: plugging file 3 for
/u02/app/oracle/oradata/ORCL/pdb1/ORCL/6EB24D9378FB7214E0535410E
D0A6F2F/datafile/o1_mf_temp_f17wskhk_.dbf

```

Performing import of metadata...

Finished restore at 15-JUN-18

RMAN>

- d. Open the transported PDB.

```
RMAN> ALTER PLUGGABLE DATABASE pdb18_in_cdb18 OPEN;
```

Statement processed

```
RMAN> EXIT
```

```
$
```

- e. Display the HR.EMPLOYEES data.

```
$ sqlplus hr@PDB18_IN_CDB18
Enter password: password

SQL> SELECT count(*) FROM hr.employees;

  COUNT (*)
-
- -
107

SQL> EXIT
$
```

## Practice 13-3: Upgrading a 12.2 CDB to an 18c CDB

---

### Overview

In this practice, you will upgrade Oracle Database 12c `cdb12` to Oracle Database 18c.

### Tasks

1. In session 12c, prepare `cdb12` to be upgraded to Oracle Database 18c.
  - a. Execute the `setup_upgrade_CDB.sh` script to create two regular PDBs, `PDB1_1` and `PDB1_2`, in `cdb12` that will be upgraded with the CDB.

```
$ $HOME/labs/upgrade/setup_upgrade_CDB.sh
...
$
```

- b. Execute the Pre-Upgrade Information Tool in 12.2 `cdb12` by executing the `preupgrd.sql` script.

```
$ . oraenv
ORACLE_SID = [CDB18] ? cdb12
The Oracle base remains unchanged with value /u01/app/oracle
$
$ rm -rf /u01/app/oracle/cfgtoollogs/cdb12/preupgrade
$ cd /u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin
$ $ORACLE_HOME/jdk/bin/java -jar preupgrade.jar TERMINAL TEXT
Report generated by Oracle Database Pre-Upgrade Information Tool
Version
18.0.0.0.0 on 2018-06-15T17:25:37

Upgrade-To version: 18.0.0.0.0

=====
Status of the database prior to upgrade
=====

Database Name: CDB12
Container Name: CDB$ROOT
Container ID: 1
Version: 12.2.0.1.0
Compatible: 12.2.0
Blocksize: 8192
Platform: Linux x86 64-bit
Timezone File: 26
Database log mode: ARCHIVELOG
 Readonly: FALSE
 Edition: EE
```

Oracle Component Current Status	Upgrade Action
-----	-----
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 OLAP API	[to be upgraded] VALID
=====	
BEFORE UPGRADE	
=====	
REQUIRED ACTIONS	
=====	
None	
RECOMMENDED ACTIONS	
=====	
1. Update NUMERIC INITIALIZATION PARAMETERS to meet estimated minimums.	
This action may be done now or when starting the database in upgrade mode	
using the 18.0.0.0.0 ORACLE HOME.	
Parameter	Currently
18.0.0.0.0 minimum	-----
-----	-----
*sga_target	788529152
1070596096	

The database upgrade process requires certain initialization parameters to meet minimum values. The Oracle upgrade process itself has minimum values which may be higher and are marked with an asterisk. After upgrading, those asterisked parameter values may be reset if needed.

2. (AUTOFIXUP) Gather stale data dictionary statistics prior to database

upgrade in off-peak time using:

```
EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;
```

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.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

3. (AUTOFIXUP) Gather statistics on fixed objects prior the upgrade.

None of the fixed object tables have had stats collected.

Gathering statistics on fixed objects, if none have been gathered yet, is

recommended prior to upgrading.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

INFORMATION ONLY

=====

4. To help you keep track of your tablespace allocations, the following  
 AUTOEXTEND tablespaces are expected to successfully EXTEND  
 during the  
 upgrade process.

Tablespace	Size	Min Size For Upgrade
SYSAUX	460 MB	680 MB
SYSTEM	800 MB	1243 MB
TEMP	33 MB	150 MB
UNDOTBS1	70 MB	439 MB

Minimum tablespace sizes for upgrade are estimates.

5. No action needed.

Using default parallel upgrade options, this CDB with 3 PDBs will first

upgrade the CDB\$ROOT, and then upgrade at most 1 PDBs at a time using 2 parallel processes per PDB.

The number of PDBs upgraded in parallel and the number of parallel processes per PDB can be adjusted as described in Database Upgrade Guide.

#### ORACLE GENERATED FIXUP SCRIPT

=====

All of the issues in database CDB12 container CDB\$ROOT which are identified above as BEFORE UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade\_fixups.sql

=====

AFTER UPGRADE

=====

REQUIRED ACTIONS

=====

None

RECOMMENDED ACTIONS

=====

6. Upgrade the database time zone file using the DBMS\_DST package.

The database is using time zone file version 26 and the target 18.0.0.0.0

release ships with time zone file version 31.

Oracle recommends upgrading to the desired (latest) version of the time

zone file. For more information, refer to "Upgrading the Time Zone File

and Timestamp with Time Zone Data" in the 18.0.0.0.0 Oracle Database

Globalization Support Guide.

7. (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.

8. Gather statistics on fixed objects after the upgrade and when there is a

representative workload on the system using the command:

```
EXECUTE DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

This recommendation is given for all preupgrade runs.

Fixed object statistics provide essential information to the Oracle optimizer to help it find efficient SQL execution plans. Those statistics are specific to the Oracle Database release that generates them, and can be stale upon database upgrade.

For information on managing optimizer statistics, refer to the 12.2.0.1 Oracle Database SQL Tuning Guide.

#### ORACLE GENERATED FIXUP SCRIPT

---

All of the issues in database CDB12 container CDB\$ROOT which are identified above as AFTER UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

```
SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.sql
```

Report generated by Oracle Database Pre-Upgrade Information Tool Version 18.0.0.0.0 on 2018-06-15T17:25:52

Upgrade-To version: 18.0.0.0.0

---

Status of the database prior to upgrade

---

Database Name: CDB12  
Container Name: PDB\$SEED  
Container ID: 2  
Version: 12.2.0.1.0  
Compatible: 12.2.0  
Blocksize: 8192  
Platform: Linux x86 64-bit  
Timezone File: 26

Database log mode:	ARCHIVELOG
Readonly:	TRUE
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 OLAP API	[to be upgraded] VALID
=====	
BEFORE UPGRADE	
=====	
REQUIRED ACTIONS	
=====	
None	
RECOMMENDED ACTIONS	
=====	
1. (AUTOFIXUP) Gather stale data dictionary statistics prior to database	
upgrade in off-peak time using:	
EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;	
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.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

2. (AUTOFIXUP) Gather statistics on fixed objects prior the upgrade.

None of the fixed object tables have had stats collected.

Gathering statistics on fixed objects, if none have been gathered yet, is recommended prior to upgrading.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

#### INFORMATION ONLY

---

3. To help you keep track of your tablespace allocations, the following

AUTOEXTEND tablespaces are expected to successfully EXTEND during the

upgrade process.

Tablespace	Size	Min Size For Upgrade
-----	-----	-----
SYSAUX	330 MB	572 MB
SYSTEM	250 MB	689 MB
TEMP	64 MB	150 MB
UNDOTBS1	100 MB	439 MB

Minimum tablespace sizes for upgrade are estimates.

4. No action needed.

Using default parallel upgrade options, this CDB with 1 PDBs will first upgrade the CDB\$ROOT, and then upgrade at most 1 PDBs at a time using 2 parallel processes per PDB.

The number of PDBs upgraded in parallel and the number of parallel

processes per PDB can be adjusted as described in Database Upgrade Guide.

#### ORACLE GENERATED FIXUP SCRIPT

All of the issues in database CDB12 container PDB\$SEED which are identified above as BEFORE UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade\_fixups.sql

#### AFTER UPGRADE

##### REQUIRED ACTIONS

None

##### RECOMMENDED ACTIONS

5. Upgrade the database time zone file using the DBMS\_DST package.

The database is using time zone file version 26 and the target 18.0.0.0.0

releases with time zone file version 31.

Oracle recommends upgrading to the desired (latest) version of the time zone file. For more information, refer to "Upgrading the Time Zone File

and Timestamp with Time Zone Data" in the 18.0.0.0.0 Oracle Database Globalization Support Guide.

6. (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.

7. Gather statistics on fixed objects after the upgrade and when there is a

representative workload on the system using the command:

```
EXECUTE DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

This recommendation is given for all preupgrade runs.

Fixed object statistics provide essential information to the Oracle

optimizer to help it find efficient SQL execution plans. Those

statistics are specific to the Oracle Database release that generates

them, and can be stale upon database upgrade.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

ORACLE GENERATED FIXUP SCRIPT

=====

All of the issues in database CDB12 container PDB\$SEED which are identified above as AFTER UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

```
SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.sql
```

Report generated by Oracle Database Pre-Upgrade Information Tool Version

18.0.0.0.0 on 2018-06-15T17:26:07

Upgrade-To version: 18.0.0.0.0

=====

Status of the database prior to upgrade

=====

Database Name:	CDB12
Container Name:	PDB1_1
Container ID:	3
Version:	12.2.0.1.0
Compatible:	12.2.0
Blocksize:	8192
Platform:	Linux x86 64-bit
Timezone File:	26
Database log mode:	ARCHIVELOG
Readonly:	FALSE
Edition:	EE

Oracle Component Current Status	Upgrade Action
-----	-----
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 OLAP API	[to be upgraded]	VALID

=====

BEFORE UPGRADE

=====

REQUIRED ACTIONS

=====

None

RECOMMENDED ACTIONS

=====

1. (AUTOFIXUP) Gather stale data dictionary statistics prior to database

upgrade in off-peak time using:

```
EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;
```

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.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

2. (AUTOFIXUP) Gather statistics on fixed objects prior the upgrade.

None of the fixed object tables have had stats collected.

Gathering statistics on fixed objects, if none have been gathered yet, is recommended prior to upgrading.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

#### INFORMATION ONLY

=====

3. To help you keep track of your tablespace allocations, the following

AUTOEXTEND tablespaces are expected to successfully EXTEND during the

upgrade process.

Tablespace	Size	Min Size For Upgrade
-----	-----	-----
SYSAUX	350 MB	573 MB
SYSTEM	250 MB	689 MB
TEMP	64 MB	150 MB
UNDOTBS1	100 MB	439 MB

Minimum tablespace sizes for upgrade are estimates.

4. No action needed.

Using default parallel upgrade options, this CDB with 1 PDBs will first

upgrade the CDB\$ROOT, and then upgrade at most 1 PDBs at a time using 2

parallel processes per PDB.

The number of PDBs upgraded in parallel and the number of parallel

processes per PDB can be adjusted as described in Database Upgrade Guide.

#### ORACLE GENERATED FIXUP SCRIPT

=====

All of the issues in database CDB12 container PDB1\_1

which are identified above as BEFORE UPGRADE "(AUTOFIXUP)" can be resolved by

executing the following from within the container

```
SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups.sql
```

```
=====
```

AFTER UPGRADE

```
=====
```

REQUIRED ACTIONS

```
=====
```

None

RECOMMENDED ACTIONS

```
=====
```

5. Upgrade the database time zone file using the DBMS\_DST package.

The database is using time zone file version 26 and the target 18.0.0.0.0

release ships with time zone file version 31.

Oracle recommends upgrading to the desired (latest) version of the time

zone file. For more information, refer to "Upgrading the Time Zone File

and Timestamp with Time Zone Data" in the 18.0.0.0.0 Oracle Database

Globalization Support Guide.

6. (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.

7. Gather statistics on fixed objects after the upgrade and when there is a representative workload on the system using the command:

```
EXECUTE DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

This recommendation is given for all preupgrade runs.

Fixed object statistics provide essential information to the Oracle

optimizer to help it find efficient SQL execution plans. Those

statistics are specific to the Oracle Database release that generates

them, and can be stale upon database upgrade.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

#### ORACLE GENERATED FIXUP SCRIPT

---

All of the issues in database CDB12 container PDB1\_1 which are identified above as AFTER UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

```
SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.sql
```

Report generated by Oracle Database Pre-Upgrade Information Tool Version

18.0.0.0.0 on 2018-06-15T17:26:22

Upgrade-To version: 18.0.0.0.0

---

Status of the database prior to upgrade		
=====		
Database Name:	CDB12	
Container Name:	PDB1_2	
Container ID:	4	
Version:	12.2.0.1.0	
Compatible:	12.2.0	
Blocksize:	8192	
Platform:	Linux x86 64-bit	
Timezone File:	26	
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 OLAP API	[to be upgraded]	VALID
=====		
BEFORE UPGRADE		
=====		
REQUIRED ACTIONS		
=====		
None		
RECOMMENDED ACTIONS		

=====

1. (AUTOFIXUP) Gather stale data dictionary statistics prior to database

upgrade in off-peak time using:

```
EXECUTE DBMS_STATS.GATHER_DICTIONARY_STATS;
```

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.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

2. (AUTOFIXUP) Gather statistics on fixed objects prior the upgrade.

None of the fixed object tables have had stats collected.

Gathering statistics on fixed objects, if none have been gathered yet, is

recommended prior to upgrading.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

#### INFORMATION ONLY

=====

3. To help you keep track of your tablespace allocations, the following

AUTOEXTEND tablespaces are expected to successfully EXTEND during the

upgrade process.

Tablespace	Size	Min Size For Upgrade
------------	------	-------------------------

SYSAUX	350 MB	573 MB	
SYSTEM	250 MB	689 MB	
TEMP	64 MB	150 MB	
UNDOTBS1	100 MB	439 MB	

Minimum tablespace sizes for upgrade are estimates.

4. No action needed.

Using default parallel upgrade options, this CDB with 1 PDBs will first upgrade the CDB\$ROOT, and then upgrade at most 1 PDBs at a time using 2 parallel processes per PDB.

The number of PDBs upgraded in parallel and the number of parallel processes per PDB can be adjusted as described in Database Upgrade Guide.

ORACLE GENERATED FIXUP SCRIPT

=====

All of the issues in database CDB12 container PDB1\_2 which are identified above as BEFORE UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade\_fixups.sql

=====

AFTER UPGRADE

=====

REQUIRED ACTIONS

=====

None

RECOMMENDED ACTIONS

=====

5. Upgrade the database time zone file using the DBMS\_DST package.

The database is using time zone file version 26 and the target 18.0.0.0.0 release ships with time zone file version 31.

Oracle recommends upgrading to the desired (latest) version of the time zone file. For more information, refer to "Upgrading the Time Zone File and Timestamp with Time Zone Data" in the 18.0.0.0.0 Oracle Database Globalization Support Guide.

6. (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.

7. Gather statistics on fixed objects after the upgrade and when there is a representative workload on the system using the command:

```
EXECUTE DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

This recommendation is given for all preupgrade runs.

Fixed object statistics provide essential information to the Oracle optimizer to help it find efficient SQL execution plans. Those

statistics are specific to the Oracle Database release that generates them, and can be stale upon database upgrade.

For information on managing optimizer statistics, refer to the 12.2.0.1

Oracle Database SQL Tuning Guide.

#### ORACLE GENERATED FIXUP SCRIPT

=====

All of the issues in database CDB12 container PDB1\_2 which are identified above as AFTER UPGRADE "(AUTOFIXUP)" can be resolved by executing the following from within the container

SQL>@/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade\_fixups.sql

#### =====

#### PREUPGRADE SUMMARY

#### =====

/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade.log

/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade\_fixups.sql

/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade\_fixups.sql

Execute fixup scripts across the entire CDB:

#### **Before upgrade:**

1. Execute preupgrade fixups with the below command

```
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -  
I$ORACLE_HOME/rdbms/admin $ORACLE_HOME/rdbms/admin/catcon.pl -l  
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/ -b preup_cdb12  
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups.sql
```

2. Review logs under

/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/

**After the upgrade:**

1. Execute postupgrade fixups with the below command

```
$ORACLE_HOME/perl/bin/perl -I$ORACLE_HOME/perl/lib -  
I$ORACLE_HOME/rdbms/admin $ORACLE_HOME/rdbms/admin/catcon.pl -l  
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/ -b postup_cdb12  
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups.  
sql
```

2. Review logs under

```
/u01/app/oracle/cfgtoollogs/cdb12/preupgrade/
```

```
Preupgrade complete: 2018-06-15T17:26:27
```

```
$
```

*Q/ For which containers in the CDB does the Pre-Upgrade Information Tool provide actions?*

**A/ 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 cdb12.

```
$ cd /u01/app/oracle/cfgtoollogs/cdb12/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 cdb12 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.

```
$ cd /u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'CDB$ROOT' -b
preupgrade
$ORACLE_BASE/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups_CDB_
ROOT.sql

catcon::set_log_file_base_path: ALL catcon-related output will
be written to
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade_
catcon_30230.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade*
.log] files for output generated by scripts
```

```
catcon::set_log_file_base_path: catcon: See  
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade_*  
.lst] files for spool files, if any
```

```
catcon.pl: completed successfully
```

```
$
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'PDB$SEED' -b  
preupgrade  
$ORACLE_BASE/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups_PDB_  
SEED.sql
```

```
catcon::set_log_file_base_path: ALL catcon-related output will  
be written to  
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade_  
catcon_30829.lst]
```

```
catcon::set_log_file_base_path: catcon: See  
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade*  
.log] files for output generated by scripts
```

```
catcon::set_log_file_base_path: catcon: See  
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade_*  
.lst] files for spool files, if any
```

```
catcon.pl: completed successfully
```

```
$
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'PDB1_1' -b preupgrade  
$ORACLE_BASE/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups_PDB1_1.sql
```

```
catcon::set_log_file_base_path: ALL catcon-related output will  
be written to  
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade_  
catcon_31299.lst]
```

```
catcon::set_log_file_base_path: catcon: See  
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade*  
.log] files for output generated by scripts
```

```
catcon::set_log_file_base_path: catcon: See  
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade_*  
.lst] files for spool files, if any
```

```
catcon.pl: completed successfully
```

```
$
```

```
$ $ORACLE_HOME/perl/bin/perl catcon.pl -c 'PDB1_2' -b preupgrade
$ORACLE_BASE/cfgtoollogs/cdb12/preupgrade/preupgrade_fixups_PDB1
_2.sql

catcon::set_log_file_base_path: ALL catcon-related output will
be written to
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade_
catcon_31747.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade*
.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/preupgrade_*
.lst] files for spool files, if any

catcon.pl: completed successfully
$
```

e. Back up cdb1.

```
$ rman target /

connected to target database: CDB12 (DBID=1996891072)

RMAN> ALTER SYSTEM SET db_recovery_file_dest_size='40G'
      SCOPE = both;
2>
using target database control file instead of recovery catalog
Statement processed

RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
...
RMAN> EXIT
$
```

f. Prepare the PDBs priority list. You want pdb1\_2 to be upgraded before pdb1\_1. Create a SQL script containing the following code. The SELECT statement creates the PDBs list, sorted and assigned with 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.

```
$ sqlplus / AS SYSDBA

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

2. Switch to Session 18c to open cdb12 in the Oracle Database 18c environment in UPGRADE mode.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
The Oracle base remains unchanged with value /u01/app/oracle
$ export ORACLE_SID=cdb12
$ cp /u01/app/oracle/product/12.2.0/dbhome_1/dbs/spfilecdb12.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 cdb12.

```
$ mkdir /tmp/upgrade_cdb12
$ cd $ORACLE_HOME/rdbms/admin
$ $ORACLE_HOME/perl/bin/perl catctl.pl -l /tmp/upgrade_cdb12 -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_cdb12
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: [18.0.0.0]
    STATUS: [Production]
    BUILD: [RDBMS_18.1.0.0.0_LINUX.X64_180103.1]

/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/orahome =
[/u01/app/oracle/product/18.1.0/dbhome_1]
/u01/app/oracle/product/18.1.0/dbhome_1/bin/orabasehome =
[/u01/app/oracle/product/18.1.0/dbhome_1]
catctlGetOrabase = [/u01/app/oracle/product/18.1.0/dbhome_1]
```

```

Analyzing file
/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/catupgrd.sql

Log file directory = [/tmp/upgrade_cdb12]

catcon::set_log_file_base_path: ALL catcon-related output will
be written to [/tmp/upgrade_cdb12/catupgrd_catcon_32464.lst]

catcon::set_log_file_base_path: catcon: See
[/tmp/upgrade_cdb12/catupgrd*.log] files for output generated by
scripts

catcon::set_log_file_base_path: catcon: See
[/tmp/upgrade_cdb12/catupgrd_*_.lst] files for spool files, if
any

Number of Cpus          = 2
Database Name           = cdb12
DataBase Version        = 12.2.0.1.0
PDB Parallel SQL Process Count = [2] is higher or equal to CPU
Count = [2]
Concurrent PDB Upgrades defaulting to CPU Count [2]
Parallel SQL Process Count (PDB)      = 2
Parallel SQL Process Count (CDB$ROOT) = 4
Concurrent PDB Upgrades            = 2
Generated PDB Inclusion:[PDB$SEED PDB1_2 PDB1_1]
Components in [CDB$ROOT]

    Installed [APS CATALOG CATJAVA CATPROC CONTEXT DV JAVAVM OLS
ORDIM OWM SDO XDB XML XOO]
Not Installed [APEX EM MGW ODM RAC WK]

-----
Phases [0-108]             Start Time:[2018_06_15 17:53:57]
Container Lists Inclusion:[CDB$ROOT] Exclusion:[NONE]
-----
***** Executing Change Scripts *****
Serial Phase #:0 [CDB$ROOT] Files:1 Time: 45s
***** Catalog Core SQL *****
Serial Phase #:1 [CDB$ROOT] Files:5 Time: 104s
Restart Phase #:2 [CDB$ROOT] Files:1 Time: 1s
***** Catalog Tables and Views *****
Parallel Phase #:3 [CDB$ROOT] Files:19 Time: 26s
Restart Phase #:4 [CDB$ROOT] Files:1 Time: 0s
***** Catalog Final Scripts *****

```

```
Serial Phase #:5 [CDB$ROOT] Files:7 Time: 21s
***** Catproc Start *****
...
***** Migration *****
Serial Phase #:100 [PDB1_1] Files:1 Time: 1s
*** End PDB Application Upgrade Pre-Shutdown ***
Serial Phase #:101 [PDB1_1] Files:1 Time: 1s
Serial Phase #:102 [PDB1_1] Files:1 Time: 4s
Serial Phase #:103 [PDB1_1] Files:1 Time: 9s
***** Post Upgrade *****
Serial Phase #:104 [PDB1_1] Files:1 Time: 4s
***** Summary report *****
Serial Phase #:105 [PDB1_1] Files:1 Time: 2s
*** End PDB Application Upgrade Post-Shutdown **
Serial Phase #:106 [PDB1_1] Files:1 Time: 1s
Serial Phase #:107 [PDB1_1] Files:1 Time: 1s
Serial Phase #:108 [PDB1_1] Files:1 Time: 0s
```

```
-----
Phases [0-108] End Time:[2018_06_15 19:44:36]
Container Lists Inclusion:[PDB1_1] Exclusion:[NONE]
-----
```

Grand Total Time: 1448s [PDB1\_1]

LOG FILES: (/tmp/upgrade\_cdb12/catupgrdpdb1\_1\*.log)

Upgrade Summary Report Located in:  
/tmp/upgrade\_cdb12/upg\_summary.log

Total Upgrade Time: [0d:0h:24m:8s]

Time: 2093s For CDB\$ROOT  
Time: 4551s For PDB(s)

Grand Total Time: 6644s

LOG FILES: (/tmp/upgrade\_cdb12/catupgrd\*.log)

Upgrade Summary Report Located in:  
/tmp/upgrade\_cdb12/upg\_summary.log

```
Grand Total Upgrade Time: [0d:1h:50m:44s]
$
```

*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
2017-01-09T06:26:45.971965+00:00
...
Pluggable database PDB1_2 opened in upgrade mode
2017-01-09T06:26:54.475684+00:00
Pluggable database PDB1_1 opened in upgrade mode
PDB1_2(5):Completed: alter pluggable database PDB1_2 open
upgrade
2017-01-09T06:26:54.644481+00:00
PDB1_1(3):Completed: alter pluggable database PDB1_1 open
upgrade
2017-01-09T06:27:09.541304+00:00
```

**If you want to know the scripts and commands performed during the upgrade process, read the /tmp/upgrade\_cdb12/catupgrd\*.log log files.**

- Finally execute the postupgrade\_fixups.sql and utlrp.sql scripts.

```
$ . oraenv
ORACLE_SID = [cdb12] ? cdb12
The Oracle base remains unchanged with value /u01/app/oracle
$ export ORACLE_HOME=/u01/app/oracle/product/18.1.0/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/cdb12/preupgrade/postupgrade_fixups_CDB
_ROOT.sql
catcon::set_log_file_base_path: ALL catcon-related output will
be written to
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
_catcon_6612.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/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/cdb12/preupgrade/postupgrade_fixups_PDB
_SEED.sql
catcon::set_log_file_base_path: ALL catcon-related output will
be written to
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
_catcon_6895.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/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_1' -b
postupgrade
$ORACLE_BASE/cfgtoollogs/cdb12/preupgrade/postupgrade_fixups_PDB
1_1.sql
catcon::set_log_file_base_path: ALL catcon-related output will
be written to
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
_catcon_7204.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/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/cdb12/preupgrade/postupgrade_fixups_PDB
1_2.sql
catcon::set_log_file_base_path: ALL catcon-related output will
be written to
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
_catcon_7499.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/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::set_log_file_base_path: ALL catcon-related output will
be written to
```

```
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
_catcon_7806.lst]

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.log] files for output generated by scripts

catcon::set_log_file_base_path: catcon: See
[/u01/app/oracle/product/18.1.0/dbhome_1/rdbms/admin/postupgrade
*.lst] files for spool files, if any

catcon.pl: completed successfully

$
```

*Q/ Is there another method to upgrade CDBs?*

**A/ DBUA can upgrade a CDB.**

**A2/ Data Pump export/import can also upgrade a 12.2 CDB into an 18c CDB with the FULL Transportable or FULL conventional export/import.**

5. Update the `cdb12` entry in `/etc/oratab` to set the `ORACLE_HOME` to 18c.

```
cdb12:/u01/app/oracle/product/12.2.0/dbhome_1:N
```



```
cdb12:/u01/app/oracle/product/18.1.0/dbhome_1:N
```

6. Create the new `cdb12` password file in the Oracle Database 18c environment.

```
$ cd /u01/app/oracle/product/18.1.0/dbhome_1/dbs
$ export PATH=/usr/lib64/qt-
3.3/bin:/usr/NX/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin
:/usr/sbin:/sbin:/home/oracle/bin:/u01/app/oracle/product/18.1.0
/dbhome_1/bin
$ orapwd file=orapwcdb12 entries=5 password=password
$
```

7. Restart the database and open all PDBs. Check that application data is accessible.

```
$ . oraenv
ORACLE_SID = [cdb12] ? cdb12
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> 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> ALTER SYSTEM SET local_listener='' SCOPE=BOTH;

System altered.

SQL> CONNECT system@PDB1_1
Enter password: password
Connected.
SQL> SELECT * FROM pdb1_1_user.smalltab;

LABEL
-----
DATA FROM source_user.smalltab

10 rows selected.

SQL> CONNECT system@PDB1_2
Enter password: password
Connected.
SQL> SELECT count(*) FROM pdb1_2_user.smalltab;

COUNT (*)
-----
```

```
-----  
10  
SQL> EXIT  
$
```

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangli@baylorhealth.edu) has a non-transferable license  
to use this Student Guide.